CHARRM – Characterizing Hardware Leakage in the Robust Random Probing Model

Armand Schinkel[1], Pascal Sasdrich[1]

Ruhr-University Bochum, Bochum, Germany
firstname.lastname@rub.de

Abstract. Masking is a prevalent countermeasure against passive physical implementation attacks such as Side-Channel Analysis (SCA). Despite its theoretical soundness, the practical application of masking is often fraught with complexity and prone to errors, underscoring the necessity for automated security verification. This paper presents a comprehensive methodology for efficient security verification in the (g, t, c)-*robust random* (p, ε)-*probing model*. In particular, we advance the state-of-the-art concept of leakage characteristics to account for robust probe extension schemes across arbitrary circuit structures, enabling efficient leakage probability bound computation for glitch-extended and transition-extended robust probing scenarios. By further linking the leakage characteristic to the Set of Input Shares (SIS) function, we extend our verification methodology to established composability notions. Our automated reasoning framework demonstrates practicality and superior performance and accuracy for larger circuits in comparative evaluations with ironMask as state-of-the-art competitor. For the first time, our method allows for the exact verification of larger, composed circuit structures, validated by corresponding case studies. Consequently, this work advances the state-of-the-art in automated hardware security verification, providing a versatile tool for evaluating the security of masked cryptographic implementations against SCA in the presence of transient physical effects.

Keywords: Side-Channel Analysis · Formal Verification · Random Probing Model · Robust Probing Model · Leakage Characteristic · Fourier-Hadamard Transform

1 Introduction

Side-Channel Analysis (SCA) [24, 25] poses a significant threat to cryptographic systems by exploiting physical characteristics such as runtime behavior [24], instantaneous power consumption [25], or electromagnetic (EM) emanations [18] that leak sensitive information.

In response, *masking* [15] has emerged as the predominant information-theoretically sound countermeasure. Based on secret sharing, it splits sensitive data into shares that are processed independently, i. e., for d-th order Boolean masking a variable x is divided into $s = d + 1$ shares $\langle x_0, x_1, \ldots, x_d \rangle$ with $x = x_0 \oplus x_1 \oplus \ldots \oplus x_d$, such that only the complete set of shares reconstructs the secret while any strict subset appears uniformly random. Given the inherent noisiness of observations, the attack complexity then grows exponentially in d.

As the complexity of cryptographic functions increases, so does the effort required for efficient masking, which has led the community to develop composable building blocks known as *gadgets* that encapsulate fundamental operations in a masked form [2, 3, 14, 21, 23]. Their development, however, has been accompanied by a significant number of vulnerabilities, as many proposed schemes were demonstrated to be insecure [26], which

renders the design of correct and secure masking schemes a delicate and error-prone process that necessitates rigorous validation.

Formal analysis of security properties has therefore gained importance, where it is crucial to distinguish *security* from *composability properties*, with the latter being auxiliary properties that often imply the former. Security is predominantly demonstrated within the threshold d-probing model [20], while well-known composability properties for Boolean masking include Non-Interference (NI) [2], Strong Non-Interference (SNI) [3], and Probe-Isolating Non-Interference (P-INI) [14]. Tools for automated security evaluation [1, 4, 7, 8, 11, 13, 19, 22] enable a more rigorous and efficient assessment of masked gadgets and the validation of security claims.

To further advance the formal analysis of hardware masking schemes, it is imperative to consider more realistic and robust security models that account for the complexities of physical implementations. The (g, t, c)-*robust threshold d-probing model* [17] extends the basic *threshold d-probing model* by incorporating physical phenomena such as glitches, transitions, and couplings, thereby providing a more accurate representation of real-world scenarios. Moreover, the *random* (p, ε)-*probing model* [16, 20] serves as a bridge between theoretical analysis and practical security, facilitating the formal evaluation of hardware masking schemes towards the noisy-leakage models. This approach enables the consideration of a broader range of effects, including horizontal attacks, which are often overlooked in simplistic security models.

However, while the (g, t, c)-*robust random* (p, ε)-*probing model*[1] exists in theoretical considerations, the practical tools available for formal verification are typically limited to either (g, t, c)-*robust threshold d-probing* or *standard* (p, ε)-*random probing*, but not both. Here, ironMask [8] appears to be a notable exception, although the specific formal methods for glitch-extended analysis remain undocumented. To address this gap, this research aims to provide a formal analysis and description of the (g, t, c)-robust random (p, ε)-probing model, along with a comprehensive methodology to characterize the leakage probability bounds of hardware circuits in this context. Furthermore, the proposed methodology is extended to reason about composability properties within the robust random probing model, drawing on the concept of the Set of Input Shares (SIS) function (introduced in [8]) to ensure a thorough evaluation of masking gadgets and cryptographic implementations.

Contributions. This work presents a comprehensive and optimized formal methodology for security verification in both the (g, t, c)-robust threshold d-probing and (g, t, c)-robust random (p, ε)-probing models. We extend the notion of leakage characteristics to capture probe extension schemes over arbitrary circuit structures and introduce efficient procedures for leakage probability bound computation that uniformly cover both glitch-, transition- and coupling-extended verification, together with proofs of their correctness. Building on this, we connect the construction of leakage characteristics with the SIS function of [8], thereby extending our framework to established composability notions in both models.

To demonstrate the practicality of these formal results, we further provide an implementation for verification in the $(g, t, 0)$-robust random probing model. Using this implementation, we conduct a comparative evaluation against ironMask, the sole existing competitor capable of verifying Random Probing Security (RPS) in a robust setting. Since ironMask supports the glitch extension only, this comparison is necessarily restricted to the $(1, 0, 0)$-robust random probing model, while for the transition extension no competitor exists. Further, for the first time, we report additional experiments including Random Probing Composability (RPC) and RPS on compositions and larger masked hardware circuits in the $(1, 0, 0)$- (glitch), the $(0, 1, 0)$- (transition) and the $(1, 1, 0)$-robust (glitch+transition)

[1] For the sake of readability, we refer to the (g, t, c)-robust random (p, ε)-probing model as the *robust random probing model* and make the probe extensions explicit whenever the concrete instantiation matters, e. g., the $(1, 0, 0)$-robust random probing model.

Table 1: Common notations used throughout this paper.

	Notation	Description
Circuit	n_i	Number of unshared inputs to a circuit
	n_r	Number of randomness inputs to a (masked) circuit
	n_o	Number of unshared outputs of a circuit
	C	Circuit
	A_C	Access function to circuit C
	$\mathcal{W}$	Set of wires
	$\mathcal{G}$	Set of gates
Math	s	Number of shares used by a masking scheme
	d	Probing security order
	$\mathcal{Q}$	Set of probes
	q_i	Probe on wire w_i
	p	Leakage rate $\in \mathcal{D}$
	ε	Leakage probability bound
	L_{Ex}	Leakage characteristic under extension Ex

setting to assess the versatility and scalability of our approach.

Outline. Section 2 summarizes foundational concepts, including the circuit model and Boolean masking. Section 3 introduces our methodology for efficient leakage probability bound computation, addressing glitch-extended, transition-extended, and coupling-extended scenarios, as well as compositional leakage within both (robust) threshold and random probing models. Section 4 details our automated reasoning framework, highlighting its practical implementation using Binary Decision Diagrams (BDDs), Multi-Terminal Binary Decision Diagrams (MTBDDs), and the Fourier-Hadamard Transform (FHT), along with key design choices and optimizations. Section 5 presents case studies that compare our approach with ironMask in the glitch-extended setting and assess the performance on larger and composed masked hardware circuit structures in the glitch-, the transition- and the glitch+transition-extended setting. Finally, Section 6 concludes our work.

2 Preliminaries

2.1 General Notation

Throughout this work, we use consistent notation, i. e., calligraphic font for sets (e. g., $\mathcal{S}$), lower case sans serif font for functions (e. g., f) and upper case sans serif font for vectored functions (e. g., F). Further, for a vectored function F that maps $x \mapsto \mathbb{F}_2^n$ and an index set $\mathcal{I}$, we denote $\mathsf{F}|_\mathcal{I}$ to be the subfunction of F with the coordinates in $\mathcal{I}$, which is $(\mathsf{F}_i)_{i\in\mathcal{I}}$. For a function $\mathsf{f}\colon \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, we also write $\mathsf{f}_x(y)$ which is equivalent to $\mathsf{f}(x, y)$ with fixed x. We call a set mapping f additive iff $\mathsf{f}(\mathcal{X} \cup \mathcal{Y}) = \mathsf{f}(\mathcal{X}) \cup \mathsf{f}(\mathcal{Y})$. Other common notation can be found in Table 1.

2.2 Circuit Model

To formally reason about security of hardware circuits, we model any deterministic logic circuit as Direct Acyclic Graph (DAG) $\mathsf{C} = (\mathcal{G}, \mathcal{W})$. Each vertex $g \in \mathcal{G} = \{\mathsf{inv}, \mathsf{and}, \mathsf{nand}, \mathsf{or}, \mathsf{nor}, \mathsf{xor}, \mathsf{xnor}, \mathsf{reg}\}$ represents a gate, and each edge $w \in \mathcal{W}$ represents a wire that connects two gates. We also recall the following definitions.

Definition 1 (Directed path)**.** Let $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ be a DAG and let $u, v \in \mathcal{G}\colon u \neq v$ be vertices. We write $u \to v$ iff $(u, v) \in \mathcal{W}$, and $u \to^* v$ iff there exists a directed path from u to v.

Definition 2 (Predecessor and Successor)**.** Let $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ be a DAG and let $v \in \mathcal{G}$ be a vertex. $\mathsf{pred}(v) := \{u \in \mathcal{G} \colon u \to^* v\}$ contains all vertices that have a directed path to v, called predecessors. Similarly, we define $\mathsf{succ}(v) := \{u \in \mathcal{G} \colon v \to^* u\}$, which contains all vertices to which v has a directed path, which we call successors.

Definition 3 (Circuit)**.** A circuit $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ is a DAG implementing a vectored Boolean function $\mathsf{F}_\mathsf{C} \colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_o}$ with the coordinate functions $\mathsf{f}_\mathsf{C}^1, ..., \mathsf{f}_\mathsf{C}^{n_o}$. Consequently, each gate (resp. wire) represents a function $\mathsf{f}_g^\mathsf{C} \colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2$ which is defined by the subgraph $\mathsf{C}_g \subseteq \mathsf{C}$ with leaf g. Each input vertex (i. e., $v \in \mathcal{G} \colon \mathsf{pred}(v) = \emptyset$) has a label from $\{x_1, \ldots, x_{n_i}\}$ and each output vertex (i. e., $v \in \mathcal{G} \colon \mathsf{succ}(v) = \emptyset$) has a label from $\{y_1, \ldots, y_{n_o}\}$. All other vertices take a label from $\mathcal{G}$. Additionally, C has the following properties:

1. Each input vertex has in-degree 0 and each output vertex has in-degree 1 and out-degree 0.
2. Each vertex with a label in $\{\mathsf{and}, \mathsf{nand}, \mathsf{or}, \mathsf{nor}, \mathsf{xor}, \mathsf{xnor}\}$ has in-degree 2.
3. Each vertex with a label in $\{\mathsf{inv}, \mathsf{reg}\}$ has in-degree 1.

To access the circuit internals and observe any gate output, we further define the access to the circuit, provided a set of (observed) gates.

Definition 4 (Circuit Access)**.** Let $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ be a circuit implementing a vectored Boolean function $\mathsf{F}_\mathsf{C} \colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_o}$. Then $\mathsf{A}_\mathsf{C} \colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{|\mathcal{W}|}$ is the access to C, where each coordinate of A_C corresponds to the function of the subgraph $\mathsf{C}_w \subseteq \mathsf{C}$ under given input $x \in \mathbb{F}_2^{n_i}$. More concretely, $\mathsf{A}_\mathsf{C}(x) = (w_0, ..., w_{|\mathcal{W}|-1})$ with $w_i \in \mathbb{F}_2$ carrying the value of gate w_i in the circuit when provided with input x.

We denote the circuit access with respect to a probeset $\mathcal{Q}$ as $\mathsf{A}_\mathsf{C}|_\mathcal{Q}$, which captures the internal circuit state at the specified probe locations under the provided input x.

2.3 Boolean Masking

In *Boolean masking* [15], each secret value $x_i \in \mathbb{F}_2$ is decomposed into s shares $\mathsf{Sh}(x_i) = \langle x_{i,0}, x_{i,1}, \ldots, x_{i,s-1} \rangle$ with $x_i = \bigoplus_{j=0}^{s-1} x_{i,j}$, such that any strict subset $\mathcal{X} \subset \mathsf{Sh}(x_i)$ is statistically independent of x_i. Accordingly, any circuit $\mathsf{C} \colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_o}$ is transformed into a masked circuit $\mathsf{C}' \colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$ that operates on the shared representations of the inputs. Formally, the masked circuit transformation is given by a triplet $(\mathsf{Enc}, \mathsf{C}', \mathsf{Dec})$, where the encoding function Enc maps the unshared inputs together with $(s-1)n_i$ uniformly random bits to a valid input sharing, the masked circuit C' consumes n_r additional random bits to compute a valid output sharing, and the decoding function Dec recovers the unshared outputs.

We define $\mathsf{H} \colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i + n_r} \to \mathbb{F}_2^{|\mathcal{W}|}$ to be the encoded access to a masked circuit C' that explicitly excludes wires in Enc and Dec. Formally, the access to the masked circuit is defined as $\mathsf{H} := \mathsf{A}_\mathsf{C} \circ \mathsf{Enc}$ such that $\mathsf{H}(x, r_0 \| r_1) = \mathsf{A}_\mathsf{C}(\mathsf{Enc}(x, r_0), r_1)$ for $x \in \mathbb{F}_2^{n_i}, r_0 \in \mathbb{F}_2^{(s-1)n_i}, r_1 \in \mathbb{F}_2^{n_r}$.

2.4 Adversary Models

To reason about the security of masked circuits, we first recall established security models and definitions that formally define attacker capabilities and limitations. These models aim to closely reflect reality, whilst providing abstractions enabling efficient formal reasoning.

2.4.1 Threshold d-Probing Security.

Adversaries in the ISW threshold d-probing model [20] may probe up to d wires (resp. gates [6]) of a circuit, which are determined before execution. The circuit is threshold d-probing secure if no observed probeset leaks any information about the secret.

Definition 5 (Leakage [4])**.** Let $\mathsf{Enc}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \to \mathbb{F}_2^{n_i \cdot s}$ be an encoding function and $\mathsf{A}_{\mathsf{C}'}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{|\mathcal{W}|}$ be the access to a masked circuit $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ that implements a function $\mathsf{F}_{\mathsf{C}'}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$. Further, let $\mathsf{H} := \mathsf{A}_{\mathsf{C}'} \circ \mathsf{Enc}$ be the encoded access to C', which is a function from $\mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \times \mathbb{F}_2^{n_r}$ to $\mathbb{F}_2^{|\mathcal{W}|}$. We define the leakage characteristic $\mathsf{L}\colon \mathbb{F}_2^{|\mathcal{W}|} \to \{0,1\}$, which describes leakage of the probeset $\mathcal{Q}$ for uniform randomly chosen $r \in \mathbb{F}_2^{(s-1)n_i} \times \mathbb{F}_2^{n_r}$, as follows:

$$\mathsf{L}(\mathcal{Q}) = \begin{cases} 0 & \text{if } \forall x \in \mathbb{F}_2^{n_i}, y \in \mathbb{F}_2^{|\mathcal{Q}|}\colon \mathsf{Pr}[\mathsf{H}|_{\mathcal{Q}}(x,r) = y] = \mathsf{Pr}[\mathsf{H}|_{\mathcal{Q}}(0,r) = y] \\ 1 & \text{else,} \end{cases}$$

A circuit is threshold d-probing secure, iff for all possible probesets with up to d probes, the leakage characteristic is zero. We can now use this definition to define threshold d-probing security formally.

Definition 6 (Threshold d-Probing Security)**.** Let $\mathsf{Enc}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \to \mathbb{F}_2^{n_i \cdot s}$ be an encoding function and let $\mathsf{C}'\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$ be a masked circuit. C' is threshold d-probing secure iff $\forall \mathcal{Q} \subseteq \mathcal{W}\colon |\mathcal{Q}| \leq d$ it holds that $\mathsf{L}(\mathcal{Q}) = 0$.

2.4.2 (g,t,c)-Robust Threshold d-Probing Security

The threshold d-probing model, though being theoretically sound, has a few shortcomings in practice as it does not consider physical defaults such as combinatorial recombinations (glitches), memory recombinations (transitions), and routing recombinations (couplings). To address these shortcomings, the (g,t,c)-robust threshold d-probing model has been introduced by Faust et *al.* [17], which extends the threshold d-probing model to capture these physical effects.

Modeling Combinatorial Recombinations (Glitch Extension). The glitch extension can be modeled as a function that maps each probe within a probeset to its respective stable (clock-synchronous) stage inputs (i. e., registers or primary circuit input gates). Formally we define the probe glitch extension as follows:

Definition 7 (Probe Glitch Extension)**.** Let $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ be a masked circuit and $\mathcal{Q} \subseteq \mathcal{W}$ be a probeset. We define the probe glitch extension as

$$\begin{aligned} \mathsf{GEx}^q(\mathcal{Q}) := \{ & w \in \mathcal{W}, \exists w' \in \mathcal{Q}\colon \\ & (w = \mathsf{reg} \vee w = \mathsf{in}) \\ & \wedge (w \in \mathsf{pred}(w') \vee w = w') \\ & \wedge (\nexists w'' \in \mathsf{succ}(w)\colon w'' = \mathsf{reg})\} \cup \mathcal{Q}. \end{aligned}$$

Modeling Memory Recombinations (Transition Extension). Modeling the transition extension consists of mapping register probes to their respective inputs, which means that an attacker may observe two values at the same probe location over two consecutive clock cycles.

Definition 8 (Probe Transition Extension)**.** For a masked circuit $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ and the probeset $\mathcal{Q} \subseteq \mathcal{W}$, the probe transition extension is defined as:

$$\mathsf{TEx}^q(\mathcal{Q}) := \{w \in \mathcal{W}, \exists w' \in \mathcal{Q}\colon (w' = \mathsf{reg}) \wedge (w \to w')\} \cup \mathcal{Q}.$$

Modeling Routing Recombinations (Coupling Extension). Lastly, the coupling extension has an admittedly loose definition in this context, which is due to the fact that our circuit abstraction does not encompass any gate or wire placement information, and thus does not provide any metric of adjacency. Still, we provide a formal definition that theoretically allows to model coupling effects (given a notion of adjacency).

Definition 9 (Probe Coupling Extension)**.** Let $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ be a masked circuit and let $\mathcal{Q} \subseteq \mathcal{W}$ be a probeset. The coupling extension of $\mathcal{Q}$ with respect to C' is defined as

$$\mathsf{CEx}^q(\mathcal{Q}) := \{w \in \mathcal{W}, \exists w' \in \mathcal{Q} \colon \Phi(w, w') = 1\} \cup \mathcal{Q},$$

where Φ is any Boolean predicate on the specified probe and a potential extension probe.

We show in Lemma 1 that these probe extension definitions fulfill additivity, which is the structural property that our construction in Section 3 requires.

The (g, t, c)-robust threshold d-probing model thus encompasses all physical defaults, where g and t take values in $\{0, 1\}$, disabling (resp. enabling) glitch/transition extension, and where c takes values in $\mathbb{N} \leq |\mathcal{W}| - 1$, allowing an attacker to observe c adjacent wires to each wire in the probeset. A circuit is secure in the (g, t, c)-robust threshold d-probing model if an attacker can not gain any information about the secret by probing d wires. Notably, if any of the probe extensions is active, the effective amount of probes the attacker gets may be larger than d, which makes this model more conservative than the standard threshold d-probing model. To denote the leakage characteristic under any probe extension, we write L_{Ex}, where $\mathsf{Ex} \subseteq \{\mathsf{GEx}^q, \mathsf{TEx}^q, \mathsf{CEx}^q\}$ and L_{id} for the leakage characteristic where the probe extension is the identity.

2.4.3 (p, ε)-Random Probing Security.

The (p, ε)-random probing model was originally introduced as means to formulate a reduction proof from security in the noisy leakage model to security in the threshold d-probing model [16,27], and assumes that an adversary observes a probe q with probability p and fails to observe that probe with probability $1 - p$. Formally, using similar notation as described for the threshold d-probing model, we can express (p, ε)-RPS in terms of the leakage characteristic.

Definition 10 ((p, ε)-Random Probing Security)**.** Let $\mathsf{Enc} \colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \to \mathbb{F}_2^{n_i \cdot s}$ be an encoding function and let $\mathsf{C}' \colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$ be a masked circuit. Further, let p describe the probe leakage rate for each wire. C' is (p, ε)-random probing secure, iff the following holds:

$$\sum_{i=1}^{|\mathcal{W}|} \sum_{\mathcal{Q} \subseteq \mathcal{W}, |\mathcal{Q}| = i} \mathsf{L}(\mathcal{Q}) \cdot p^i \cdot (1-p)^{(|\mathcal{W}| - i)} \leq \varepsilon.$$

Intuitively, we scale each possible probeset $\mathcal{Q}$ by its leakage rate and measure whether it exceeds the leakage probability bound ε. Notably, this security notion can also be extended to the robust setting, yielding the (g, t, c)-robust random (p, ε)-probing model.

2.5 Efficient Leakage Computation

As shown by Beierle et *al.* [4], there exists a link between the FHT over preimage differences (for different unshared inputs) and the leakage characteristic. Hence, using the leakage characteristic for security verification has the advantage of being efficiently computable, e. g., using the Fast Fourier-Hadamard Transform (FFHT) over BDD representations of circuit functions.

We define the support of a set $\mathcal{S} \subseteq \mathbb{F}_2^n$ as $\mathsf{supp}(S) := \{i \colon \mathcal{S}_i = 1\}$. Subsequently, we define $\mathsf{prec}(S)$ as $|\mathcal{S}|$-dimensional subspace $\{\beta \in \mathbb{F}_2^n \colon \beta_i = 0 \text{ if } i \notin \mathsf{supp}(\mathcal{S})\}$, that is, the

set of coordinate positions in which $\mathcal{S}$ is nonzero. The FHT of any function $\mathsf{f}\colon \mathbb{F}_2^n \to \mathbb{R}$ is defined as follows:

$$\widehat{\mathsf{f}}\colon \mathbb{F}_2^n \to \mathbb{R}, \beta \mapsto \sum_{y \in \mathbb{F}_2^n} (-1)^{\langle \beta, y \rangle} \mathsf{f}(y).$$

For a circuit $\mathsf{C}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{n_o}$ and for $y \in \mathbb{F}_2^{n_o}$, we denote the preimages of y under C_x as $(\mathsf{C})_x^{-1}(y)$ which is the set $\{r \in \mathbb{F}_2^{(s-1)n_i+n_r}\colon \mathsf{C}_x(r) = y\}$. Let $\mathsf{D}_{\mathsf{H},x}\colon \mathbb{F}_2^{|\mathcal{W}|} \to \mathbb{R}$ be the preimage-difference function for inputs restricted to x and inputs restricted to 0:

$$\mathsf{D}_{\mathsf{H},x}(y) := |(\mathsf{H})_x^{-1}(y)| - |(\mathsf{H})_0^{-1}(y)|.$$

We now recall the link between the FHT over aforementioned frequency difference, and the leakage characteristic [4].

Theorem 1. *Let* $\mathsf{H}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{|\mathcal{W}|}$ *be the encoded access to a masked circuit* $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ *that implements a function* $\mathsf{F}_{\mathsf{C}'}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{n_o \cdot s}$*. Then,* $\mathsf{L}_{\mathsf{Ex}}(\mathcal{Q}) = 0$ *iff, for all* $x \in \mathbb{F}_2^{n_i}$*, the following holds:*

$$\forall \beta \in \mathsf{prec}(\mathsf{Ex}(\mathcal{Q}))\colon \widehat{\mathsf{D}_{\mathsf{H},x}}(\beta) = 0.$$

3 Methodology

In this section, we introduce our methods for assessing leakage probability bounds in the (g,t,c)-*robust random* (p,ε)-*probing model*, which provide novel tools that enable efficient verification of masked circuits. Further, we establish a link between the leakage characteristic notion and the SIS notion, introduced by Belaïd et *al.* in [8], demonstrating that our robust probing analysis methods can be applied to assess composability notions of masked gadgets in both the (robust) threshold d-probing and (robust) random (p,ε)-probing models.

3.1 Probe-Extended Leakage Characteristic

Theorem 1 links the leakage characteristic to the FHT over preimage differences, yet this link is confined to the unextended setting. Computing the FHT in the presence of a non-bijective probe extension function Ex remains, to date, an open problem and is not generally tractable with existing methods, as missing bijectivity prevents a direct application of standard transform-based techniques. Closing this gap is the core contribution of this work. Instead of transforming under Ex, we construct the extended leakage characteristic recursively from the leakage characteristic defined over the circuit inputs. The result is a consolidated treatment of non-bijective probe extensions that captures the glitch extension, the transition extension, and their combination within a single recursion, and that maps directly onto efficient BDD operations (Section 4). This consolidation, rather than the treatment of any individual extension, is the focus of our work.

The only structural property that this recursion requires is that the extension distributes over unions of probes, which we establish for the probe extensions of Definition 7 and Definition 8 first.

Lemma 1 (Probe Extension Additivity)**.** *A probe extension* $\mathsf{Ex} \subseteq \{\mathsf{GEx}^q, \mathsf{TEx}^q, \mathsf{CEx}^q\}$ *is additive iff for any probesets* $\mathcal{Q}, \mathcal{Q}' \subseteq \mathcal{W}$ *the following holds:*

$$\mathsf{Ex}(\mathcal{Q} \cup \mathcal{Q}') = \mathsf{Ex}(\mathcal{Q}) \cup \mathsf{Ex}(\mathcal{Q}').$$

In particular, this also means $\mathsf{Ex}(\mathcal{Q}) = \bigcup_{q \in \mathcal{Q}} \mathsf{Ex}(\{q\})$.

Proof (Lemma 1). Definition 7, Definition 8 and Definition 9 all have the form

$$\mathsf{Ex}(\mathcal{Q}) = \{w \in \mathcal{W}, \exists q \in \mathcal{Q}\colon \Phi(w, q) = 1\} \cup \mathcal{Q}$$

for a predicate $\Phi\colon \mathcal{W}^2 \mapsto \mathbb{F}_2$. Writing

$$\mathsf{Ex}(\{q\}) = \{q\} \cup \{w \in \mathcal{W}\colon \Phi(w, q) = 1\}$$

for the single probe extension, the definition is a union of these over the probes of $\mathcal{Q}$:

$$\mathsf{Ex}(\mathcal{Q}) = \bigcup_{q\in\mathcal{Q}} \{q\} \cup \bigcup_{q\in\mathcal{Q}} \{w \in \mathcal{W}\colon \Phi(w, q) = 1\} = \bigcup_{q\in\mathcal{Q}} \mathsf{Ex}(\{q\}).$$

Hence, the following completes the proof:

$$\mathsf{Ex}(\mathcal{Q} \cup \mathcal{Q}') = \bigcup_{q\in\mathcal{Q}\cup\mathcal{Q}'} \mathsf{Ex}(\{q\}) = \bigcup_{q\in\mathcal{Q}} \mathsf{Ex}(\{q\}) \cup \bigcup_{q\in\mathcal{Q}'} \mathsf{Ex}(\{q\}) = \mathsf{Ex}(\mathcal{Q}) \cup \mathsf{Ex}(\mathcal{Q}').$$

□ (Lemma 1)

Since both extensions are defined *per probe*, Lemma 1 allows us to state a single theorem for a generic extension $\mathsf{Ex} \subseteq \{\mathsf{GEx}^q, \mathsf{TEx}^q\}$ and to give explicit proof details only where the two extensions genuinely differ, namely in the termination argument. Note that we exclude the coupling extension here although Lemma 1 covers it as well: additivity follows from the shape of the definitions alone and holds for any predicate Φ. What the recursion additionally requires is that the expansion of a probe terminates, which holds for the glitch extension, as it terminates at the stage inputs, and for the transition extension, as it adds a single register instance per probed register, but not for an unconstrained coupling predicate, as we discuss in Section 3.2.

Notation. A probe $q \in \mathcal{W}$ always denotes a *wire*. Whenever a probe symbol q_i appears in a Boolean position (as in the recursion and expansion below), it denotes the associated *coordinate* $\beta_{q_i} \in \{0, 1\}$ of the Walsh–Hadamard index $\beta \in \mathbb{F}_2^{|\mathcal{W}|}$, i. e., the Shannon variable on which the underlying disjunction is cofactored.

To establish the probe-extended leakage characteristic, we first define a partially extended leakage characteristic as helper construct. Its recursion must distinguish between committed and flexible probes, which is what the helper provides.

Definition 11 (Partially Extended Leakage Characteristic)**.** For $\mathcal{S}, \mathcal{R} \subseteq \mathcal{W}$ where $\mathcal{S}$ is the set of committed probes (inflexible) and $\mathcal{R}$ the set of flexible probes ($\mathcal{Q}\backslash\mathcal{S}$), the following holds:

$$\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) := \bigvee_{\beta\in\mathsf{prec}(\mathcal{S}\cup\mathsf{Ex}(\mathcal{R}))} \Phi(\beta)$$

In particular $\mathsf{L}_{\mathsf{Ex}}(\mathcal{Q}) = \mathsf{L}^{\emptyset}_{\mathsf{Ex}}(\mathcal{Q})$ and $\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\emptyset) = \bigvee_{\beta\in\mathsf{prec}(\mathcal{S})} \mathsf{L}_{id}(\mathcal{S})$.

This definition allows us to split the leakage characteristic in evaluated probes $\mathcal{S}$ from the probeset $\mathcal{Q}$ and remaining probes $\mathcal{R}$, which closely reflects how the leakage characteristic is evaluated in practice (see Section 4). We now define the probe-extended leakage characteristic.

Theorem 2 (Probe-Extended Leakage)**.** *Let* $\mathsf{Ex} \subseteq \{\mathsf{GEx}^q, \mathsf{TEx}^q\}$ *and* $\mathcal{Q} \subseteq \mathcal{W}$*. Then* $\mathsf{L}_{\mathsf{Ex}}(\mathcal{Q}) = \mathsf{L}^{\emptyset}_{\mathsf{Ex}}(\mathcal{Q})$ *satisfies, for every committed probeset* $\mathcal{S} \subseteq \mathcal{Q}$*, every flexible probeset* $\mathcal{R} \subseteq \mathcal{W}$ *and any* $q_i \in \mathcal{R}$*:*

$$\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) = \begin{cases} \neg q_i \wedge \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\}) \vee q_i \wedge \mathsf{L}^{\mathcal{S}\cup\mathsf{Ex}(\{q_i\})}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\}) & \text{if } |\mathcal{R}| \neq 0, \\ \mathsf{L}_{id}(\mathcal{R}) & \text{else.} \end{cases}$$

Intuitively, L_{Ex} takes a probe from the probeset one at a time and adds its probe extension to the set of evaluated probes. Within a register stage all probes are extended to their stage inputs (register outputs resp. immediate register predecessors), which form the base case. Any probe not yet contained in $\mathsf{Ex}(\mathcal{Q})$ is expanded towards a probe in $\mathsf{Ex}(\mathcal{Q})$, so that each extended probeset is obtained by a finite recursive expansion over the input probes, which can be implemented efficiently in practice. To provide a proof for Theorem 2, we first need to show that the decompositional property holds (Theorem 3).

Theorem 3 (Leakage Characteristic Decomposition)**.** *Let* $\mathsf{Ex} \subseteq \{\mathsf{GEx}^q, \mathsf{TEx}^q\}$, $\mathcal{S}, \mathcal{R} \subseteq \mathcal{W}$ *and let* $q_i \in \mathcal{R}$*. Then the following holds:*

$$\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) = \neg q_i \wedge \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\}) \vee q_i \wedge \mathsf{L}^{\mathcal{S}\cup\mathsf{Ex}(\{q_i\})}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\})$$

Proof (Theorem 3). For an arbitrary fixed probe $q_i \in \mathcal{R}$ we have:

$$\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) = \bigvee_{\beta\in\mathsf{prec}(\mathcal{S}\cup\mathsf{Ex}(\mathcal{R}))}$$

Because Ex is additive, we can write:

$$\Phi(\beta) = \bigvee_{\beta\in\mathsf{prec}(\mathcal{S}\cup\bigcup_{w\in\mathcal{R}}\mathsf{Ex}(\{w\}))} \Phi(\beta).$$

The probe q_i contributes to exactly the one extension $\mathsf{Ex}(\{q_i\})$. Thus, we have the following two cases:

$$q_i = 0\colon \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) = \bigvee_{\beta\in\mathsf{prec}(\mathcal{S}\cup\bigcup_{w\in\mathcal{R}\backslash\{q_i\}}\mathsf{Ex}(\{w\}))} \Phi(\beta)$$

$$q_i = 1\colon \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) = \bigvee_{\beta\in\mathsf{prec}((\mathcal{S}\cup\mathsf{Ex}(\{q_i\}))\cup\bigcup_{w\in\mathcal{R}\backslash\{q_i\}}\mathsf{Ex}(\{w\}))} \Phi(\beta)$$

Each right-hand side of the sets we are accumulating over are on the remaining set $\mathcal{R}\backslash\{q_i\}$. Hence, the leakage characteristics for each cofactor are:

$$\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R})|_{q_i=0} = \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\}), \quad \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R})|_{q_i=1} = \mathsf{L}^{\mathcal{S}\cup\mathsf{Ex}(\{q_i\})}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\})$$

Now using Boolean expansion, this yields the following and concludes the proof:

$$\begin{aligned}\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}) &= \neg q_i \wedge \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R})|_{q_i=0} \vee q_i \wedge \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R})|_{q_i=1} \\ &= \neg q_i \wedge \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\}) \vee q_i \wedge \mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\mathcal{R}\backslash\{q_i\})\end{aligned}$$

□ (Theorem 3)

Now that we have proven the first part of the recursive probe-extended leakage (Theorem 2) by showing correctness of Theorem 3, what remains to be shown is that the base case holds true and that the recursion terminates.

Proof (Theorem 2). For $\mathcal{R} = \emptyset$, we have $\mathsf{L}^{\mathcal{S}}_{\mathsf{Ex}}(\emptyset) = \bigvee_{\beta\in\mathsf{prec}(\mathcal{S})} \Phi(\beta) = \mathsf{L}^{\mathcal{S}}_{id}$, so the base case is correct. $\mathcal{R} \neq \emptyset$ is Theorem 3 and, given the correctness of our proof, holds true. The recursion terminates, as $|\mathcal{R}|$ strictly decreases because $|\mathcal{R}\backslash\{q_i\}| \leq |\mathcal{R}|$, thus reaching the base case $\mathcal{R} = \emptyset$. Only the probeset influences termination, so this argument is identical for GEx^q, TEx^q and any other additive probe extension Ex. □ (Theorem 2)

3.2 Coupling-Extended Leakage Characteristic

Providing an expansion that fits our recursive split scheme for the coupling extension is not generally possible. Notably, the obstruction is not additivity, which Lemma 1 establishes for the coupling extension as well, but the termination of the recursion. From a theoretical perspective, the reason for this is that we cannot extend a probeset to successing probes, as we cannot use a divide and conquer approach for those not yet reached probes within the FHT. Further, the recursion cannot be guaranteed to terminate, if no constraints are set to the coupling predicate Φ. An intuitive example would be two probes within a probeset that are coupling-extended to each other, which would result in endless recursion. Also, our circuit abstraction does not encode localization or routing information, which makes determining gates/wires in close proximity to target probes impossible. All of the above renders providing a coupling-extended leakage characteristic definition out of scope for this work.

3.3 Compositional Leakage

In addition to expressing leakage in the robust probing models, the adapted *compositional leakage characteristic* is a viable notion to express composability of circuits [4]. Yet, no methods to utilize this in practice have been proposed. Thus, we first recall the definition of compositional leakage, again emphasizing the link to the FHT, therefore establishing a viable method to assess composability of masked circuits in practice.

Definition 12 (Compositional Leakage (adapted from [4]))**.** Let $\mathcal{I} \subseteq \{0, 1, ..., (s-1) \cdot n_i\}$ be an index set and $\mathsf{H}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{|\mathcal{W}|}$ be the encoded access to a masked circuit $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ computing the corresponding function $\mathsf{F}_{\mathsf{C}'}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{n_o \cdot s}$. The compositional leakage of the probeset $\mathcal{Q} \subseteq \mathcal{W}$ for uniform random $r \in \mathbb{F}_2^{n_r}$ is defined as:

$$\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}) = \begin{cases} 0 & \text{if } \forall a \in \mathsf{prec}(\mathcal{I}), b \in \mathsf{prec}(\overline{\mathcal{I}}), y \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{Q})|}\colon \\ & \mathsf{Pr}[\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})}(a \oplus b, r) = y] = \mathsf{Pr}[\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})}(a, r) = y] \\ 1 & \text{else.} \end{cases}$$

Intuitively, compositional leakage is an indistinguishability test under toggling an input offset b supported on $\mathcal{I}$. More precisely, observed probe values are indifferent to toggles on inputs in $\overline{\mathcal{I}}$.

3.3.1 Link to FHT.

Now that we have defined compositional leakage, we can again show a link to the FHT, which again provides practical (i. e., efficient) computation methods to assess composability of masked circuits.

Theorem 4. *Let $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ be a masked circuit implementing a function $\mathsf{F}_{\mathsf{C}'}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{n_o \cdot s}$ with encoded access $\mathsf{H}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{|\mathcal{W}|}$. Further, let $\mathsf{Ex}\colon \mathbb{F}_2^{|\mathcal{W}|} \to \mathbb{F}_2^{|\mathcal{W}|}$ be a probe extension function. Let $\mathcal{I} \subseteq \{0, 1, ..., n_i \cdot s - 1\}$ be a fixed index set and $\mathcal{Q} \subseteq \mathcal{W}$ be a probeset. For $a, b \in \mathsf{prec}(\mathcal{I}) \subseteq \mathbb{F}_2^{n_i \cdot s}$, we define the preimage-difference function $\mathsf{D}_{\mathsf{H},(a,b)}(y)\colon \mathbb{F}_2^{|\mathcal{W}|} \to \mathbb{R}$ as follows:*

$$\mathsf{D}_{\mathsf{H},(a,b)}(y) := |(\mathsf{H})^{-1}_{a \oplus b}(y)| - |(\mathsf{H})^{-1}_{a}(y)|.$$

Consequently, the compositional leakage $\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q})$ for a probeset $\mathcal{Q} \subseteq \mathcal{W}$ is 0 if and only if the following holds:

$$\forall a \in \mathsf{prec}(\mathcal{I}), b \in \mathsf{prec}(\overline{\mathcal{I}}), \forall \beta \in \mathsf{prec}(\mathcal{Q})\colon \widehat{\mathsf{D}_{\mathsf{H},(a,b)}}(\beta) = 0.$$

Proof (Theorem 4). First, we show that the compositional leakage (through indistinguishability of probabilities) can be expressed as equality of the preimage frequencies on the projected view. By definition, $\mathsf{L}_{(\mathcal{I},\mathsf{Ex})}(\mathcal{Q}) = 0$ if and only if for all $a \in \mathsf{prec}(\mathcal{I}), b \in \mathsf{prec}(\overline{\mathcal{I}}), y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{Q})|}$ the following holds:

$$\Pr_{r\in\mathbb{F}_2^{n_r}}[\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})}(a\oplus b, r) = y'] = \Pr_{r\in\mathbb{F}_2^{n_r}}[\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})}(a, r) = y']. \tag{1}$$

Since r is uniform in $\mathbb{F}_2^{n_r}$, this is equivalent to

$$|(\mathsf{H})^{-1}_{a\oplus b}(y')| = |(\mathsf{H})^{-1}_{a}(y')|, \forall a \in \mathsf{prec}(\mathcal{I}), \forall b \in \mathsf{prec}(\overline{\mathcal{I}}), \forall y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{Q})|}. \tag{2}$$

For $y \in \mathbb{F}_2^{|\mathcal{W}|}$, we denote $\pi_{\mathsf{Ex}(\mathcal{Q})}(y)$ as the projection of y to the coordinates indexed by $\mathsf{Ex}(\mathcal{Q})$. Then $\pi_{\mathsf{Ex}}(y) = y'$ for $y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{Q})|}$. We define the projected difference for a fixed probeset $\mathcal{Q} \subseteq \mathcal{W}$ as

$$\mathsf{D}^{\mathsf{Ex}(\mathcal{Q})}_{\mathsf{H},(a,b)} := \sum_{y\in\mathbb{F}_2^{|\mathcal{W}|}, \pi_{\mathsf{Ex}(\mathcal{Q})}(y)=y'} \mathsf{D}_{\mathsf{H},(a,b)}(y). \tag{3}$$

We can now show that $\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}) = 0$ iff $\mathsf{D}^{\mathsf{Ex}(\mathcal{Q})}_{\mathsf{H},(a,b)} = 0$ for all $a \in \mathsf{prec}(\mathcal{I}), b \in \mathsf{prec}(\overline{\mathcal{I}}), y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{Q})|}$:

$$\begin{aligned}
\mathsf{D}^{\mathsf{Ex}(\mathcal{Q})}_{\mathsf{H},(a,b)}(y') &= \sum_{y\in\mathbb{F}_2^{|\mathcal{W}|}, \pi_{\mathsf{Ex}(\mathcal{Q})}(y)=y'} (|(\mathsf{H})^{-1}_{a\oplus b}(y)| - |(\mathsf{H})^{-1}_{a}(y)|) \\
&= \sum_{y\in\mathbb{F}_2^{|\mathcal{W}|}, \pi_{\mathsf{Ex}(\mathcal{Q})}(y)=y'} |(\mathsf{H})^{-1}_{a\oplus b}(y)| - \sum_{y\in\mathbb{F}_2^{|\mathcal{W}|}, \pi_{\mathsf{Ex}(\mathcal{Q})}(y)=y'} |(\mathsf{H})^{-1}_{a}(y)| \\
&= \{r \in \mathbb{F}_2^{n_r} : \mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})}(a\oplus b, r) = y'\} - \{r \in \mathbb{F}_2^{n_r} : \mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})}(a, r) = y'\} \\
&= |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})})^{-1}_{a\oplus b}(y')| - |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})})^{-1}_{a}(y')|.
\end{aligned} \tag{4}$$

Now that we have shown the equivalence of the compositional leakage to the difference of preimage counts on the projected view, it remains to show that all coefficients of the FHT over $\mathsf{D}_{\mathsf{H},(a,b)}$ are 0 for all $a \in \mathsf{prec}(\mathcal{I}), b \in \mathsf{prec}(\overline{\mathcal{I}}), \beta \in \mathsf{prec}(\mathcal{Q})$ if and only if $|(\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})})^{-1}_{a\oplus b}(y')| - |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{Q})})^{-1}_{a}(y')| = 0$. For that, we fix $a \in \mathsf{prec}(\mathcal{I})$ and $b \in \mathsf{prec}(\overline{\mathcal{I}})$. For any $\beta \in \mathsf{prec}(\mathsf{Ex}(\mathcal{Q}))$ it holds that

$$\begin{aligned}
\widehat{\mathsf{D}_{\mathsf{H},(a,b)}}(\beta) &= \sum_{y\in\mathbb{F}_2^{|\mathcal{W}|}} (-1)^{\langle\beta,y\rangle} \mathsf{D}_{\mathsf{H},(a,b)}(y) \\
&= \sum_{y'\in\mathbb{F}_2^{|\mathsf{Ex}(\mathcal{Q})|}} (-1)^{\langle\pi_{\mathsf{Ex}(\mathcal{Q})}(\beta),y'\rangle} \mathsf{D}^{\mathsf{Ex}(\mathcal{Q})}_{\mathsf{H},(a,b)}(y').
\end{aligned} \tag{5}$$

□ (Theorem 4)

3.3.2 Link to SIS.

ironMask introduced the SIS function to formalize probing security and composability properties of masked circuits [8]. For a given probe set, the SIS function specifies the minimal tuple of input-share indices required to simulate the joint distribution of the observed wires. The SIS definition can be related directly to an indistinguishability-based leakage formulation. In particular, a simulator exists with respect to an input index set $\mathcal{I}$

if and only if the corresponding compositional leakage characteristic evaluates to zero when conditioning on the shares indexed by $\mathcal{I}$. This equivalence enables composability notions expressed via SIS to be reformulated in terms of leakage characteristics. Table 2 summarizes the resulting leakage-based composability notions. First, we recall the definition of the SIS function [8].

Definition 13 (Set of Input Shares (adapted from [8]))**.** Let $\mathsf{G} = (\mathcal{G}, \mathcal{W})$ be an (s-share, n_i-to-1) gadget implementing a function $\mathsf{F_G}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^s$ that maps an n_i input sharings $(x_1, ..., x_{n_i}) \in \mathbb{F}_2^{s \cdot n_i}$ to an output sharing $y \in \mathbb{F}_2^s$. Let $\mathcal{Q} \subseteq \mathcal{W}$ be a probeset on G and $\mathcal{O} \subseteq \{1, ..., n_o\}$ a set of output indices. The function $\mathsf{SIS_G}$ maps $\mathcal{Q}$ and $\mathcal{O}$ to the unique smallest sets of input indices $\mathcal{I}_1, ..., \mathcal{I}_{n_i}$, such that $(\mathcal{Q}, y|_{\mathcal{O}})$ can be perfectly simulated from $x_1|_{\mathcal{I}_1}, ..., x_s|_{\mathcal{I}_s}$.

Theorem 5. *Let $\mathsf{C}' = (\mathcal{G}, \mathcal{W})$ be a masked circuit implementing a function $\mathsf{F_{C'}}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{n_o \cdot s}$. Further, let $\mathcal{Q} \subseteq \mathcal{W}$ be a probeset, let $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_{n_i})$ be a tuple of input indices for each shared input and $\mathcal{O} \subseteq \{0, 1, ..., s-1\}$ be an output share index set. The SIS-function over a gadget G is related to the compositional leakage characteristic as follows:*

$$\mathsf{SIS}(\mathcal{Q}, \mathcal{O}) = \arg \min_{(\mathcal{I}_1, ..., \mathcal{I}_{n_i})} \{\mathcal{I} = (\mathcal{I}_1, ..., \mathcal{I}_{n_i})\colon \mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0\}.$$

Intuitively, $\mathsf{L}_{\mathcal{I}}$ can distinguish whether the simulation succeeds with the provided input index tuple $\mathcal{I}$ and SIS computes a minimal simulation-enabling input index tuple. Thus, $\mathsf{L}_{\mathcal{I}}$ is a witness that the SIS computation is correct.

Proof (Theorem 5). Suppose $\mathcal{I} = (\mathcal{I}_1, ..., \mathcal{I}_{n_i}) = \mathsf{SIS_G}(\mathcal{Q}, \mathcal{O})$. For simplicity, we henceforth denote $\{\mathcal{I} = (\mathcal{I}_1, ..., \mathcal{I}_{n_i})\colon \mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0\}$ as $\mathcal{K}$.

1. **Feasibility:** By definition of SIS, $(\mathcal{Q}, y|_{\mathcal{O}})$ is perfectly simulatable from the index set $(x_1|_{\mathcal{I}_1}, ..., x_{n_i}|_{\mathcal{I}_{n_i}})$. By definition of compositional leakage, this means that $\mathsf{L}_{\mathcal{I}}(\mathcal{Q}, \mathcal{O}) = 0$. Hence, $\mathcal{I}$ is in $\mathcal{K}$.
2. **Minimality:** Let $\mathcal{J} = (\mathcal{J}_1, ..., \mathcal{J}_{n_i}) \in \mathcal{K}$. Then $\mathsf{L}_{\mathcal{J}}(\mathcal{Q}, \mathcal{O}) = 0$. Since SIS is defined to output the unique smallest tuple $\mathcal{I} = (\mathcal{I}_1, ..., \mathcal{I}_{n_i})$, the following holds:

$$\forall i \in 1, ..., n_i : \mathcal{I}_i \subseteq \mathcal{J}_i.$$

□ (Theorem 5)

Assessing composability with this definition is advantageous compared to INDIANA [4], as we now have a concrete computation method using the FHT. Furthermore, we can now use this link to define threshold composability notions via compositional leakage (Table 2).

Table 2: Threshold composability notions with d_1 probes on input- and intermediate gates, d_2 probes on the output gates and $d = d_1 + d_2$. Notably, the condition $\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0$ always has to be fulfilled for each notion to hold.

Notion	**Condition**
d-NI [2]	$\forall(\mathcal{Q}, \mathcal{O}) \subseteq \mathbb{F}_2^{\lvert\mathcal{W}\rvert}, \exists\, \mathcal{I}\colon (\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0) \wedge (\forall i, \lvert\mathcal{I}_i\rvert \leq d)$
d-SNI [3]	$\forall(\mathcal{Q}, \mathcal{O}) \subseteq \mathbb{F}_2^{\lvert\mathcal{W}\rvert}, \exists\, \mathcal{I}\colon (\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0) \wedge (\forall i, \lvert\mathcal{I}_i\rvert \leq d_1)$
d-TNI [5]	$\forall(\mathcal{Q}, \mathcal{O}) \subseteq \mathbb{F}_2^{\lvert\mathcal{W}\rvert}, \exists\, \mathcal{I}\colon (\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0) \wedge (\forall i, \lvert\mathcal{I}_i\rvert \leq d_1 + d_2)$
d-P-INI [14]	$\forall(\mathcal{Q}, \mathcal{O}) \subseteq \mathbb{F}_2^{\lvert\mathcal{W}\rvert}, \exists\, \mathcal{I}\colon (\mathsf{L}_{\mathcal{I},\mathsf{Ex}}(\mathcal{Q}, \mathcal{O}) = 0) \wedge (\forall i, \lvert(\bigcup_{i=1}^{n_i} \mathcal{I}_i) \backslash \mathcal{O}\rvert \leq d_1)$

Having established this link to argue the definition and correctness of evaluating composability with the compositional leakage characteristic, the question arises whether computing

Algorithm 1: Algorithm computing SIS using the compositional leakage characteristic.

Require: Masked circuit C', probe set $\mathcal{Q}$, output index set $\mathcal{O}$.
Ensure: $\mathcal{I} = (\mathcal{I}_1, ..., \mathcal{I}_{n_i})$

1 **for** i *in* 1 *to* n_i **do**
2 $\quad \mathcal{I}_i \leftarrow \emptyset$
3 $\mathcal{J} \leftarrow \{(i,j) \colon i \in \{1,...,n_i\}, j \in \{0,...,s-1\}\}$ ▷ TRACK SHARES NOT IN $\mathcal{I}$
4 **while** $\mathsf{L}_{\mathcal{I}}(\mathcal{Q}, \mathcal{O}) = 1$ **do**
5 $\quad$ pick $(i,j) \in \mathcal{J}$ such that $\forall \beta \in \mathsf{prec}(\mathcal{Q})$,
6 $\quad\quad a_1 \in \mathsf{prec}(\mathcal{I} \cup \mathcal{J}), a_2 \in \mathsf{prec}(\mathcal{I}), b_1 \in \mathsf{prec}(\overline{\mathcal{I} \cup \mathcal{J}}), b_2 \in \mathsf{prec}(\overline{\mathcal{I}})$:
7 $\quad\quad\quad \mathsf{D}_{\mathsf{H},(a_1,b_1)}(\beta) \leq \mathsf{D}_{\mathsf{H},(a_2,b_2)}(\beta)$
8 $\quad \mathcal{I}_i \leftarrow \mathcal{I}_i \cup \{j\}\ \mathcal{J} \leftarrow \mathcal{J} \backslash \{(i,j)\}$
9 **for** $(i,j) \in \{(i,j) \colon j \in \mathcal{I}_i\}$ **do**
10 $\quad \mathcal{I}'_i \leftarrow \mathcal{I}'_i \backslash \{j\}\ \mathcal{I}' \leftarrow (\mathcal{I} \backslash \{\mathcal{I}_i\}) \cup \mathcal{I}'_i$ ▷ REMOVE SHARE j FOR INPUT i FROM $\mathcal{I}$
11 $\quad$ **if** $\mathsf{L}_{\mathcal{I}'}(\mathcal{Q}, \mathcal{O}) = 0$ **then**
12 $\quad\quad \mathcal{I} \leftarrow \mathcal{I}'$
13 **return** $\mathcal{I}$

the leakage-based definitions is viable in practice. For that, we describe a naive greedy algorithm that computes SIS using the compositional leakage characteristic (Algorithm 1).

The algorithm can be divided into three distinct phases, the first being initialization (ll. 1–3), the second being the core greedy algorithm where the index share sets are computed (ll. 4–6), and third the share index pruning phase where I is minimized (ll. 7–10). We start off with an empty index set that we continuously fill with share indices, until the compositional leakage characteristic over the input share sets is zero. This ensures that we retrieve a share index set tuple that fulfills the criteria for perfect simulation. The index pruning phase then makes sure that the share index set tuple is minimal, i. e., that no share indices can be removed from any input set.
The advantage of this approach, even though we might be less efficient than ironMask, is that we can use this on arbitrarily structured circuits, and are thus more flexible than existing tools.

3.4 Random Composability

With ironMask, Belaïd et *al.* [8] provided random composability notions in terms of SIS, which means with the help of our link between SIS and compositional leakage characteristic, we are able to express these notions in terms of the compositional leakage characteristic. Again, this shows the flexibility of our approach, as the leakage characteristic can be evaluated for arbitrarily structured circuits.

To begin, we first recall the SIS-based RPC definition (Definition 14).

Definition 14 ((t,p,ε)-Random Probing Composability [8])**.** Let $\mathsf{G}' = (\mathcal{G}, \mathcal{W})$ be a masked gadget that implements a gadget function $\mathsf{F}_{\mathsf{G}'} \colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^s$. Further, let $p, \varepsilon \in [0,1]$ and $\mathcal{Q}$ be a probeset that is queried from the LeakageSampler function, which takes the masked gadget G' and the leakage rate p as inputs and probabilistically samples said probeset $\mathcal{Q} \subseteq \mathcal{W}$. G' is (t,p,ε)-random probing composable, if for every $\mathcal{O} \subseteq \{1,...,s\}$ with $|\mathcal{O}| = t$, the input share sets $(\mathcal{I}_1, ...\mathcal{I}_{n_i}) = \mathsf{SIS}(\mathcal{Q}, \mathcal{O})$ satisfy $\mathsf{Pr}[(|\mathcal{I}_1| >$

$t) \vee ... \vee (|\mathcal{I}_{n_i}| > t)] \leq \varepsilon$ where the probability is taken over all probesets obtained by $\mathsf{LeakageSampler}(\mathsf{G}', p)$.

The failure event $(|\mathcal{I}_1| > t) \vee ... \vee (|\mathcal{I}_{n_i}| > t)$ is defined as the event that any of the input index sets for a single input contains more than t shares. From this, we can now construct a more compact definition based on the compositional leakage characteristic, which still yields the advantage that it can be computed on arbitrarily structured circuits.

Theorem 6. *Let $\mathsf{G}' = (\mathcal{G}, \mathcal{W})$ be a masked gadget that implements a function $\mathsf{F}_{\mathsf{G}'} \colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^s$. Let $\tilde{\mathcal{I}} := \arg\min_{\mathcal{I}}\{\mathcal{I} \colon \mathsf{L}_{\mathcal{I}}(\mathcal{Q}, \mathcal{O}) = 0\}$ for a probeset $\mathcal{Q}$ and an output index set $\mathcal{O} \subseteq \{1, ..., s\}$ with $|\mathcal{O}| = t$. G' is (t, p, ε)-random probing composable if for all $\mathcal{O} \subseteq \{1, ..., s\}$ with $|\mathcal{O}| = t$, the following holds:*

$$\Pr_{\mathcal{Q} \leftarrow \mathsf{LeakageSampler}}[\bigvee_{j=1}^{n_i} (|\tilde{\mathcal{I}}(\mathcal{Q}, \mathcal{O})| > t)] \leq \varepsilon.$$

Here, a failure event occurs if more than t shares of an input are necessary to simulate $(\mathcal{Q}, \mathcal{O})$, which is equivalent to the aforementioned failure event definition, with the advantage of being expressable via the compositional leakage characteristic. Thus, correctness is trivial, as SIS and $\tilde{\mathcal{I}}$ are equivalent (Theorem 5). Consequently, we can also express (t, p, ε)-Random Probing Expandability (RPE) notions, which logically follow the adapted definition of RPC in Table 3. Notably, the link extends to finer-grained composability notions. Cardinal-Random Probing Composability (CRPC) [9], for instance, replaces the threshold failure event by a probability envelope over the cardinalities $(|\mathcal{I}_1|, \ldots, |\mathcal{I}_{n_i}|)$ of the required input share sets, indexed by the cardinalities of the simulated output shares. As $\tilde{\mathcal{I}}(\mathcal{Q}, \mathcal{O})$ already yields these sets for every sampled probeset, expressing CRPC amounts to accumulating probesets per cardinality tuple rather than testing a single threshold. We do not pursue this here, as our focus is the treatment of robust probe extensions rather than the composition framework itself.

Table 3: Random probing composability/expandability notions defined over compositional leakage characteristics.

Notion	Condition
(t, p, ε)-RPE1	RPC and $\forall \mathcal{O} \subseteq \{1, ..., s\}$, $\lvert\mathcal{O}\rvert = t$: $\Pr_{\mathcal{Q}}[\bigwedge_{j=1}^{n_i} (\lvert\tilde{\mathcal{I}}(\mathcal{Q}, \mathcal{O})\rvert > t)] \leq \varepsilon^2$
(t, p, ε)-RPE2	RPC and $\exists \mathcal{O} \subseteq \{1, ..., s\}$, $\lvert\mathcal{O}\rvert = s - 1$: $\Pr_{\mathcal{Q}}[\bigwedge_{j=1}^{n_i} (\lvert\tilde{\mathcal{I}}(\mathcal{Q}, \mathcal{O})\rvert > t)] \leq \varepsilon^2$

4 Automated Reasoning

To reason about the performance and efficiency of our new methods, we implemented a proof-of-concept verification tool called CHARRM[2] that performs leakage assessment in the $(g, t, 0)$-robust random probing model. The focus lies on demonstrating that our methods are not merely theoretical, but practically feasible and competitive with state-of-the-art verification tooling. We describe the implementation top-down, from a high level overview to the algorithms that perform the leakage assessment.

[2]The tool and the corresponding case studies will be released as open source upon acceptance of this paper.

4.1 Implementation

4.1.1 Assumptions and Constraints.

We first state the assumptions our implementation places on its input. We assume non-iterative target circuits, i. e., circuits without cycles, which is reasonable as gadgets are our primary target and larger iterative circuits may be analyzed after unrolling. For the transition extension, we accordingly assume fully unrolled and preprocessed circuits, as compatible unrolling requires extensive engineering effort and is out of scope for this work. We further expect fully pipelined circuits and independent and identically distributed inputs, i. e., initial sharings and refresh values, which requires input share annotations, as well as register cycle annotations that identify the same register across its execution cycles. These assumptions ease the implementation without limiting the method itself.

4.1.2 Overview.

CHARRM is a C++ tool built around a custom circuit abstraction that closely aligns with our circuit model. Its analysis workflow comprises the following steps:

1. **Circuit Parsing:** The target circuit is parsed into the internal abstraction. As a proof of concept, we opted for a simplified netlist format that is easy to parse, but requires conversion from a gate level netlist.

2. **Circuit Generation:** Explicit gate and wire functions are generated to model the functional behavior and the signal dependencies of the circuit.

3. **Circuit Partitioning:** The circuit is partitioned in depth at register boundaries, which separates the combinational segments between sequential elements, as required for the glitch extension.

4. **Leakage Assessment:** The leakage probability bound is computed for each subcircuit.

5. **Reporting:** The results are formatted into a structured summary of the analysis outcomes.

4.1.3 Partitioning.

By Definition 7, the probe glitch extension adds probes on all inputs of the combinational logic that influence the probed computation, all of which lie within the same register stage. We therefore divide the target circuit before each register stage and analyze the stages separately and in order, which shrinks the complexity of the analysis and yields a leakage probability bound per clock cycle.

For the transition- and glitch+transition extension, the partitioning optimization is disabled. Our circuit model carries no notion of time, hence a circuit that is evaluated over several clock cycles is unrolled into a single acyclic circuit in which every clock cycle is represented by its own copy of the combinational logic and of the registers. A register that is written in two consecutive cycles thus occurs as two distinct instances in two different copies, and a transition-extended probe on that register relates precisely these two instances. Partitioning the circuit at its register boundaries would place the two instances in different partitions and therefore discard the probe pairs that constitute transition leakage. The instances of one register are linked across the copies by an annotation that identifies them as the same physical register.

4.1.4 Analysis.

The analysis routine itself follows the symbolic procedure of INDIANA [4]: The discrete input distribution, represented as MTBDD, is initialized uniformly over all unshared secrets, the preimage difference $\mathsf{D}_{\mathsf{H},x}$ between a fixed secret and all other secrets is computed symbolically, and the FHT over this difference yields the leakage characteristic. From the leakage characteristic of a stage, the leakage probability bound is derived with the modified satisfiability count algorithm of [4], before the discrete probability distribution $|(\mathsf{H})_x^{-1}(y)|$ is transitioned to the inputs of the next stage, if it exists.

The probe extensions enter this routine at two points. For the $(1,0,0)$-extended assessment, the input distribution is transitioned to the stage inputs, whose leakage characteristic constitutes the base case of Theorem 2, and is then recursively extended to all probes in the probeset $\mathcal{Q}$. For the $(0,1,0)$-extended assessment, the distribution is instead transitioned to $\mathcal{Q}$ extended by the preceding instance q' of every probed register q, which the register annotations of the unrolled circuit identify across clock cycles, and the extension is realized as the simultaneous substitution $\beta_{q'} := \beta_{q'} \vee \beta_q$ on the probe variables, after which every probe variable outside $\mathcal{Q}$ is restricted to unprobed. For the $(1,1,0)$-extended assessment, both extensions are applied to the original probeset rather than nested into one another, which reflects their additivity. Since every transition target is a register and therefore already a leaf of the glitch extension, computing the glitch extension of the transition-extended probeset yields exactly the union of the glitch- and the transition-extended probeset, which is the extended probeset that the $(1,1,0)$-robust model prescribes.

5 Case Studies

To demonstrate the practical applicability and feasibility of our approach, we implemented our verification concept for $(g,t,0)$-robust random probing within an automated security verification framework and tool. We begin with a comparative evaluation against ironMask, the sole existing competitor that supports verification in a robust random probing model, which restricts the comparison to the $(1,0,0)$-robust random probing model. We then analyze verification in the $(0,1,0)$-robust random probing and $(1,1,0)$-robust random probing models, demonstrate the versatility of our approach across arbitrary circuit structures, and complement these case studies with a composability assessment. Couplings remain out of scope for the implementation, as discussed in Section 3.2.

Verification Setup. All experiments were conducted on a high-performance computing system equipped with an Intel Xeon E5-1660v4 CPU with 16 cores, a clock frequency of 3.2 GHz, and 128 GB of RAM.

5.1 $(1,0,0)$-Robust Random Probing Leakage Assessment

In the $(1,0,0)$-robust random probing model, we compare against ironMask on fundamental ISW addition, multiplication, refresh, and copy gadgets [20]. The comparison requires caution, as both tools employ different circuit representations. ironMask describes gadgets in a software-oriented manner, where multiple usages of an intermediate require additional wires and copy gadgets, whereas our tool models circuits as DAGs with wires as 1-to-1 connections and gates of arbitrary fan-out. The resulting difference in the number of probe positions leads to differing leakage characteristics and leakage probability bounds. To ensure comparability, we therefore implemented a compatibility mode that adapts the circuit representation while retaining the formal methods of Section 3, and we reran the ironMask experiments on the same machine. Table 4 reports the resulting metrics.

Table 4: Comparison of verification performance between CHARRM (compatibility mode) and ironMask for selected gadgets (based on the ironMask repository) under $(1, 0, 0)$-robust RPS conditions ($p = 2^{-8}$). In compatibility mode, CHARRM matches ironMask's circuit representation so that wire counts (resp. positions) coincide. All experiments were subject to a timeout limit (TO.) of 1h.

Scheme	Shares	Circuit					ironMask		CHARRM	
	$d+1$	Wires	Comb.	Rand.	Seq.	Lat.	ε	Time	ε	Time
ISW *refresh*	2	5	2	1	–	–	$1.526e^{-5}$	0 ms	$1.526e^{-5}$	11 ms
	3	12	6	3	–	–	$4.740e^{-7}$	1 ms	$4.740e^{-7}$	16 ms
	4	30	12	6	–	–	$1.857e^{-8}$	28 ms	$1.857e^{-8}$	35 ms
	5	50	20	10	–	–	$9.044e^{-10}$	115.2 s	$9.044e^{-10}$	119 ms
ISW *copy*	2	12	4	2	–	–	$1.363e^{-4}$	1 ms	$1.363e^{-4}$	14 ms
	3	33	12	6	–	–	$7.278e^{-6}$	12 ms	$7.278e^{-6}$	35 ms
	4	64	24	12	–	–	$5.334e^{-7}$	14 ms	$5.334e^{-7}$	192 ms
	5	105	40	20	–	–	–	TO.	$4.967e^{-8}$	1.1 s
ISW *add*	2	14	6	2	–	–	$1.216e^{-4}$	1 ms	$1.216e^{-4}$	17 ms
	3	36	15	6	–	–	$3.181e^{-6}$	1.4 s	$3.181e^{-6}$	62 ms
	4	68	28	12	–	–	–	TO.	$1.164e^{-7}$	699 ms
	5	110	45	20	–	–	–	TO.	$5.467e^{-9}$	10.4 s
ISW *multiply*	2	21	8	1	6	2	$4.753e^{-3}$	2 ms	$\{4.753e^{-3}, 6.080e^{-5}, -\}$	41 ms
	3	57	21	3	12	2	$8.740e^{-4}$	24.9 s	$\{8.740e^{-4}, 1.591e^{-6}, -\}$	118 ms
	4	110	40	6	20	2	–	TO.	$\{1.748e^{-4}, 5.822e^{-8}, -\}$	2.1 s
	5	180	65	10	30	2	–	TO.	$\{4.910e^{-5}, 2.733e^{-9}, -\}$	33.6 s

While ironMask performs better for small security orders, the difference remains within milliseconds and is attributable to setup overhead rather than to the verification itself. As the number of shares increases, our approach gains a significant performance advantage across all gadgets and security orders. ironMask scales by constraining the coefficient estimation through the c_{max} parameter, which trades precision for performance, whereas our verification is exhaustive and therefore exact in all experiments. Notably, our tool derives bounds for circuits that ironMask cannot handle even with restricted c_{max}.

A further limitation of ironMask is that it reports a single global leakage probability bound even for non-linear gadgets, despite the well-documented necessity of additional registers to mitigate glitch propagation in ISW multiplication. We therefore reimplemented the case studies with the required pipelining registers, which increases the latency from 0 to 2 cycles, and report a bound for each cycle, resp. register stage. The last pipelining stages exhibit no leakage, as they contain only the output registers, which our model excludes as probe locations. Since both tools insert copy gates to bound the fan-out by 2, these results do not reflect the leakage of a hardware realization, for which we report our native circuit representation in Table 5.

Both representations yield matching leakage probability bounds for *refresh*, *add*, and the second register stage of *multiply* across all considered security orders, and only the first-stage *multiply* bound is slightly lower under the native representation. While the numerical impact is modest, the native representation more accurately reflects the leakage exposed by a hardware realization, in which fan-out is implemented through wire branching rather than explicit copy gadgets.

Table 5: Verification performance of CHARRM on the same gadgets as Table 4, using CHARRM's native circuit representation.

Scheme	Shares	Circuit					Verification	
	$d+1$	Wires	Comb.	Rand.	Seq.	Lat.	ε	Time
ISW	2	6	2	1	–	–	$1.526e^{-5}$	15 ms
refresh	3	15	6	3	–	–	$4.740e^{-7}$	18 ms
	4	28	12	6	–	–	$1.857e^{-8}$	34 ms
	5	45	20	10	–	–	$9.044e^{-10}$	69 ms
ISW	2	12	4	2	–	–	$1.526e^{-5}$	21 ms
copy	3	30	12	6	–	–	$1.591e^{-6}$	24 ms
	4	56	24	12	–	–	$1.410e^{-7}$	57 ms
	5	90	40	20	–	–	$1.442e^{-8}$	207 ms
ISW	2	14	6	2	–	–	$1.216e^{-4}$	18 ms
add	3	33	15	6	–	–	$3.181e^{-6}$	44 ms
	4	60	28	12	–	–	$1.164e^{-7}$	260 ms
	5	95	45	20	–	–	$5.467e^{-9}$	2.187 s
ISW	2	24	8	1	6	2	$\{4.221e^{-3}, 6.080e^{-5}, -\}$	48 ms
multiply	3	57	21	3	12	2	$\{4.611e^{-4}, 1.591e^{-6}, -\}$	70 ms
	4	104	40	6	20	2	$\{8.253e^{-5}, 5.822e^{-8}, -\}$	318 ms
	5	165	65	10	30	2	$\{1.767e^{-5}, 2.733e^{-9}, -\}$	3.062 s

5.2 $(g, 1, 0)$-Robust Random Probing Leakage Assessment

Besides glitches, transitions constitute the second physical default commonly encountered in masked hardware, as overwriting a register reveals a function of two consecutively stored values. We therefore verify first-order HPC_2 and HPC_3 AND gadgets placed in an iterative feedback loop, i. e., where the gadget output is fed back to its input in the subsequent clock cycle, which is the canonical construction in which transition leakage manifests. For each gadget, we additionally consider a *hardened* variant that spends additional randomness and registers to counteract transition and glitch leakage. Every construction is verified in the $(g, t, 0)$-robust random probing model for all combinations of glitch and transition extension with $p = 2^{-8}$, and Table 6 reports the results.

In contrast to the preceding experiments, these analyses are conducted without register stage partitioning. Partitioning decomposes a circuit at its register boundaries and is admissible as long as no extended probe crosses such a boundary, which holds for the glitch extension, as its expansion terminates at the stage inputs. The transition extension, however, attaches a register probe to the instance of the same register one clock cycle earlier and thus spans a register boundary by construction, so that partitioning would split the timing domain of a register and omit precisely those probe pairs that constitute transition leakage. We therefore unroll each construction over two clock cycles, such that every register is written twice and contributes exactly one transition pair, and analyze the result as a single subcircuit.

Across all constructions, the leakage probability bound ε increases monotonically from $(0, 0, 0)$, over $(0, 1, 0)$ and $(1, 0, 0)$, to $(1, 1, 0)$. The glitch extension consistently exhibits the larger effect, which is best observable for the plain HPC_3 loop, where the transition extension raises ε from $2.864e^{-3}$ to $3.577e^{-3}$, while the glitch extension yields $6.627e^{-3}$. This follows from the size of the extended probe sets, as a glitch-extended probe observes all stage inputs within the combinational cone of the probed wire, whereas a transition-extended probe contributes a single additional observation. Since both extensions act on the original probe set and compose as a union, their combination yields the largest bound throughout.

The individual constructions differ in their susceptibility to transitions. For the plain HPC_2 loop, the transition extension increases ε by a factor of 2.3 (from $6.655e^{-3}$ to $1.563e^{-2}$), rendering transitions nearly as damaging as glitches, whereas the plain HPC_3 loop grows by a factor of 1.2 only. Accordingly, the hardened variants pay off where the plain construction is susceptible, as they roughly halve the bound of the HPC_2 loop under transition extension

Table 6: Verification results for HPC_2 and HPC_3 AND gadgets placed in an iterative feedback loop under $(g, t, 0)$-robust RPS conditions ($p = 2^{-8}$), evaluated without register stage partitioning. Each netlist is unrolled over two clock cycles, such that every register contributes exactly one transition pair. The *hardened* variants employ additional randomness and registers to counteract transition and glitch leakage.

Scheme	Shares	Circuit				Extension	Verification	
	$d+1$	Wires	Rand.	Comb.	Seq.		ε	Time
HPC_2 *plain*	2	82	2	32	24	$(0,0,0)$	6.655e^{-3}	4.436 s
						$(0,1,0)$	1.563e^{-2}	4.406 s
						$(1,0,0)$	1.633e^{-2}	210 ms
						$(1,1,0)$	2.506e^{-2}	163 ms
HPC_2 *hardened*	2	104	4	34	40	$(0,0,0)$	5.721e^{-3}	2.928 s
						$(0,1,0)$	7.645e^{-3}	2.750 s
						$(1,0,0)$	8.943e^{-3}	1.353 s
						$(1,1,0)$	1.098e^{-2}	900 ms
HPC_3 *plain*	2	80	4	34	16	$(0,0,0)$	2.864e^{-3}	912 ms
						$(0,1,0)$	3.577e^{-3}	905 ms
						$(1,0,0)$	6.627e^{-3}	261 ms
						$(1,1,0)$	7.445e^{-3}	260 ms
HPC_3 *hardened*	2	100	6	38	28	$(0,0,0)$	2.708e^{-3}	10.404 s
						$(0,1,0)$	3.279e^{-3}	7.057 s
						$(1,0,0)$	6.534e^{-3}	1.933 s
						$(1,1,0)$	7.266e^{-3}	1.315 s

(7.645e^{-3}) and under the combined extension (1.098e^{-2}), while the HPC_3 values remain virtually unchanged. They achieve this despite being the larger circuits, i. e., the additional randomness more than compensates for the additional probe positions at a fixed leakage rate p.

With respect to runtime, all constructions are verified within seconds, and the runtime decreases monotonically as the leakage probability bound increases, across all settings and constructions. This inverse ordering follows from the leakage characteristic being a monotone Boolean function. A stronger probe extension only ever adds leaking probesets and thereby pushes the characteristic towards the constant-one function, whose BDD is trivially small, so that the satisfiability count traverses a smaller diagram the more probesets leak. The two pairs that leave the set of probe variables unchanged isolate this effect, namely $(0,0,0)$ against $(0,1,0)$ and $(1,0,0)$ against $(1,1,0)$, where the runtime drops although only leaking probesets are added, e. g., from 1.933 s to 1.315 s for the hardened HPC_3 loop. The $(1,1,0)$-robust assessment therefore benefits twice, as it inherits the cheap stage input base case of the glitch extension and gains further leaking probesets from the transition substitution. Across constructions, the runtimes are not explained by circuit size alone, as the HPC_3 loop requires 0.912 s unextended at 80 wires but 10.404 s at 100 wires in its hardened variant, whereas the HPC_2 pair shows no comparable growth. We attribute this to structural properties of the gadgets and the resulting BDD- and MTBDD-based variable ordering.

The absence of partitioning imposes two limitations. First, the verification yields a single global bound for the entire unrolled circuit instead of a value per register stage, as reported in Table 5. This is the sound choice in the presence of transition probes, since probe sets may combine wires from both clock cycles, but it prevents attributing leakage to an individual register stage, which is precisely the granularity that enabled our analysis of the pipelined ISW multiplication in Section 5.1. For the same reason, these values are not directly comparable to the per-stage bounds given elsewhere in this section. Second, the probe set search space is no longer bounded by the largest register stage but by the unrolled circuit, whose sequential part is doubled. Together with the exponential growth of the search space in the number of wires, this confines the assessment to gadget-level constructions with small wire counts, while larger circuits, e. g., several gadgets per loop iteration or higher security orders, exceed a practical time budget. This limitation is

implementation-bound rather than methodological, as the leakage characteristic itself is agnostic to the probe extension. A decomposition that keeps the timing domain of each register intact, e. g., by partitioning along two-cycle windows rather than individual register stages, therefore presents a promising direction to restore the scalability of the $(1, 0, 0)$-robust analyses.

5.3 Random Probing Composability Assessment

To show the flexibility of the leakage characteristic across leakage models, we also integrated RPC validation as a baseline for its feasibility in practice. Table 7 reports the probability bounds for t-RPC on first- to second-order secure gadgets, where we opted for $t = \lfloor \frac{s-1}{2} \rfloor$ as recommended in [10], which is 1 for our benchmarks.

Table 7: Verification performance of our tool for selected AND gadgets under $(1, 0, 0)$-robust t-RPC conditions with $t = 1$ and $p = 2^{-8}$. All experiments were subject to a timeout limit (TO.) of 1h.

Scheme	Shares	Circuit			Verification	
	$d+1$	Wires	Comb.	Seq.	ε	Time
DOM	2	24	8	6	$4.057e^{-4}$	54 ms
	3	57	21	12	$2.908e^{-4}$	22.8 s
HPC_1	2	31	10	9	$1.644e^{-2}$	102 ms
	3	75	27	18	$3.485e^{-4}$	30.0 s
HPC_2	2	43	14	15	$2.162e^{-2}$	206 ms
	3	108	36	36	–	TO.
ISW	2	24	8	6	$1.611e^{-2}$	36 ms
	3	57	21	12	$2.908e^{-4}$	2.3 s

Unlike the preceding comparison with ironMask, this evaluation omits the insertion of copy-gates and therefore reflects the random probing composability of hardware circuits more accurately. Across the considered gadgets, the bound ε consistently decreases as the security order d increases, which matches the expectation that the larger probe sets required at higher orders occur with decreasing probability. Verification remains practical for first- and second-order constructions, whereas third-order verification exceeds the timeout.

The runtimes again vary beyond what circuit size explains. At second order, ISW and DOM share a wire count of 57, yet ISW completes in 2.26 s whereas DOM requires 22.85 s, and HPC_1 takes a comparable 30.05 s at 75 wires. As above, we attribute this to the randomness usage of the gadget and the induced probe-set search space. With 108 wires at second order, HPC_2 is the largest construction in the set and exceeds the one-hour timeout, thereby marking a current practical boundary of our proof-of-concept implementation and pointing to future work on higher-order masked circuits.

5.4 Verification of Compositions and Larger Constructions

Beyond individual gadgets, ironMask is constrained to the algebraic characterization of specific circuit structures and gadget types, notably Linear Randomness (LR) and Non-Linear Randomness (NLR) gadgets. Our approach instead supports arbitrary circuit structures, which enables the verification of larger constructions and of compositions of gadgets.

To exemplify this ability, we conducted a case study based on the PRESENT S-box, implemented using the HADES framework [12]. Specifically, we generated masked S-boxes utilizing P-INI-secure gadgets (HPC_1 and HPC_2) and a Probe Strong Non-Interference (P-SNI)-secure DOM multiplier for first- and second-order masking. The results of this evaluation are presented in Table 8.

Table 8: Verification results for masked PRESENT S-boxes (generated by HADES [12]) under $(1,0,0)$-robust RPS conditions ($p = 2^{-8}$). All experiments were subject to a timeout limit (TO.) of 1h.

Scheme	Shares	Circuit				Verification	
	$d+1$	Wires	Comb.	Seq.	Lat.	ε	Time
DOM	2	228	72	80	4	$\{4.83e^{-3}, 3.95e^{-4}, 3.20e^{-3}, 2.29e^{-4}, 7.61e^{-5}\}$	846 ms
	3	431	145	138	4	$\{5.51e^{-4}, 1.47e^{-5}, 2.52e^{-4}, 7.87e^{-6}, 5.93e^{-7}\}$	31.514 s
HPC_1	2	240	82	72	3	$\{2.49e^{-3}, 5.59e^{-4}, 5.41e^{-3}, 2.73e^{-4}\}$	868 ms
	3	468	166	129	3	$\{1.31e^{-4}, 2.52e^{-5}, 6.27e^{-4}, 7.31e^{-6}\}$	5.45 min
HPC_2	2	277	91	96	3	$\{3.06e^{-3}, 2.73e^{-4}, 5.26e^{-3}, 5.12e^{-4}\}$	4.159 s
	3	595	199	201	3	–	TO.

The tool verifies the compositional security of these constructions automatically and with full accuracy, providing exact leakage probability bounds for each register stage, which ironMask does not support.

6 Conclusion

In conclusion, this work introduces a robust and versatile methodology for automated security verification in the context of masking schemes, addressing both the (g,t,c)-robust threshold d-probing and (g,t,c)-robust random (p,ε)-probing models. By extending leakage characteristics to accommodate various probe extension schemes and linking them to the SIS function, we have advanced the state-of-the-art in security verification. Our practical implementation, leveraging BDDs, MTBDDs, and the FHT, demonstrates prevalent performance and scalability, as evidenced by comparative evaluations with ironMask and comprehensive case studies. Consequently, this work extends the field of automated hardware security verification, offering a versatile framework to assess the security of masked cryptographic implementations against SCA.

References

[1] Barthe, G., Belaïd, S., Cassiers, G., Fouque, P.A., Grégoire, B., Standaert, F.X.: maskVerif: Automated verification of higher-order masking in presence of physical defaults. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019, Part I. LNCS, vol. 11735, pp. 300–318. Springer, Cham (Sep 2019). 10.1007/978-3-030-29959-0_15

[2] Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y.: Verified proofs of higher-order masking. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 457–485. Springer, Berlin, Heidelberg (Apr 2015). 10.1007/978-3-662-46800-5_18

[3] Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 116–129. ACM Press (Oct 2016). 10.1145/2976749.2978427

[4] Beierle, C., Feldtkeller, J., Guinet, A., Güneysu, T., Leander, G., Richter-Brockmann, J., Sasdrich, P.: INDIANA - verifying (random) probing security through indistinguishability analysis. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VIII. LNCS, vol. 15608, pp. 33–63. Springer, Cham (May 2025). 10.1007/978-3-031-91101-9_2

[5] Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 616–648. Springer, Berlin, Heidelberg (May 2016). 10.1007/978-3-662-49896-5_22

[6] Belaïd, S., Cassiers, G.: PERSEUS – probabilistic evaluation of random probing SEcurity using efficient sampling. Cryptology ePrint Archive, Report 2025/1884 (2025), https://eprint.iacr.org/2025/1884

[7] Belaïd, S., Coron, J.S., Prouff, E., Rivain, M., Taleb, A.R.: Random probing security: Verification, composition, expansion and new constructions. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 339–368. Springer, Cham (Aug 2020). 10.1007/978-3-030-56784-2_12

[8] Belaïd, S., Mercadier, D., Rivain, M., Taleb, A.R.: IronMask: Versatile verification of masking security. In: 2022 IEEE Symposium on Security and Privacy. pp. 142–160. IEEE Computer Society Press (May 2022). 10.1109/SP46214.2022.9833600

[9] Belaïd, S., Rivain, M., Rossi, M.: New techniques for random probing security and application to raccoon signature scheme. In: Fehr, S., Fouque, P.A. (eds.) EUROCRYPT 2025, Part VIII. LNCS, vol. 15608, pp. 94–123. Springer, Cham (May 2025). 10.1007/978-3-031-91101-9_4

[10] Belaïd, S., Rivain, M., Taleb, A.R.: On the power of expansion: More efficient constructions in the random probing model. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 313–343. Springer, Cham (Oct 2021). 10.1007/978-3-030-77886-6_11

[11] Bloem, R., Groß, H., Iusupov, R., Könighofer, B., Mangard, S., Winter, J.: Formal verification of masked hardware implementations in the presence of glitches. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 321–353. Springer, Cham (Apr / May 2018). 10.1007/978-3-319-78375-8_11

[12] Buschkowski, F., Land, G., Höher, N., Richter-Brockmann, J., Sasdrich, P., Güneysu, T.: HADES: Automated hardware design exploration for cryptographic primitives. IACR TCHES **2025**(4), 1–45 (2025). 10.46586/tches.v2025.i4.1-45

[13] Cassiers, G., Faust, S., Orlt, M., Standaert, F.X.: Towards tight random probing security. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 185–214. Springer, Cham, Virtual Event (Aug 2021). 10.1007/978-3-030-84252-9_7

[14] Cassiers, G., Standaert, F.: Trivially and efficiently composing masked gadgets with probe isolating non-interference. IEEE Trans. Inf. Forensics Secur. **15**, 2542–2555 (2020). 10.1109/TIFS.2020.2971153, https://doi.org/10.1109/TIFS.2020.2971153

[15] Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 398–412. Springer, Berlin, Heidelberg (Aug 1999). 10.1007/3-540-48405-1_26

[16] Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Berlin, Heidelberg (May 2014). 10.1007/978-3-642-55220-5_24

[17] Faust, S., Grosso, V., Merino Del Pozo, S., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR TCHES **2018**(3), 89–120 (2018). 10.13154/tches.v2018.i3.89-120, https://tches.iacr.org/index.php/TCHES/article/view/7270

[18] Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Çetin Kaya., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Berlin, Heidelberg (May 2001). 10.1007/3-540-44709-1_21

[19] Gigerl, B., Hadzic, V., Primas, R., Mangard, S., Bloem, R.: CoCo: Co-design and co-verification of masked software implementations on CPUs. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/gigerl

[20] Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Berlin, Heidelberg (Aug 2003). 10.1007/978-3-540-45146-4_27

[21] Knichel, D., Moradi, A.: Low-latency hardware private circuits. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 1799–1812. ACM Press (Nov 2022). 10.1145/3548606.3559362

[22] Knichel, D., Sasdrich, P., Moradi, A.: SILVER - statistical independence and leakage verification. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 787–816. Springer, Cham (Dec 2020). 10.1007/978-3-030-64837-4_26

[23] Knichel, D., Sasdrich, P., Moradi, A.: Generic hardware private circuits towards automated generation of composable secure gadgets. IACR TCHES **2022**(1), 323–344 (2022). 10.46586/tches.v2022.i1.323-344

[24] Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer, Berlin, Heidelberg (Aug 1996). 10.1007/3-540-68697-5_9

[25] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 388–397. Springer, Berlin, Heidelberg (Aug 1999). 10.1007/3-540-48405-1_25

[26] Moos, T., Moradi, A., Schneider, T., Standaert, F.X.: Glitch-resistant masking revisited. IACR TCHES **2019**(2), 256–292 (2019). 10.13154/tches.v2019.i2.256-292, https://tches.iacr.org/index.php/TCHES/article/view/7392

[27] Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Berlin, Heidelberg (May 2013). 10.1007/978-3-642-38348-9_9