INDIANA – Verifying (Random) Probing Security through Indistinguishability Analysis

Christof Beierle[1], Jakob Feldtkeller[1], Anna Guinet[1], Tim Güneysu[1,2], Gregor Leander[1], Jan Richter-Brockmann[1], Pascal Sasdrich[1],

[1] Ruhr-University Bochum, Bochum, Germany
`firstname.lastname@rub.de`
[2] DFKI GmbH, Bremen, Germany

Abstract. Despite masking being a prevalent protection against passive side-channel attacks, implementing it securely in hardware remains a manual, challenging, and error-prone process.
This paper introduces INDIANA, a comprehensive security verification tool for hardware masking. It provides a hardware verification framework, enabling a complete analysis of simulation-based security in the glitch-extended probing model, with cycle-accurate estimations for leakage probabilities in the random probing model. Notably, INDIANA is the first framework to analyze arbitrary masked circuits in both models, even at the scale of full SPN cipher rounds (e.g., AES), while delivering exact verification results. To attain precise and extensive verifiability, we introduce a partitionable probing distinguisher that enables rapid verification of probe tuples, outperforming state-of-the-art methods based on statistical independence. Most interestingly, our approach inherently facilitates extensions to the random probing model by leveraging Fast Fourier-Hadamard Transformations (FFTs).
Benchmark results show that INDIANA competes effectively with leading probing model verification tools, such as maskVerif and VERICA. Notably, INDIANA is the first tool that is capable to provide cycle-accurate estimations of random probing leakage probabilities for large-scale masked circuits.

Keywords: Indistinguishability Analysis · Side-Channel Analysis · Probing Security · Random Probing Security · Security Verification · Discrete Probability Distribution

1 Introduction

Ever since the discovery of Side-Channel Analysis (SCA) [34, 35], it has been widely acknowledged that hardware implementations may inadvertently disclose sensitive information while executing cryptographic algorithms.

While *masking* has been recognized as a sound and effective countermeasure, extensively applied in practice, its correct and secure implementation in hardware circuits continues to pose a significant challenge. In particular, the abundance of proposed schemes [15, 27–31, 33, 41, 43], along with a notable portion proven to be vulnerable [38], bears witness to the fact that design and implementation of hardware masking schemes still remains a complex, delicate, and error-prone process.

To address this challenge in a systematic approach, formal definition of adversary models [6, 19, 30, 42], security properties [4, 5, 16, 18], and architectural conditions [6, 20] is absolutely necessary. Instead, adhering to such formal criteria is essential for streamlining, accelerating, and automating the evaluation of implementation correctness and security.

Consequently, to assess the security of hardware circuits, particularly in the context of masked implementations and passive implementation attacks, these criteria are generally formalized and systematized in terms of leakage models. In this context, a widely accepted model is the d-probing model, introduced by Ishai, Sahai, and Wagner in 2003 [30], where the attacker's view on leakage is represented by the choice and exact knowledge of d intermediate variables. This model was later complemented by Faust et *al.* [20] with consideration of unintentional physical effects naturally occurring in modern CMOS technology, e.g., glitches, transitions, and coupling effects.

Although probing-based leakage models have been shown to facilitate and accelerate security reasoning, they occasionally fall short in accurately representing the intricacies of hardware circuits. Notably, these models neglect certain aspects, such as horizontal attacks [8], which specifically exploit the repeated manipulation of variables within iterative implementations. As a result, the research community is increasingly leaning towards more realistic leakage models, such as the random probing model [19, 30] where leakage is assumed to capture the exact value carried by each wire of the circuit with a probability p which is intricately linked to the security in the most realistic noisy leakage model [17], where each variable leaks a noisy function of its value.

As leakage models continually advance and security proofs become more complex, there is a growing interest within the community in automating formal security verification. As a result, the development and implementation of automated security reasoning tools in the realm of hardware masking has emerged as a recent and evolving area of research [2, 3, 10, 11, 15, 25, 32, 44].

Being a pioneering work, rebecca [11] employs Fourier coefficient estimation to verify standard and glitch-extended probing models. Its successor, COCO [25], extends this methodology to verify masked software running on processors. Shortly after, maskVerif [3] was introduced as an efficient probing security tool; however, its incomplete verification may result in false negatives. Additionally, scVerif [7] extends maskVerif by incorporating customizable leakage models. SILVER [32], the first tool to ensure comprehensive verification, however, is limited by constraints in circuit size and security order. VERICA [44] is an extension of SILVER with Fault Injection Analysis (FIA) support, which allows combined analysis for the first time, but maintains similar complexity limitations. Most recently, PROLEAD [39] applies logic simulation for probing-based leakage assessment.

Expanding the scope of security verification research to the random probing model, VRAPS [9] specifically addresses random probing security, although it is notably limited in terms of efficiency. Similarly, STRAPS [14] adopts the rules from maskVerif to quantify the random probing security, but inheriting the constraints of the latter tool. The latest contribution in this domain, ironMask [10], stands out as a high-performing and complete tool, but is strictly constrained to small masking circuits with specific structure, i.e., only supports verification for linear randomness and quadratic gadgets.

Concerning pertinent prior research, it becomes evident that these tools either rely on estimates and heuristics, consequently delivering incomplete results, or they exhibit significant restrictions in terms of the structure and size of masked circuits they can handle. Further, all existing tools also have limited capabilities in accommodating more realistic leakage models, such as those considering glitch-extended d-probing or the random probing leakage model. As a result, the open research challenge is to develop sound, accurate and efficient tools for verifying the security of arbitrary and complex masked hardware circuits under more realistic leakage models (e.g., considerations of glitches or random probing leaks) to overcome the existing limitations.

Contributions. In this work, we introduce INDIANA, a comprehensive and powerful automated verification tool designed for verifying security of masked hardware. This tool facilitates effective security validation for large-scale digital circuits, accommodating both the glitch-extended and random probing security models. To outline, our primary contributions can be succinctly described as follows:

- We initially formalize the prevailing d-probing security concepts by expressing them in terms of the *indistinguishability* of secret-dependent probability distributions observed by adversaries. This formalization is specifically tailored for hardware-oriented masking verification, addressing and acknowledging the glitch-extended robust d-probing model. In essence, commencing with secret-dependent input distributions and the gate-level netlist of a masked circuit, these distributions continuously transition through the circuit. This process makes it possible to partition the verification procedure into iterative steps, generating discrete multi-variate probability distributions for adversarial observations (i.e., probed intermediate wires). The circuit is verified secure if an attacker cannot differentiate all secret-dependent distributions for all observations.
- We extend our primary d-probing security distinguisher to accommodate the more realistic random probing security model by employing Fast Fourier-Hadamard Transformations (FFTs). This extension, in particular, enables the conversion of differences in observed secret-dependent probability distributions into corresponding sets of probes responsible for the information leakage. To elaborate, the utilization of the FFT yields a succinct representation of all probe tuples contributing to the leak, facilitating the swift calculation of leakage probabilities in the context of random probing security.

– We present INDIANA, a complete and versatile verification framework designed for the rapid validation of large-scale masked hardware circuits within the glitch-extended probing model. Furthermore, INDIANA facilitates cycle-accurate estimation of leakage probabilities in the random probing model for arbitrary large-scale circuits (and exact computation for small-scale designs). We showcase the exceptional performance and capabilities of our novel verification tool through comparison to state-of-the-art competitors and various case studies, including the instantiation of a full AES round (i.e., 16 masked S-boxes in parallel) with shared data and key inputs. This results in the (symbolic) simulation of 2^{256} distinct secrets (resulting in 2^{512} different shared input combinations). Notably, INDIANA achieves exhaustive verification in the glitch-extended probing model within less than 45 minutes for the full round of AES, while the cycle-accurate estimation of leakage probabilities in the random probing model terminates in less than 4 hours.

Outline. The remainder of this work is organized as follows. Section 2 introduces fundamental concepts for side-channel security, including our considered circuit model, the adversary model and the essential definitions of leakage and security. In this context, for the first time, we present the definitions for threshold d-probing security and random probing security based on *indistinguishability* and discover an interesting link to d–resiliency, which we can apply for efficient leakage detection using the FFT. In Section 3, we then discuss approaches and optimizations for the practical implementation of security verification. In particular, we use novel techniques based on Binary Decision Diagrams (BDDs) and Multi-Terminal Binary Decision Diagrams (MTBDDs) to practically verify the indistinguishability-based definitions and perform leakage detection in the random probing model. Eventually, Section 4 presents several experiments and case studies to empirically demonstrate the capabilities and limitations of our novel verification tool INDIANA, before the paper concludes in Section 5.

Notation. Our basic notation used throughout this work is summarized in Table 1. In general, we write functions in sans serif font (e.g., F) and sets in upper-case characters in a calligraphic font (e.g., $\mathcal{S}$). For a set $\mathcal{S}$, we denote by $2^{\mathcal{S}}$ its power set, i.e., the set of all subsets of $\mathcal{S}$.

2 Side-Channel Security

In SCA an adversary exploits the correlation of physical characteristics, such as timing behavior [34], instantaneous power consumption [35], or electromagnetic emanations [24], with a secret-dependent internal state to gain knowledge of a secret. The physical reality of such attacks is complex and often highly stochastic. Therefore, the research community has focused on the development and analysis of *security models* that abstract reality to its core component to enable formal reasoning about SCA security. In the following section, we describe

Table 1: Notations used throughout this work.

	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	Digital logic circuit
	$\mathcal{W}$	Set of wires
SCA	s	Number of shares used by a masking scheme
	d	Security order of a masking scheme (countermeasure)
	$\mathcal{P}$	Set of probes
	Enc	Encoding function for masking
	A_C	Access function to circuit C
	Ex	Probe-extension function
	$\mathcal{L}()$	Adversarial observation (leakage)
	$\texttt{ProbeSelect}()$	Random variable for probe selection

the formal context of this work. In particular, we formally define our model for hardware circuits and describe *masking* as a popular countermeasure against SCA. Then, we define the well-known threshold d-probing and random probing model [30] based on indistinguishability of secret-dependent distributions, which is equivalent to the standard simulation-based definitions found in literature.

2.1 Circuit Model

We model a circuit as a Direct Acyclic Graph (DAG) $\mathsf{C} = (\mathcal{G}, \mathcal{W})$, where each vertex (or node) $g \in \mathcal{G}$ represents a gate, an input, or an output of the circuit and each edge $w \in \mathcal{W} = \{w_0, \ldots, w_{|\mathcal{W}|-1}\}$ a wire that carries a value from $\mathbb{F}_2$ and connects two gates. Without loss of generality, we restrict the set of logical gates to $\mathcal{G}_\mathsf{c} = \{\mathsf{xor}, \mathsf{or}, \mathsf{and}, \mathsf{nand}, \mathsf{xnor}, \mathsf{nor}, \mathsf{inv}\}$, which each implement the respective Boolean function. We further restrict the set of memory gates to $\mathcal{G}_\mathsf{m} = \{\mathsf{reg}\}$, where reg represents a clocked register. Let $\mathcal{G}_\mathsf{all}$ be $\mathcal{G}_\mathsf{c} \cup \mathcal{G}_\mathsf{m}$. For convenience, we sometimes assume $\mathcal{W}$ to be an index set, where each wire $w_i \in \mathcal{W}$ is labeled by its index i, i.e., we will use $\mathcal{W} = \{w_0, \ldots, w_{|\mathcal{W}|-1}\}$ and $\mathcal{W} = \{0, \ldots, |\mathcal{W}| - 1\}$ interchangeably.

Definition 1 (Circuit). *A* circuit *with n_i inputs and n_o outputs is a DAG $\mathsf{C} = (\mathcal{G}, \mathcal{W})$, where each vertex takes a label from $\{x_1, \ldots, x_{n_i}, y_1, \ldots, y_{n_o}\} \cup \mathcal{G}_\mathsf{all}$, and which fulfills the following properties:*

1. *For each $j \in \{1, \ldots, n_i\}$ there is exactly one vertex labeled x_j and for each $k \in \{1, \ldots, n_o\}$ there is exactly one vertex labeled y_k.*
2. *Each vertex with a label x_j has in-degree 0 and each vertex with label y_k has in-degree 1 and out-degree 0.*
3. *Each vertex with a label in $\{\mathsf{xor}, \mathsf{or}, \mathsf{and}, \mathsf{nand}, \mathsf{xnor}, \mathsf{nor}\}$ has in-degree 2.*

4. Each vertex with label inv *or* reg *has in-degree* 1.

A circuit is a representation of a function $\mathsf{F}\colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_o}$, that is computed by going iteratively through the gates and assign each wire with the result of the respective logical gate given the values of the input wires. In this sense, each wire $w \in \mathcal{W}$ computes a function $\mathsf{F}_w\colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2$ that is computed by considering the subgraph with the leave w. This motivates the definition of an access function, where we consider each wire $w \in \mathcal{W}$ to be connected to an output node. Hence, intuitively this function provides access to all internal values of C.

Definition 2 (Access to a Circuit). *Given a circuit* $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ *that implements a function* $\mathsf{F}\colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_o}$*. The* access to C *is a function* $\mathsf{A}_\mathsf{C}\colon \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{|\mathcal{W}|}$ *where the coordinates of* A_C *correspond to all the edges in* C*, i.e., for* $x \in \mathbb{F}_2^{n_i}$*, we have* $\mathsf{A}_\mathsf{C}(x) = (w_0, \dots, w_{|\mathcal{W}|-1})$*, where* $w_i \in \mathbb{F}_2$ *takes the value of the i-th wire in* C *when the input nodes of* C *are instantiated with* x*.*

Note that the notion of the access to a circuit depends on the labeling of the wires. Throughout this work, we assume lexicographic ordering. Further, for a function F mapping to $\mathbb{F}_2^{n_o}$ and a subset $\mathcal{I} \subseteq \{0, \dots n_o - 1\}$ of output indices, we denote by $\mathsf{F}|_\mathcal{I} = (\mathsf{F}_i)_{i \in \mathcal{I}}$ the subfunction of F that consists of the coordinates of $\mathcal{I}$, mapping to $\mathbb{F}_2^{|\mathcal{I}|}$. Again, we assume that the coordinates of $\mathsf{F}|_\mathcal{I}$ are ordered by the lexicographic ordering of the indices in $\mathcal{I}$.

2.2 Security Mechanism: Masking

One widely-studied countermeasure against SCA is *Boolean masking* [17], where every secret $x_i \in \mathbb{F}_2$ is replaced by a vector $\mathsf{Sh}(x) = \langle x_{i,0}, \dots x_{i,s-1}\rangle \in \mathbb{F}_2^s$ such that each subset $\hat{\mathcal{X}}$ of $\{x_{i,j} \mid j \in [0, s-1]\}$ with $|\hat{\mathcal{X}}| < s$ is independent of $x_i = \bigoplus_{j=0}^{s-1} x_{i,j}$. We call $x_{i,j}$ a *share* of x_i with *share index* j.

Similarly, a circuit is transformed into a *shared circuit* by transforming each gate into a set of gates that operate on shared values [30]. Specifically, every circuit $\tilde{\mathsf{C}}$, with n_i inputs and n_o outputs, is split into three circuits with the following properties:

$\mathsf{Enc}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \to \mathbb{F}_2^{n_i \cdot s}$: A circuit implementing a function that takes as input the n_i inputs to the original circuit $x_1 \dots x_{n_i}$ and $(s-1)n_i$ random bits $r \in \mathbb{F}_2^{(s-1)n_i}$ and outputs a valid sharing of the input, i.e., for all $i \le n_i$ an element from $\mathbb{F}_2^s$ with the above property[3] towards the input x_i.

$\mathsf{C}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$: A masked version of the original circuit, that takes as input the shared values of the inputs and n_r random bits and produces a valid sharing of the output, i.e., for all $j \le n_o$ an element of $\mathbb{F}_2^s$ with the above property towards the output y_j.

[3] Due to its practical relevance, we focus on Boolean masking here, but Enc can easily be extended to any other masking scheme.

$\mathsf{Dec}\colon \mathbb{F}_2^{n_o \cdot s} \to \mathbb{F}_2^{n_o}$: A circuit that implements a function that given a valid sharing of n_o values produces the unshared values by summing up the shares, i.e., for all $i \leq n_o$ computes $y_i = \bigoplus_{j=0}^{s-1} y_{i,j}$.

For the correctness of the masked circuit, we require that for all inputs $x \in \mathbb{F}_2^{n_i}$, $r_0 \in \mathbb{F}_2^{(s-1)n_i}$, and $r_1 \in \mathbb{F}_2^{n_r}$ it holds that $\tilde{\mathsf{C}}(x) = \mathsf{Dec}(\mathsf{C}(\mathsf{Enc}(x, r_0), r_1))$. Note, if we compute the access A_C of the masked circuit C, we consider only the wires in C, not those in Enc or Dec. In addition, we define the composition $\mathsf{H} := \mathsf{A}_\mathsf{C} \circ \mathsf{Enc}$ to be a function mapping from $\mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i + n_r}$ to $\mathbb{F}_2^{|\mathcal{W}|}$ such that $\mathsf{H}(x, r_0 \| r_1) = \mathsf{A}_\mathsf{C}(\mathsf{Enc}(x, r_0), r_1)$ for all $x \in \mathbb{F}_2^{n_i}$, $r_0 \in \mathbb{F}_2^{(s-1)n_i}$, and $r_1 \in \mathbb{F}_2^{n_r}$.

2.3 Probing Security

A well-established model to argue about side-channel security of masked circuits is the *d-probing model* [30]. Here, an adversary $\mathcal{A}_p$ gets access to a circuit C that can be invoked multiple times. Prior to each invocation, $\mathcal{A}_p$ can select a set $\mathcal{P} \subseteq \mathcal{W}$ of up to d wires to be probed. In the *standard* d-probing model, the view of $\mathcal{A}_p$ on each invocation is defined by the exact values of the probed wires. More specifically, we can express the view of $\mathcal{A}_p$ for a set of probes $\mathcal{P}$ by a subfunction of the encoded access to the circuit C, i.e., by $(\mathsf{A}_\mathsf{C} \circ \mathsf{Enc})|_{\mathcal{P}}$.

However, in hardware physical defaults such as glitches, transitions, or couplings can cause additional leakage and need to be considered in the security model. For this, the *robust* d-probing model was introduced [20] that extends the set of probes $\mathcal{P}$ by additional probes to capture worst-case assumptions of physical defaults. For example, glitches are temporary and unintentional values in a combinatorial circuit, caused by timing differences in the computation paths. In the robust d-probing model, those are represented by giving $\mathcal{A}_p$ access to the exact values of the stable inputs to the probes, i.e., registers or primary inputs with no register on the path to the probed wires. To formally capture physical defaults, we define a probe-extension function $\mathsf{Ex}(\mathcal{P})$ for a subset of wires $\mathcal{P} \subseteq \mathcal{W}$ that produces another subset of wires $\mathcal{P}' \subseteq \mathcal{W}$. This allows us to model any additional leakage that is a function of the probed wires, which contains (but is not restricted to) the probe extensions defined in the robust d-probing model. With this, we can again capture the view of the adversary by a subfunction of the access of the circuit C, namely $(\mathsf{A}_\mathsf{C} \circ \mathsf{Enc})|_{\mathsf{Ex}(\mathcal{P})}$. Note, when setting Ex to the identity function id, we get the standard definition of the d-probing model.

In this context, a circuit C is d-probing secure if any set $\mathcal{P}$ of up to d probes is statistically independent of the secret input [30, 32]. For this work, we reformulate the definition based on indistinguishability. In short, two events $\mathcal{X}$ and $\mathcal{Y}$ are indistinguishable if the two underlying distributions of $\mathcal{X}$ and $\mathcal{Y}$ are equal. To formally capture the definition of probing security, we introduce a leakage function based on indistinguishability.

Definition 3 (Leakage). *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 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* C *implementing a* *function* $\mathsf{F}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$*. Let* $\mathsf{H} := \mathsf{A_C} \circ \mathsf{Enc}$*, which is a function from* $\mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r}$ *to* $\mathbb{F}_2^{|\mathcal{W}|}$*. We define*

$$\mathcal{L}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex})}\colon 2^{\mathcal{W}} \to \{0,1\}$$

$$\mathcal{P} \mapsto \begin{cases} 0, \textit{if } \forall x \in \mathbb{F}_2^{n_i}, y \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|} : \mathsf{Pr}_r[\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}(x,r) = y] = \mathsf{Pr}_r[\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}(0,r) = y] \\ 1, \textit{else} \end{cases}$$

to be the leakage of the probeset $\mathcal{P}$ *in the masked circuit* C*, where the probabilities are taken over uniformly random choices of* $r \in \mathbb{F}_2^{(s-1)n_i+n_r}$*.*

Then, for probing security, we require that there is no leakage regardless of the probe position, for a probe cardinality at most d.

Definition 4 (Probing Security - Indistinguishability). *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 and* $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ *be a circuit implementing a function* $\mathsf{F}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$*. We say that* C *is* d*-probing secure with respect to* Enc *and a probe extension* Ex *if the access* $\mathsf{A_C}$ *to* C *fulfills* $\mathcal{L}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex})}(\mathcal{P}) = 0$ *for all sets* $\mathcal{P} \subseteq \mathcal{W}$ *of up to* d *probes.*

Intuitively, the definition for d-probing security fixes the view of $\mathcal{A}_p$ (by fixing the set of probes $\mathcal{P}$) and then requires that $\mathcal{A}_p$ cannot distinguish between different input values $x \in \mathbb{F}_2^{n_i}$. It is easy to see that this is equivalent to a definition requiring statistical independence between the set of probes $\mathcal{P}$ and the input x [32].

2.4 Random Probing Security

Another established security model for SCA is the *random probing model* [30]. Here, instead of letting the adversary choose a bounded set of probes, an unbounded set of probes is randomly selected and given to the adversary, such that each wire $w \in \mathcal{W}$ is in the set of probes with the same probability p. For this, let $\texttt{ProbeSelect}(\mathsf{C}, p)$, cf. $\mathsf{LeakingWires}$ in [9], be a random variable that, given a circuit C and a probability p, probabilistically chooses a set of probes $\mathcal{P}$, such that each wire $w \in \mathcal{W}$ is probed with probability p independent of all other wires, i.e., $\mathsf{Pr}[w \in \mathcal{P}] = p$. Note, in contrast to the d-probing model, the set $\mathcal{P}$ is not restricted in size. Then, for random probing security, we require the probability that the adversary gets a useful set of probes to be small.

Definition 5 (Random Probing Security - Indistinguishability). *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 and* $\mathsf{C} = (\mathcal{G}, \mathcal{W})$ *be a circuit implementing a function* $\mathsf{F}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$*. We say that* C *is* (p, ϵ)*-random probing secure with respect to* Enc *and probe extension* Ex *if the access* $\mathsf{A_C}$ *to* C *fulfills*

$$\mathsf{Pr}[\mathcal{L}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex})}(\texttt{ProbeSelect}(\mathsf{C}, p)) = 1] \leq \epsilon\,.$$

For the standard definition of the random probing model the probe extension function Ex is selected to be the identity function id, i.e., there is no probe extension. Then, it is easy to see, that this definition is equivalent to the simulation-based definition from Belaïd et al. [9], where the simulation of the probes fails with probability ϵ.

2.5 Using FFT for Detecting Leakage

We show how to evaluate the leakage function by computing the Fourier-Hadamard transforms of some associated functions. First, we recall that the *Fourier-Hadamard transform* $\widehat{\mathsf{f}}$ of a function $\mathsf{f}\colon \mathbb{F}_2^m \to \mathbb{R}$ is defined as

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

where $\langle u, v\rangle$ denotes the canonical inner product of the vectors $u, v \in \mathbb{F}_2^m$. For a subset $\mathcal{I}$ of $\{0, 1, \ldots, m-1\}$, we denote by $\operatorname{prec}(\mathcal{I})$ the $|\mathcal{I}|$-dimensional subspace $\{\beta \in \mathbb{F}_2^m \mid \beta_j = 0 \text{ if } j \notin \mathcal{I}\}$ of $\mathbb{F}_2^m$.

Given a function $\mathsf{F}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{n_o}$ and $x \in \mathbb{F}_2^{n_i}$, we denote by F_x the function obtained from F by fixing x, i.e., $\mathsf{F}_x(z) = \mathsf{F}(x, z)$. For $y \in \mathbb{F}_2^{n_o}$, we denote by $(\mathsf{F})_x^{-1}(y)$ the preimages of y under F_x, i.e., the set of $z \in \mathbb{F}_2^{(s-1)n_i+n_r}$ such that $\mathsf{F}_x(z) = y$. For a given $\mathsf{H}\colon \mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i+n_r} \to \mathbb{F}_2^{|\mathcal{W}|}$ and $y \in \mathbb{F}_2^{|\mathcal{W}|}$, let us denote $\mathsf{D}_{\mathsf{H},x}(y) \coloneqq |(\mathsf{H})_x^{-1}(y)| - |(\mathsf{H})_0^{-1}(y)| \in \mathbb{R}$. Then, $\mathsf{D}_{\mathsf{H},x}\colon y \mapsto \mathsf{D}_{\mathsf{H},x}(y)$ is a function mapping from $\mathbb{F}_2^{|\mathcal{W}|}$ to $\mathbb{R}$. The main result of this subsection is the following link between the leakage function and the Fourier-Hadamard transform of the functions $\mathsf{D}_{\mathsf{H},x}$. Note that, in the following we identify the wires by their indices, i.e., $\mathcal{W} = \{0, 1, \ldots, |\mathcal{W}| - 1\}$.

Theorem 1. *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 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* C *implementing a function* $\mathsf{F}\colon \mathbb{F}_2^{n_i\cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o\cdot s}$. *Let* $\mathsf{H} = \mathsf{A}_\mathsf{C} \circ \mathsf{Enc}$. *Then,* $\mathcal{L}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex})}(\mathcal{P}) = 0$ *if and only if, for all* $x \in \mathbb{F}_2^{n_i}$, *we have*

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

The advantage of this characterization is the fact that the Fourier-Hadamard transform of a function $\mathsf{f}\colon \mathbb{F}_2^m \to \mathbb{R}$ can be computed using $\mathcal{O}(m\cdot 2^m)$ elementary arithmetic operations by using the *Fast Fourier-Hadamard Transform (FFT)*, see [13, pp.53–54].

For the proof, we use the following two lemmas.

Lemma 1. *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 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* C *implementing a function* $\mathsf{F}\colon \mathbb{F}_2^{n_i\cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o\cdot s}$. *Then,* $\mathcal{L}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex})}(\mathcal{P}) = 0$ *if and only if*

$$\forall x \in \mathbb{F}_2^{n_i}, y \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}\colon |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_x^{-1}(y)| = |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_0^{-1}(y)|.$$

In particular, C *is* d*-probing secure with respect to* Enc *and* Ex *if and only if, for the access* $\mathsf{A_C}$ *to* C*, we have*

$$\forall \mathcal{P} \in 2^{\mathcal{W}} \text{ with } |\mathcal{P}| \leq d \colon \forall x \in \mathbb{F}_2^{n_i}, y \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|} \colon |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_x^{-1}(y)| = |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_0^{-1}(y)|.$$

Proof. This is an immediate consequence of the fact that

$$\mathsf{Pr}_r[\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}(x,r) = y] = \frac{|\{r \in \mathbb{F}_2^{(s-1)n_i+n_r} \mid \mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}(x,r) = y\}|}{2^{(s-1)n_i+n_r}} = \frac{|(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_x^{-1}(y)|}{2^{(s-1)n_i+n_r}}.$$

□

Lemma 2. *Let* $\mathsf{f} \colon \mathbb{F}_2^m \to \mathbb{R}$*. Then,* f *is constant and equal to zero if and only if* $\widehat{\mathsf{f}}$ *is constant and equal to zero.*

Proof. This follows directly from the well-known inverse Fourier-Hadamard relation $\widehat{\widehat{\mathsf{f}}} = 2^m \cdot f$, see [13, Cor.4, p.59]. □

Proof (of Thm.1). Let us fix one input $x \in \mathbb{F}_2^{n_i}$. For easier notation, we omit the index H, x in $\mathsf{D}_{\mathsf{H},x}$ and simply write D instead. Let us fix $\mathcal{P} \in 2^{\mathcal{W}}$, which is a subset of $\{0, 1, \ldots, |\mathcal{W}| - 1\}$.

For $y \in \mathbb{F}_2^{|\mathcal{W}|}$, we denote by $\pi_{\mathsf{Ex}(\mathcal{P})}(y)$ the projection of y to the coordinates indexed by $\mathsf{Ex}(\mathcal{P})$. Then, $\pi_{\mathsf{Ex}(\mathcal{P})}(y) = y'$ for $y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$. Let us define

$$\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}(y') := \sum_{y \in \mathbb{F}_2^{|\mathcal{W}|}, \pi_{\mathsf{Ex}(\mathcal{P})}(y) = y'} \mathsf{D}(y). \tag{1}$$

We have $|(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_x^{-1}(y')| - |(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_0^{-1}(y')| = \mathsf{D}_{\mathsf{Ex}(\mathcal{P})}(y')$, hence, by Lemma 1, we have $\mathcal{L}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex})}(\mathcal{P}) = 0$ if and only if $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}(y') = 0$ for all $y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$ (and all such functions $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}$ defined by all $x \in \mathbb{F}_2^{n_i}$). Thus, we are going to show that $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}(y') = 0$ for all $y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$ holds if and only if $\widehat{\mathsf{D}}(\beta) = 0$ holds for all $\beta \in \mathrm{prec}(\mathsf{Ex}(\mathcal{P}))$.

Clearly, the mapping $\beta \mapsto \pi_{\mathsf{Ex}(\mathcal{P})}(\beta)$ is a vector space isomorphism from $\mathrm{prec}(\mathsf{Ex}(\mathcal{P}))$ to $\mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$. Let $\beta \in \mathrm{prec}(\mathsf{Ex}(\mathcal{P}))$. Then,

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

Now, if $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}(y') = 0$ holds for all $y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$, clearly, by the above equation we have $\widehat{\mathsf{D}}(\beta) = 0$ for all $\beta \in \mathrm{prec}(\mathsf{Ex}(\mathcal{P}))$. For the converse, let us assume that $\widehat{\mathsf{D}}(\beta) =$

0 for all $\beta \in \mathrm{prec}(\mathsf{Ex}(\mathcal{P}))$. By the above equation, $\widehat{\mathsf{D}}(\beta)$ is equal to the Fourier-Hadamard transform at point $\pi_{\mathsf{Ex}(\mathcal{P})}(\beta)$ of the function $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})} \colon \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|} \to \mathbb{R}$. Hence, $\widehat{\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}} = 0$ by assumption. By Lemma 2, it follows that $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}(y') = 0$ for all $y' \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$, which concludes the proof. □

Theorem 1 gives us a way to compute the leakage as follows: For $x \in \mathbb{F}_2^{n_i}$, let $\mathsf{g}_{\mathsf{H},x} \colon \mathbb{F}_2^{|\mathcal{W}|} \to \mathbb{F}_2$ be the Boolean function defined by $\mathsf{g}_{\mathsf{H},x}(\beta) = 0$ if and only if $\widehat{\mathsf{D}_{\mathsf{H},x}}(\beta) = 0$ and $\mathsf{g}_{\mathsf{H},x}(\beta) = 1$ otherwise. Then,

$$\mathcal{L}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex})}(\mathcal{P}) = \bigvee_{x \in \mathbb{F}_2^{n_i}} \bigvee_{\beta \in \mathrm{prec}(\mathsf{Ex}(\mathcal{P}))} \mathsf{g}_{\mathsf{H},x}(\beta). \tag{2}$$

This formula is useful to assess the security of a circuit in the random probing model, where we need to analyze $\mathsf{Pr}[\mathcal{L}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex})}(\mathtt{ProbeSelect}(\mathsf{C}, p)) = 1]$.

The case of size-preserving probe-extension functions. In the special case where Ex is bijective and preserves the size of its input set (for example when $\mathsf{Ex} = \mathsf{id}$), Theorem 1 implies a link between d-probing security and d-resiliency of the functions $\mathsf{D}_{\mathsf{H},x}$. A function $\mathsf{f} \colon \mathbb{F}_2^m \to \mathbb{R}$ is called *d-resilient*[4] if $\widehat{\mathsf{f}}$ is zero on all inputs with Hamming weight less than or equal to d.

Corollary 1. *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 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 C implementing a function $\mathsf{F} \colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$. Let Ex be a bijective probe-extension function that preserves the size of its input set. Then, the following assertions are equivalent:*

1. *C is d-probing secure with respect to Enc and Ex.*
2. *The functions $\mathsf{D}_{\mathsf{H},x}$ are d-resilient for all $x \in \mathbb{F}_2^{n_i}$.*

Proof. By definition, C is d-probing secure with respect to Enc and Ex if and only if $\mathcal{L}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex})}(\mathcal{P}) = 0$ for all $\mathcal{P} \subseteq \mathcal{W}$ with $|\mathcal{P}| \leq d$. If Ex is bijective and preserves the size of its input set, this is equivalent to $\mathcal{L}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{id})}(\mathcal{P}) = 0$ for all $\mathcal{P} \subseteq \mathcal{W}$ with $|\mathcal{P}| \leq d$. The result now follows from Theorem 1 and the fact that the set $\{\beta \in \mathbb{F}_2^{|\mathcal{W}|} \mid \mathrm{wt}(\beta) \leq d\}$ is equal to the union of all sets $\mathrm{prec}(\mathcal{P})$ with $|\mathcal{P}| \leq d$. □

In this case, the maximum d for which a circuit is d-probing secure can be computed by Algorithm 1 given below. Note that in our application, the probe-extension function is more complex, so we do not actually use Algorithm 1. Still, we decided to list it for completeness.

3 Security Verification

This section explains the practical implementation of the formal models and indistinguishability analysis enabling an automated verification of the security definitions outlined in Section 2.

[4] In the literature, d-resiliency is usually defined for a Boolean function $\mathsf{g} \colon \mathbb{F}_2^m \to \mathbb{F}_2$. Then, the usual notion of d-resiliency corresponds to ours if f is the sign function of g, i.e., $\mathsf{f} = (-1)^{\mathsf{g}}$.

Algorithm 1: Finding the largest d such that C is d-probing secure with respect to Enc and Ex, where Ex is bijective and preserves the size of the input set

```
1 $d \leftarrow |\mathcal{W}|$
2 for $x \in \mathbb{F}_2^{n_i}$ do
3     Compute $\mathsf{D}_{\mathsf{H},x} \colon \mathbb{F}_2^{|\mathcal{W}|} \to \mathbb{R}$
4     Compute $\widehat{\mathsf{D}_{\mathsf{H},x}}$ from $\mathsf{D}_{\mathsf{H},x}$ using FFT        ▷ Complexity $|\mathcal{W}| \cdot 2^{|\mathcal{W}|}$
5     if $\max\{t \mid \forall \beta \in \mathbb{F}_2^{|\mathcal{W}|}, \mathrm{wt}(\beta) \leq t \colon \widehat{\mathsf{D}_{\mathsf{H},x}}(\beta) = 0\} < d$ then
6         $d \leftarrow \max\{t \mid \forall \beta \in \mathbb{F}_2^{|\mathcal{W}|}, \mathrm{wt}(\beta) \leq t \colon \widehat{\mathsf{D}_{\mathsf{H},x}}(\beta) = 0\}$
7 return $d$
```

3.1 Preliminaries

To facilitate the efficient representation and manipulation of Boolean functions and probability distributions, we revisit the fundamental concepts and principles underlying BDDs and MTBDDs.

Binary Decision Diagrams. BDDs are a fundamental data structure in computer science to represent Boolean functions [1,12]. In general, a BDD is a concise and unique (i.e., canonical) graph-based representations of a Boolean function.

Definition 6. *A Reduced Ordered Binary Decision Diagram (ROBDD)*[5] *is a pair (π, G), where π denotes a fixed ordering over an variable set $\mathcal{X} := \{x_1, x_2, \ldots, x_n\}$ and $\mathsf{G} = (\mathcal{V}, \mathcal{E})$ represents a rooted DAG with the following properties:*

1. *There is a single root and each vertex $v \in \mathcal{V}$ is either a non-terminal node or a terminal node. A terminal node is taking a value in the value set $\mathbb{B} := \{0,1\}$. There is only one terminal node labeled 0 and only one terminal node labeled 1 (no duplicate terminal nodes).*
2. *Each non-terminal node v is labeled with a variable in $x_i \in \mathcal{X}$, denoted as $\mathsf{var}(v)$, and has exactly two child nodes in $\mathcal{V}$ which are denoted as $\mathsf{then}(v)$ and $\mathsf{else}(v)$.*
3. *For each path from the root node to a terminal node, the variables in $\mathcal{X}$ are encountered at most once and in the same order defined by the ordering π that is a bijection $\pi : \{1, 2, ..., n\} \to \mathcal{X}$.*
4. *There is no node $v \in \mathcal{V}$ such that $\mathsf{then}(v) = \mathsf{else}(v)$ and, if $v, v' \in \mathcal{V}$ with $(\mathsf{var}(v), \mathsf{then}(v), \mathsf{else}(v)) = (\mathsf{var}(v'), \mathsf{then}(v'), \mathsf{else}(v'))$, we have $v = v'$.*

Each BDD with single root $v \in \mathcal{V}$ recursively defines a Boolean function $\mathsf{f}_v \colon \mathbb{F}_2^n \to \mathbb{F}_2$ such that, if v is a terminal node $b \in \mathbb{B}$, then $\mathsf{f}_v = b$. Otherwise,

[5] For the sake of simplicity, we use the term BDD instead of ROBDD in the context of this paper.

if v is a non-terminal node and $\mathsf{var}(v) = x_i$, then f_v is defined by the Shannon decomposition $\mathsf{f}_v = x_i \cdot \mathsf{f}_{\mathsf{then}(v)} + \overline{x_i} \cdot \mathsf{f}_{\mathsf{else}(v)}$. Conversely, any Boolean function can be described by a BDD in this way.

Boolean operations over BDDs. For any BDD, the RESTRICT(f, x, b) operator returns the Shannon co-factor $\mathsf{f}|_{x=b}$ while the If-Then-Else operator ITE$(\mathsf{f}, \mathsf{g}, \mathsf{h})$ composes the functions $\mathsf{f}, \mathsf{g}, \mathsf{h}\colon \mathbb{F}_2^n \to \mathbb{F}_2$ such that

$$\mathrm{ITE}(\mathsf{f}, \mathsf{g}, \mathsf{h}) = \mathsf{f} \cdot \mathsf{g} + \bar{\mathsf{f}} \cdot \mathsf{h}.$$

The ITE operator can be extended to functions $\mathsf{f}, \mathsf{g}, \mathsf{h}$ with not necessarily equal input domains $\mathbb{F}_2^n$, $\mathbb{F}_2^{n'}$, and $\mathbb{F}_2^{n''}$, respectively, by expanding the function f (resp., g, h) to a function $\tilde{\mathsf{f}}\colon \mathbb{F}_2^{\max\{n,n',n''\}} \to \mathbb{F}_2$ defined by $\tilde{\mathsf{f}}(x_1, \ldots, x_n, \ldots, x_{\max\{n,n',n''\}}) = \mathsf{f}(x_1, \ldots, x_n)$ (defining $\tilde{\mathsf{g}}, \tilde{\mathsf{h}}$ respectively) and defining $\mathrm{ITE}(\mathsf{f}, \mathsf{g}, \mathsf{h}) \coloneqq \mathrm{ITE}(\tilde{\mathsf{f}}, \tilde{\mathsf{g}}, \tilde{\mathsf{h}})$.

Further, any function $\mathsf{f} = \mathsf{f}_{v_1} \star \mathsf{f}_{v_2}$, where $\star$ is an arbitrary binary Boolean operation, can be derived and composed recursively as:

$$\begin{aligned} \mathsf{f} &= x_i \cdot \mathsf{f}|_{x_i=1} + \overline{x_i} \cdot \mathsf{f}|_{x_i=0} \\ &= x_i \cdot (\mathsf{f}_{v_1} \star \mathsf{f}_{v_2})|_{x_i=1} + \overline{x_i} \cdot (\mathsf{f}_{v_1} \star \mathsf{f}_{v_2})|_{x_i=0} \\ &= x_i \cdot (\mathsf{f}_{v_1}|_{x_i=1} \star \mathsf{f}_{v_2}|_{x_i=1}) + \overline{x_i} \cdot (\mathsf{f}_{v_1}|_{x_i=0} \star \mathsf{f}_{v_2}|_{x_i=0}) \end{aligned} \tag{3}$$

Moreover, *existential quantification* with respect to a variable x is a common BDD operation that computes $\exists_x \mathsf{f} \coloneqq \mathsf{f}|_{x=1} + \mathsf{f}|_{x=0}$. If $x_1, \ldots, x_k$ are k variables, the existential quantification $\exists_{x_1,\ldots,x_k} \mathsf{f}$ is recursively defined as $\exists_{x_k} \exists_{x_1,\ldots,x_{k-1}} \mathsf{f}$ (note that this is independent of the ordering of the variables).

Multi-Terminal Binary Decision Diagrams. MTBDDs are an extension of BDDs to represent functions from a multi-dimensional Boolean domain to an arbitrary value set $\mathbb{D}$. Specifically, if $\mathbb{D} = \mathbb{B}$ then MTBDDs reduce to BDDs while otherwise they can represent arbitrary functions of type $\mathsf{f} : \mathbb{F}_2^n \to \mathbb{D}$.

Remark. In the remainder of this work, we specifically assume $\mathbb{D} = \mathbb{Z}$ and heavily rely on the *existential quantification* operation over MTBDDs. Please note, that in this case, the $+$ operation is an arithmetic addition rather than the logic OR operation.

3.2 Fundamental Implementation Assumptions

Before we explain the details of the security verification concept, we briefly summarize essential initial assumptions.

Circuit Representation. Due to the limitation of the circuit model to represent circuits in terms of DAGs, as outlined in Definition 1, it is necessary that any iterative or looped circuit is first transformed into a functionally equivalent circuit by unrolling and pipelining the clocked stages, which can then be easily represented by a DAG.

Input Distributions. For both considered information leakage models, we assume independent and identically distributed (i.i.d.) secrets $x \in \mathbb{F}_2^{n_i}$. Further, for both the initial sharing $r_0 \in \mathbb{F}_2^{(s-1)n_i}$ of secrets and the refreshing $r_1 \in \mathbb{F}_2^{n_r}$ of intermediate computations, we assume the application of i.i.d. random bits.[6]

Extension Function. Glitches are generally known as a major source of information leakage [36, 37] for hardware implementations. Since our verification is primarily focused on hardware-related verification of security properties, we opted for a glitch-extension according to [20] as part of the probing model. In particular, it is assumed that each probe has a worst case extension that additionally leaks all (stable) inputs necessary for the generation of the probed signal.

For the random probing model, however, hardware effects such as glitches have not yet been considered in related literature. We adhere to common practice and therefore only use the identity function id as an extension function within the random probing model, although an extension is theoretically possible (see [10]) and generally not inhibited by our methodology.

3.3 Glitch-Extended Threshold Probing Model

Verifying probing security necessitates the exact knowledge of the leakage function in accordance with Definition 3. Accordingly, this requires the knowledge of the encoding function Enc, the access function A_C, and potentially the extension function Ex.

Encoding function. The encoding function is used to achieve a valid and secure sharing of the inputs that are subsequently processed by the masked version of the circuit. In this regard, the type of encoding function is purely determined by the specific masking scheme used to generate the masked circuit.

In our implementation, we have chosen to primarily support Boolean masking due to its prevalent adoption in symmetric cryptography. Nevertheless, it is important to emphasize that our methodology and indistinguishability verification framework are theoretically capable of supporting a wide range of encoding functions, e.g., used for arithmetic or multiplicative masking schemes.

Glitch-extended access function. According to Section 2.1, the function $\mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{m \cdot s}$ of a masked circuit is computed by iteratively assigning each wire with the result of its driving logic gate. In particular, since each wire $w \in \mathcal{W}$ computes a Boolean function $\mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i} \times \mathbb{F}_2^{n_r} \mapsto \mathbb{F}_2$, this can be generated and stored efficiently with the aid of BDDs. More specifically,

[6] It should be noted that these assumptions are usually made in the literature and simplify our methodology. However, they are not strictly necessary, so that arbitrary probability distributions for inputs and randomness are also theoretically supported.

each input of the circuit is assigned with a new Boolean variable and the logic gates of the circuit are evaluated iteratively given the BDDs of their inputs. Then, given the set of all BDDs associated with a wire of the circuit, this directly provides the access function of the circuit that can be used to compute the function $\mathsf{H} = \mathsf{A}_\mathsf{C} \circ \mathsf{Enc}$.

For glitch-extended probing security verification, the limited view of an adversary is further captured by a subfunction $\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}$, with $\mathsf{Ex}(\mathcal{P})$ being the worst-case glitch-extension function as defined before. Hence, in selecting all subsets of wires $\mathcal{P} \subseteq \mathcal{W}$ with $|\mathcal{P}| \leq d$ and generation of the associated extension sets $\mathcal{P}' \subseteq \mathcal{W}$, we can select the corresponding BDDs of the probed wires, to easily express all glitch-extended adversarial views $\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}$.

Please note, as indicated before, that we have decided not to use the FFT here for two reasons: Firstly, the glitch-extension function is not bijective and size-preserving, and secondly, in practice often only small d (compared to the total number of possible probe locations) are considered, for which a naive generation of all possible functions is still feasible so that the application of the FFT shows hardly any advantages.

Leakage function. According to Lemma 1, knowledge of $|(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_x^{-1}(y)|$ (for all $x \in \mathbb{F}_2^{n_i}$ and all $y \in \mathbb{F}_2^{|\mathsf{Ex}(\mathcal{P})|}$) is sufficient to verify absence of information leakage for any adversarial observation $\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}$. For this, we introduce our generation methodology, which particularly relies on BDDs and MTBDDs, in the following paragraphs in more detail.

Definition 7 (Transition). *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 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* C *implementing a function* $\mathsf{F}\colon \mathbb{F}_2^{n_i \cdot s} \times \mathbb{F}_2^{n_r} \to \mathbb{F}_2^{n_o \cdot s}$*. Let* $\mathsf{H} \coloneqq \mathsf{A}_\mathsf{C} \circ \mathsf{Enc}$*, which is a function from* $\mathbb{F}_2^{n_i} \times \mathbb{F}_2^{(s-1)n_i + n_r}$ *to* $\mathbb{F}_2^{|\mathcal{W}|}$*. We define*

$$\begin{aligned} \mathcal{T}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex}(\mathcal{P}))}\colon \mathbb{F}_2^{s \cdot n_i + n_r + |\mathsf{Ex}(\mathcal{P})|} &\to \{0,1\} \\ (x,r,y) &\mapsto \begin{cases} 1 & \textit{if } \mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}(x,r) = y \\ 0 & \textit{else} \end{cases} \end{aligned}$$

to be the transition function for a tuple (x,r,y) *in the masked circuit* C.

Given this definition and knowledge of $\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}$, we can directly calculate the transition as follows:

$$\mathcal{T}_{(\mathsf{Enc},\mathsf{A}_\mathsf{C},\mathsf{Ex}(\mathcal{P}))}(x,r,y) = \prod_{i=0}^{|\mathsf{Ex}(\mathcal{P})|-1} \overline{(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})_i}(x,r) \oplus y_i)}$$

where $\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})_i}$ and y_i denote the i-th coordinate function of the adversarial view and the i-th bit of y, respectively.

Based on this, the frequencies $|(\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})})_x^{-1}(y)| = \exists_r \mathcal{T}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex}(\mathcal{P}))}(x,r,y)$ are generated by using the existential quantification operator for MTBDDs with respect to r. Note that this implicitly considers a uniform distribution for $|x|$ and $|r|$ (according to our basic implementation assumptions). However, accounting for non-uniform distributions requires additional scaling of the frequencies with respect to the different occurrences of x and r.

Eventually, indistinguishability of the adversarial observations, according to Definition 4, is implemented and checked with simple subtraction according to $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}$ (as defined in Eq.(1)) such that $\mathcal{L}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex})}(\mathcal{P}) = 0$ if $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})} = 0$ (for all such functions $\mathsf{D}_{\mathsf{Ex}(\mathcal{P})}$ defined by all $x \in \mathbb{F}_2^{n_i}$) and $\mathcal{L}_{(\mathsf{Enc},\mathsf{A_C},\mathsf{Ex})}(\mathcal{P}) = 1$ otherwise.

3.4 Standard Random Probing Model

Since we assume the identity as an extension function, i.e., $\mathsf{Ex} = \mathsf{id}$ and $\mathcal{P} = \mathcal{W}$ for verification in the standard random probing model, it is possible to efficiently use the FFT to calculate the random probing leakage probabilities. In the following, we therefore describe the efficient calculation of FFT for MTBDDs on the one hand and the straightforward extraction of the leakage probabilities on the other.

The Fast Fourier-Hadamard Transformation. For the FFT, we assume that the function $\mathsf{H}|_{\mathcal{W}} = \mathsf{A_C} \circ \mathsf{Enc}$, considering full access to the masked circuit C, was generated according to the description in the previous section and is provided as MTBDD. In this case, Algorithm 2 describes the exact procedure for generation of the FFT using two fundamental BDD and MTBDD operators RESTRICT and ITE.

Algorithm 2: Computing the Fast Fourier-Hadamard Transformation using the RESTRICT and ITE MTBDD operations with linear complexity in the number of variables.

```
1 $\widehat{\mathsf{D}_{\mathsf{H},x}} \leftarrow \mathsf{D}_{\mathsf{H},x}$
2 for $x_i \in x$ do
3     $\mathsf{D}_0 \leftarrow \widehat{\mathsf{D}_{\mathsf{H},x}}|_{x_i=0}$                          ▷ RESTRICT
4     $\mathsf{D}_1 \leftarrow \widehat{\mathsf{D}_{\mathsf{H},x}}|_{x_i=1}$                          ▷ RESTRICT
5     $\widehat{\mathsf{D}_{\mathsf{H},x}} \leftarrow \overline{x_i}(\mathsf{D}_0 + \mathsf{D}_1) + x_i(\mathsf{D}_0 - \mathsf{D}_1)$     ▷ ITE
6 return $\widehat{\mathsf{D}_{\mathsf{H},x}}$
```

The leakage function. Given $\widehat{\mathsf{D}_{\mathsf{H},x}}(\beta)$ in MTBDD representation, $\mathsf{g}_{\mathsf{H},x}$ can be retrieved rapidly in converting all non-zero terminal nodes to 1, i.e., $\mathsf{g}_{\mathsf{H},x}(\beta) = 0$ if and only if $\widehat{\mathsf{D}_{\mathsf{H},x}}(\beta) = 0$ and $\mathsf{g}_{\mathsf{H},x}(\beta) = 1$ otherwise. Then, $\bigvee_{\beta \in \mathrm{prec}(\mathsf{Ex}(\mathcal{P}))} \mathsf{g}_{\mathsf{H},x}(\beta)$ is derived according to Algorithm 3 (using the RESTRICT operator).

Algorithm 3: Expanding the FFT to the leakage function using RESTRICT MTBDD operator.

```
1 $\mathsf{g}_{\mathsf{H},x} \leftarrow \widehat{\mathsf{D}_{\mathsf{H},x}}$
2 for $\beta_i \in \beta$ do
3     $\mathsf{g}_0 \leftarrow \mathsf{g}_{\mathsf{H},x}(\beta)|_{\beta_i=0}$        ▷ RESTRICT
4     $\mathsf{g}_{\mathsf{H},x} \leftarrow \mathsf{g}_{\mathsf{H},x} \vee \mathsf{g}_0$
5 return $\mathsf{g}_{\mathsf{H},x}$
```

Eventually, computation of the random probing leakage probability requires a slightly modified SatCount operator, according to Algorithm 4.

Algorithm 4: Modified SATCOUNT BDD operator to compute the leakage probability for a leakage function f and a leakage probability p.

```
1 Function LeakageProbability(f,$p$):
2     if $\mathsf{f} = 0$ then
3         return 0
4     if $\mathsf{f} = 1$ then
5         return 1
6     Pick next $x_i \in x$
7     $p_0 \leftarrow$LeakageProbability($\mathsf{f}|_{x_i=0}, p$)        ▷ RESTRICT
8     $p_1 \leftarrow$LeakageProbability($\mathsf{f}|_{x_i=1}, p$)        ▷ RESTRICT
9     return $p_0 + p \cdot (p_1 - p_0)$
```

3.5 Additional Implementation Optimizations

To enable practical verification, especially for the targeted large-scale circuits, we discuss additional implementation optimizations in the following paragraphs that simplify and accelerate execution of our verification tool.

Problem-space partitioning. A classic approach in computer science for handling large problems and search spaces is to break the problem down into smaller sub-problems that can be examined independently of each other before the partial results are finally combined to form the overall result. In this context, we would like to discuss in particular the partitioning of the circuit in width and depth respectively spatial and temporal direction.

Space (circuit breadth). First, it can be observed that for independent modules and components that process independent parts of the secret, the adversarial observations are also independent and can only leak information about the processed part of the secret. Accordingly, these parts can be verified independently and only then the results can be combined. For this purpose, we have implemented a initial information flow analysis for partitioning the circuit with respect to the processed parts of the secret. The examination of smaller sub-components in particular can considerably accelerate the overall verification process without the results losing their significance and precision.

Time (circuit depth). Leveraging the fact that we only consider unrolled and pipelined circuits, we can also partition the verification effort in the temporal dimension of the circuit. Specifically, limiting the verification results to only cycle-accuracy, i.e., only considering probes placed within a single cycle respectively pipelining stage, we can partition the computational effort of generating the adversarial view $\mathsf{H}|_{\mathsf{Ex}(\mathcal{P})}$.

In fact, this approach is based on a the implementation of a function that allows generating the appropriate output distribution for a given input distribution and a corresponding transition function. Given this, we can compute the distribution of the adversarial view for each cycle individually in first generating the input distributions of each cycle iteratively as the output distribution of the previous cycle and then leveraging the same function to compute the adversarial view. This approach particularly limits the complexity of the intermediate transition functions in order to improve the evaluation performance. Note, that this decision is also based on the assumption that common practical leakage assessment methodologies (e.g. Test Vector Leakage Assessment (TVLA)) for hardware platforms also mostly combine information from single clock cycle to provide univariate test results.

Parallelism. In particular, the decomposition of the problem space into independent sub-problems favors the parallelization of processing, especially for modern multi-core processor architectures. Accordingly, the different partitions of the individual stages are processed in parallel on independent processor cores, depending on availability, so that the execution time of our verification can continue to be significantly accelerated.

Approximation. Another option for optimizing the tool runtime concerns the accuracy of the verification results. This is particularly interesting for the random probing model, as the combinatorial complexity increases exponentially in the circuit size due to the complete enumeration of all probe combinations, so that even the benefits of partitioning quickly reach their limits.

In this case, we suggest that both the maximum number of probes and the number of probe combinations taken into account can be limited by the user. However, this means that not all combinations are enumerated and considered. As a result, the tool no longer reports an exact leakage probability, but instead determines a lower and an upper bound for the leakage probability. In particular, it is conservatively assumed for the lower bound that all unconsidered combinations are secure, whereas the exact opposite is assumed for the upper bound, i.e.

that all unconsidered combinations are insecure (for more details, we refer the interested reader to [9]).

By varying the number and combination of samples, the runtime and memory requirements can be adjusted, but at the expense of the accuracy of the verification results.

4 Evaluation

In this section, we evaluate and discuss the performance of INDIANA implementing the verification strategies presented in Section 3. To this end, we verify and benchmark several large-scale cryptographic designs taken from the literature.

We focus our evaluation in particular on the following three aspects. First, we demonstrate that our new approach to verification in the glitch-extended (standard) probing model can also be applied to large-scale cryptographic circuits that cannot be analyzed with previous state-of-the-art tools. In particular, we analyze and verify different masking schemes applied to round-based SPN implementations, e.g., PRESENT, SKINNY and AES. Next, we present verification and performance results in the random probing model. To this end, we first focus on an empirical investigation of the parameter choice (i.e. the number of probes and samples) on the accuracy of the leakage probability bounds using an exemplary PRESENT implementation. Finally, we turn to a first-order masked AES implementation, as a case study of greatest practical relevance, and perform random probing checks. In particular, we compare the results with practical measurements that allow us to connect the theoretical models implemented in INDIANA with real implementations.

Verification Setup. All reported numbers regarding verification results are derived at a machine equipped with 128 GB RAM and an Intel Xeon E5-1660 Central Processing Unit (CPU) running at 3.2 GHz. The CPU has 16 cores that we fully utilize in case the corresponding tool supports multithreading.

4.1 Glitch-Extended Threshold Probing Verification Results

In this section, we present verification results of single-round implementations of common block ciphers and compare the performance of INDIANA to frameworks from the literature. More precisely, we compare INDIANA to the frameworks maskVerif [3] and VERICA [44]. Before presenting the results, we briefly discuss the reasoning behind this selection and introduce both tools in more detail.

Related Work. Over the last years, many computer-aided verification frameworks for analyzing protected hardware implementations have been presented. However, only a few are able to analyze full rounds of protected block ciphers. For example, we tried to execute our case studies with SILVER [32] which, however, is not able to process that many input variables. Other tools, e.g., ironMask, are limited in the circuit structures that can be parsed and processed, and therefore they are not able to analyze entire rounds. To this end, we only managed to perform our case studies with maskVerif and VERICA which we briefly discuss in the following.

maskVerif. In 2019, Barthe et *al.* presented a novel tool to automatically verify the security of masked implementations in the presence of physical defaults (e.g., glitches and transitions). For that, maskVerif [3] is designed to perform multivariate security analysis, i.e., the tool can analyze probe combinations with probes located at different points in time (e.g., across multiple clock cycles). However, it has been shown that the tool may report false negatives [32]. Indeed, it may falsely categorize a secure circuit as insecure, causing unnecessary design overhead to mitigate such flaws.

VERICA. Richter-Brockmann et *al.* presented the first tool for security verification in the presence of combined attacks, i.e., combining active information tampering with passive information leakage. The tool emerged from the consolidation of SILVER [32] and FIVER [45], a state-of-the-art automated verification tool for FIA. Therefore, VERICA employs BDDs as well for verification of security properties (both for SCA and FIA) [23, 45]. Hence, VERICA can also perform stand-alone security verification in the glitch-extended d-probing model.

Interestingly, VERICA has been implemented as a modular and extendable security verification framework that easily adds additional analysis and verification strategies to the existing tool structure. For this, we opted to integrate our verification concept into this framework to implement INDIANA for verification based on the indistinguishability of probability distributions.

Security Verification. Before we present the verification results in detail, we reason about the selection of target designs and describe how we generate them.

Selection of case studies. Table 2 lists all designs that we consider for our case study. The selected single-round designs cover a range of lightweight ciphers (i.e., PRESENT and SKINNY-64), slightly more complex ciphers (i.e., LED-64 and SKINNY-128), and AES as a widely deployed block cipher. Despite for AES, we consider three different protection schemes: designs protected by Threshold Implementation (TI) [40], designs protected by HPC1 gadgets [15], and designs protected by HPC2 gadgets [15]. For the gadget-based designs, we consider configurations protected against first-order and second-order attacks, i.e., $d = 1$ and $d = 2$, respectively.

For the PRESENT and LED implementations protected by TI, we use the same S-box implementation as in [47]. We implemented the remaining (masked) combinatorial logic manually. The TI designs of SKINNY have been designed and implemented by us. The S-boxes for all the HPC1 and HPC2 are generated by SAIREDA [21, 22]. Again, we implemented the missing (masked) combinatorial

Table 2: Comparison between our indistinguishability strategy and tools from the literature verifying single rounds of various cryptographic schemes. All experiments highlighted in green pass the verification for the corresponding security order. Experiments highlighted in red fail the verification and are reported as insecure for all security orders by the corresponding tool. By $\perp$, we denote experiments where the verification did not finish under 8 h. By $\times$, we denote that the computation of the corresponding tool runs out of memory during parsing or evaluation.

	Ord.	**Impl.**				
Design	d	comb.	mem.	**maskVerif**^{I} [3]	**VERICA**^{II} [44]	**OUR**^{II}
PRESENT (TI)	1	11 760	192	†	76.5 s	40.8 s
PRESENT (HPC1)	1	1 360	1 152	0.5 s	3.5 s	3.8 s
PRESENT (HPC1)	2	2 816	2 064	3.2 min	$\perp$	55.9 s
PRESENT (HPC2)	1	1 552	1 536	0.5 s^f	4.5 s	5.4 s
PRESENT (HPC2)	2	3 328	3 216	0.9 s^f	$\perp$	11.6 min
LED-64 (TI)	1	12 240	384	†	$\times$	69.3 s
LED-64 (HPC1)	1	1872	1152	0.3 s	$\times$	5.5 s
LED-64 (HPC1)	2	3 584	2 064	88.8 s	$\times$	93.9 s
LED-64 (HPC2)	1	2 064	1 536	0.6 s^f	$\times$	8.4 s
LED-64 (HPC2)	2	4 096	3 216	1.1 s^f	$\times$	25.1 min
SKINNY-64 (TI)	1	1 401	192	–*	$\perp$	1.1 s
SKINNY-64 (HPC1)	1	1 064	1 024	–*	9.7 s	2.3 s
SKINNY-64 (HPC1)	2	2 169	1 728	–*	$\perp$	9.1 s
SKINNY-64 (HPC2)	1	1 256	1 408	–*	8.8 s	3.5 s
SKINNY-64 (HPC2)	2	2 873	2 880	–*	$\perp$	39.4 s
SKINNY-128 (TI)	1	1 641	384	–*	$\perp$	2.4 s
SKINNY-128 (HPC1)	1	2 120	3 072	–*	$\perp$	14.0 s
SKINNY-128 (HPC1)	2	4 329	4 992	–*	$\perp$	52.8 s
SKINNY-128 (HPC2)	1	2 504	3 840	–*	$\perp$	21.5 s
SKINNY-128 (HPC2)	2	5 737	7 296	–*	$\perp$	57.4 min
AES-128 [28]	1	11 312	3 584	4.4 min	$\times$	45.0 min
AES-128 [28]	2	24 184	6 240	$\perp$	$\times$	$\times$
AES-128 (HPC1)	1	11 584	9 088	97.7 s	$\times$	$\times$
AES-128 (HPC2)	1	13 312	12 544	10.5 s^f	$\times$	$\times$

^{I} Single-core analysis. ^{II} Multi-threading (16 cores). † Stack overflow. ^{f} False-negative.
* Round constants cannot be processed by maskVerif.

logic manually. The remaining two AES-128 designs were presented in [28] and are based on the Domain-Oriented Masking (DOM) principle.

The implementation costs for the selected case studies are reported in Table 2. We divide the costs into the number of combinatorial gates and the number of memory gates, i.e., registers. Again, these numbers perfectly show the different complexities of the selected designs.

Verification Results. We start our evaluation with the protected implementations of PRESENT. As shown in Table 2, INDIANA is able to verify the security of all five protection schemes. While almost all verifications can be performed in under one minute, the verification of the second-order HPC2 design takes 11.6 min. VERICA is able to analyze the first-order implementations in a similar time range but fails to finish the verification of the second-order designs in under 8 h. For the HPC1 designs, maskVerif is slightly slower for the second-order protected design than INDIANA. However, maskVerif performs a bivariate analysis while INDIANA is limited to cycle-accurate verifications. We were not able to perform the verification of the TI design due to a stack overflow. As already mentioned in the introduction of this section, maskVerif may report false negatives. This also happens when verifying the HPC2 designs. We assume that this occurs due to the computation $a_0 \cdot r + \bar{a_0} \cdot [b_1 + r]$ that is performed inside a HPC2 gadget which is also referred as the *masked shares multiplication trick*.

Similar results can be observed when analyzing the LED designs. However, VERICA is not able to verify any of these designs because it runs out of memory on our machine.

Next, we verify a number of SKINNY implementations. INDIANA is capable of analyzing all designs while the verification of the SKINNY-128 implementation protected by HPC2 gadgets is the slowest one taking 57.4 min. VERICA only reports results for the first-order HPC designs of SKINNY-64 while performing slightly worse than INDIANA. Eventually, maskVerif is not able to process the SKINNY designs since the tool cannot process round constants.

In our last case study, we perform verifications on AES designs. While VERICA runs out of memory for all designs, INDIANA reports results for the first-order protected implementation from [28]. The verification takes 45 min. Additionally to the first-order design from [28], maskVerif is also capable of analyzing the first-order HPC implementations. As above, it reports false negative results for the HPC2 design.

4.2 Standard Random Probing Verification Results

In this section, we will now focus on the verification and evaluation of different case studies in the standard random probing model in computing cycle-accurate random probing leakage probabilities. For the sake of clarity, we will focus on two case studies as examples. However, we would like to note that INDIANA is also able to analyze and verify all designs listed in Table 2 cycle-accurately in the standard random probing model.

The Leakage Probability. For all our case studies and experiments, we consider a hypothetical leakage probability $p = 10^{-2}$. However, we would like to note that this choice is not based on practical considerations, but is merely for illustrative purposes, i.e., to provide ranges for the upper and lower limits bounds are visible and displayable. In particular, we would like to emphasize that the actual choice of p has no influence on the evaluation and verification performance, as this is determined solely by the generation of the leakage function, while the evaluation for different p has the same computational effort.

Case Study I - PRESENT Round Function. Our first case study is therefore dedicated to a first-order HPC2-masked round implementation for the PRESENT cipher, where 16 masked S-boxes are instantiated in parallel (followed by a share-wise bit permutation layer). Moreover, each S-box is composed of PINI-secure HPC2 multiplication gadgets and is evaluated within four clock cycles. Notably, since each of the S-boxes processes an individual part of the secret (i.e., the unshared round input), the verification effort can be divided in *space* (16 parallel instances) and *time* (four clock cycles).

Table 3: This table presents verification results for a parameter sweep for a first-order protected implementation of PRESENT (single round) in the random probing model. The table reports the verification results cycle accurate for different numbers of probes and samples that are randomly selected by INDIANA. The number of positions corresponds to the total number of wires that is available for probing. By $\perp$, we identify experiments that did not finish in under 8 h.

			Samples				
Cycle	**Positions**	**Probes**	16×10	16×20	16×30	16×40	16×50
		∞	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×10	0.012/0.026	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×20	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×30	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
1	16×15	16×40	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×50	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×60	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×70	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		16×80	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013	0.013/0.013
		∞	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019
		16×10	0.005/0.908	0.009/0.757	0.011/0.632	0.014/0.565	0.015/0.491
		16×20	0.014/0.559	0.017/0.274	0.019/0.158	0.019/0.111	0.019/0.086
		16×30	0.018/0.168	0.019/0.050	0.019/0.030	0.019/0.024	0.019/0.022
2	16×58	16×40	0.019/0.031	0.019/0.020	0.019/0.019	0.019/0.019	0.019/0.019
		16×50	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019	$\perp$
		16×60	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019
		16×70	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019
		16×80	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019	0.019/0.019
		∞	0.079/0.079	0.079/0.079	0.079/0.079	0.079/0.079	0.079/0.079
		16×10	0.012/0.995	0.016/0.965	0.027/0.932	0.033/0.904	0.039/0.880
		16×20	0.034/0.918	0.055/0.775	0.064/0.654	0.069/0.556	0.073/0.480
		16×30	0.059/0.717	0.071/0.447	0.076/0.296	0.078/0.231	0.078/0.191
3	16×82	16×40	0.073/0.400	0.078/0.182	0.078/0.126	0.078/0.103	0.078/0.093
		16×50	0.078/0.171	0.078/0.092	0.079/0.082	0.079/0.080	$\perp$
		16×60	0.078/0.091	0.079/0.079	0.079/0.079	$\perp$	$\perp$
		16×70	0.079/0.079	0.079/0.079	$\perp$	$\perp$	$\perp$
		16×80	0.079/0.079	0.079/0.079	0.079/0.079	0.079/0.079	0.079/0.079
		∞	0.063/0.063	0.063/0.063	0.063/0.063	0.063/0.063	0.063/0.063
		16×10	0.020/0.834	0.032/0.625	0.042/0.515	0.048/0.434	0.052/0.372
		16×20	0.048/0.377	0.060/0.188	0.062/0.122	0.063/0.099	0.063/0.087
		16×30	0.061/0.120	0.063/0.069	0.063/0.065	0.063/0.064	0.063/0.063
4	16×52	16×40	0.063/0.064	0.063/0.063	0.063/0.063	0.063/0.063	0.063/0.063
		16×50	0.063/0.063	0.063/0.063	0.063/0.063	0.063/0.063	$\perp$
		16×60	0.063/0.063	0.063/0.063	0.063/0.063	$\perp$	$\perp$
		16×70	0.063/0.063	0.063/0.063	$\perp$	$\perp$	$\perp$
		16×80	0.063/0.063	0.063/0.063	0.063/0.063	0.063/0.063	0.063/0.063

Verification Results. In particular, we opted for this case study as we were able to compute exact leakage probabilities for all partitions and cycles of the

Table 4: This table reports the performance numbers of INDIANA for the verification results presented in Table 3. The performance numbers correspond to the verification time that is required to analyze all four cycles of the target design. By $\perp$, we identify experiments that did not finish in under 8 h.

	Samples				
Probes	16×10	16×20	16×30	16×40	16×50
∞	23.0 min	23.0 min	23.1 min	23.0 min	23.1 min
10	6.8 s	6.9 s	7.1 s	7.2 s	7.3 s
20	7.7 s	10.7 s	25.9 s	31.5 s	26.0 s
30	20.4 s	34.6 s	52.4 s	1.1 min	1.4 min
40	51.6 s	1.7 min	2.8 min	5.9 min	11.2 min
50	3.3 min	8.2 min	19.2 min	34.9 min	$\perp$
60	18.3 min	35.5 min	1.3 h	$\perp$	$\perp$
70	49.6 min	2.2 h	$\perp$	$\perp$	$\perp$
80	1.7 h	1.2 h	1.5 h	2.3 h	1.6 h

design, as indicated by all lines in Table 3 marked with ∞ for the number of probes. Based on this, we further opted to investigate various parameters for the maximum number of probes and combinations (denoted as samples) with regard to the accuracy of the leakage probability bounds as part of this experiment. In particular, we carried out various experiments that increased the maximum number of probes and samples in steps of ten and tested all their combinations (with a maximum of 80 probes and 50 samples allowed).

In this context, we would like to highlight and discuss some interesting observations in relation to the results provided in Table 3 and Table 4. However, due to the empirical and incomplete nature of this experiment, we must interpret the results with caution and refrain from making general statements.

Increasing the number of probes. As expected, increasing the number of probes ensures that larger parts of the search space can be covered, which provides more accurate results, especially with regard to the upper bound. It is noteworthy that the lower bound is already quite accurate for most experiments, indicating that the most influential key leakage occurs mainly for smaller combinations of probes.

Increasing the number of samples. Similarly, a larger number of randomly selected samples provides more accurate bounds, although the maximum number of 50 samples was hardly sufficient for any experiment to generate accurate results.

Verification timeouts. Although we were able to generate the exact leak probabilities for all cycles within 23 minutes, we observed timeouts after 8 hours in some experiments. This may seem surprising at first, but can be explained by the way the CUDD[7] BDD library works. In fact, the library heavily relies on caching BDDs calculations. This significantly improves performance, especially when calculations are performed repeatedly, i.e., when large parts of the sample combinations overlap. However, if the randomly selected samples are mostly disjoint, this leads to the eviction of cached intermediate

[7] `https://github.com/ivmai/cudd`

Table 5: This table presents verification results for a first-order protected implementation of AES (single round) in the random probing model. The table reports the verification results cycle accurate for combinations of at most 2 probes that are exhaustively generated by INDIANA. The number of positions corresponds to the total number of wires that is available for probing.

Cycle	Positions	Probes	Samples	Leakage	Total Elapsed Time
1	16×72	16×2	16×2556	0.056/0.458	1.20 min
2	16×138	16×2	16×9453	0.785/0.966	6.25 min
3	16×72	16×2	16×2556	0.099/0.472	39.33 min
4	16×52	16×2	16×1326	0.145/0.296	39.43 min
5	16×52	16×2	16×1326	0.034/0.236	39.53 min
6	16×92	16×2	16×4186	0.406/0.738	39.79 min
7	16×304	16×2	16×46056	0.992/0.999	3.33 h
8	16×102	16×2	16×5151	0.149/0.767	3.58 h
9	4×324	16×2	4×52326	0.051/0.981	3.76 h

results, which ultimately results in slower performance when more samples are considered.

Case Study II - AES Round Function. Our second case study is dedicated to the standardized and widely used AES cipher to demonstrate both the power and practical relevance of our approach. In particular, we focus on a first-order DOM-based AES round implementation (i.e., including key addition, 16 parallel S-box instances, and four parallel MixColumn operations) and provide cycle-accurate ranges for the leakage probabilities in the standard random probing model.

Verification Results. Table 5 lists the probes, samples, lower and upper bounds, and total elapsed verification time for the individual clock cycles of the AES round (assuming a leakage probability $p = 10^{-2}$). In particular, due to the complexity of the design (especially for the non-linear stages), we performed our verification with a maximum number of two probes but instead considered all possible combinations (samples). Noting that the design under test is only first-order protected, we assume that the key leakage (i.e., the main contributing failing probe combinations) will be observable with only two probes. However, this still should provide good estimates for the lower bounds while the upper bounds may lack in accuracy due to the large number of unconsidered combinations (as confirmed by the results in Table 5). Most importantly, the overall verification takes less than 4 hours which is mostly spend in the non-linear layers of cycle 3 and 7.

Case Study III - AES S-Box. For our last case study, we extracted a single S-box instance from the first-order DOM-based AES round function to perform an isolated verification in the standard random probing model alongside empirical TVLA. In particular, the verification results expectedly match with the previous case study. Still, mainly due to the limited capacity of our target Field-Programmable Gate Arrays (FPGAs) platform and in order to reduce the measurement noise for the practical evaluations, we provide this third case study considering an isolated AES S-box instance.

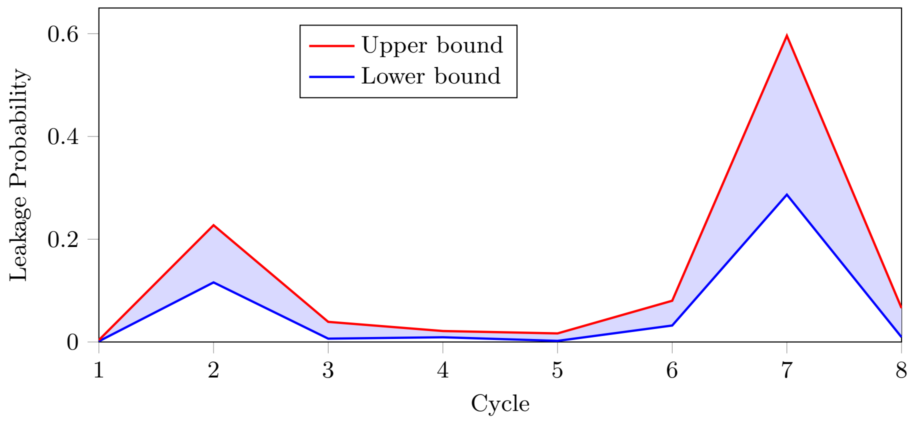
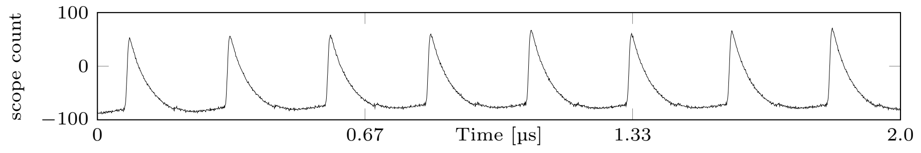


Fig. 1: Verification results of a first-order protected AES implementation (single S-box) in the standard random probing model. The number of probes has been fixed to two while the probe positions have been considered exhaustively.

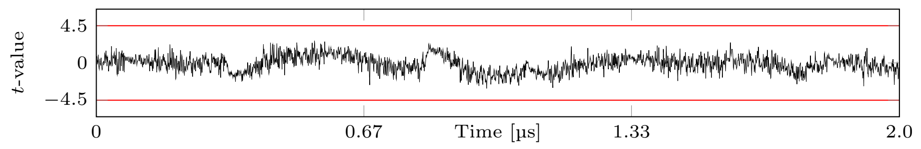
Verification Results. Similar to the previous case study, we limited the maximum number of samples to two, but took full account of all probe combinations. The results for both the upper bound (red) and the lower bound (blue) are shown in Figure 1.

Practical Evaluation. Additionally to the computer-aided verification results, we perform practical measurements for the same protected AES S-box. Therefore, we synthesize the S-box for the Sakura-G side-channel evaluation board that is equipped with a Xilinx Spartan 6 FPGA. The required fresh randomness for the AES S-box is provided by a KECCAK core instantiated as Pseudo-Random Number Generator (PRNG). In order to collect clean power traces, we set the clock frequency to 4 MHz and measure the power consumption indirectly via a 1 Ω shunt resistor placed in the supply path of the FPGA. The voltage signal is amplified by a ZFL-2000GH+ Low-Noise Amplifier (LNA) configured with a 19 dB gain and digitalized by a Spectrum M4 oscilloscope (8 bit resolution) with a sampling rate of 2.5 GS/s.

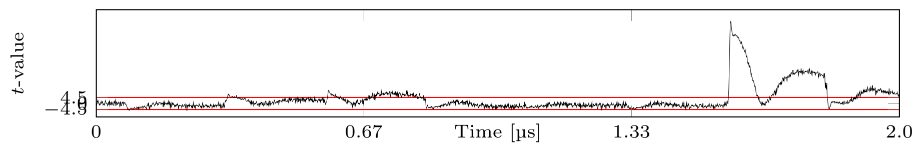
The security analysis is performed based on the TVLA methodology originally presented in [26]. Based on Welsh's t-test, we analyze the first two statistical moments with the *fixed vs. random* strategy as described in [46]. Here,



(a) Sample trace.



(b) First-order t-test results.



(c) Second-order t-test results.

Fig. 2: Measurement results for a first-order protected AES S-box (200 Million traces).

the absolute value of the t-test is commonly compared to a threshold of 4.5 corresponding to a confidence of 0.99999 rejecting the null hypothesis. Informally speaking, if we do not identify any point in time that exceeds this threshold, we consider the implementation to be secure. In contrast, if we identify peaks in the test results of the analyzed statistical moments, the power consumption is not independent of the processed input data such that we cannot conclude that the implementation is secure.

Figure 2 shows the measurement results of the protected AES S-box. More precisely, Figure 2a exemplary shows a power trace acquired with the described setup. The eight clock cycles of the S-box can clearly be seen. In total, we collect 200 million power traces (roughly 100 million traces with random input data and 100 million traces with fixed input data). Based on these data, we compute the first-order t-test which is shown in Figure 2b. As expected, we cannot identify any leakage. However, when performing a second-order t-test shown in Figure 2c, we clearly see some peaks indicating the existence of leakage. Particularly, we identify a huge peak in the seventh clock cycle and two smaller peaks in the second and third clock cycle.

Coherency between Theory and Practice. An interesting observation is that the verification and evaluation results seem to coincide. In both experiments, we have increased probabilities of detecting leakage in clock cycles 2 and 7, with the latter being the most signification position for leakage. Additionally, we have limited the experiments to second-order analyses in both cases.

Nevertheless, the results should be considered with caution, as no generally valid conclusions can be drawn from a single experiment. Instead, these observations provide further directions for interesting future work in order to determine and examine the coherency between the theoretical security models and the practical implementations in an in-depth and systematic analysis.

5 Conclusion

In this work, we present INDIANA as a comprehensive verification tool for hardware masking offering comprehensive verification in the well-established glitch-extended probing model. Additionally, INDIANA provides cycle-accurate estimations for the leakage probabilities in the standard random probing model even for large-scale circuits, e.g., full Substitution-Permutation Network (SPN) ciphers including PRESENT and AES. All this is only possible thanks to our novel and partitionable probing distinguisher and FFT application that enables rapid verification in both models and significantly outperforms state-of-the-art methods based on statistical independence. This is practically demonstrated in providing a comprehensive set of benchmarks and comparisons to state-of-the-art probing security tools.

Acknowledgements

The work described in this paper has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and 510964147 (CAVE). Moreover, this work has been funded in parts by the ERC project 101097056 (SYMTRUST), by the European Commission under the grant agreement number 101070374 (CONVOLVE), and by the German Federal Ministry of Education and Research (BMBF) through the projects VE-HEP (16KIS1345) and 6GEM (16KISK038).

References

1. Akers, S.B.: Binary Decision Diagrams. IEEE Trans. Computers **27**(6), 509–516 (1978). `https://doi.org/10.1109/TC.1978.1675141`, `https://doi.org/10.1109/TC.1978.1675141`
2. Arribas, V., Nikova, S., Rijmen, V.: VerMI: Verification Tool for Masked Implementations. In: ICECS. pp. 381–384. IEEE (2018)
3. 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: 24th European Symposium on Research in Computer Security, Part I. Lecture Notes in Computer Science, vol. 11735, pp. 300–318. Springer, Heidelberg, Germany,

Luxembourg (Sep 23–27, 2019). https://doi.org/10.1007/978-3-030-29959-0_15

4. 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.) Advances in Cryptology – EUROCRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 457–485. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46800-5_18
5. 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: 23rd Conference on Computer and Communications Security. pp. 116–129. ACM Press, Vienna, Austria (Oct 24–28, 2016). https://doi.org/10.1145/2976749.2978427
6. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.X., Strub, P.Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 535–566. Springer, Heidelberg, Germany, Paris, France (Apr 30 – May 4, 2017). https://doi.org/10.1007/978-3-319-56620-7_19
7. Barthe, G., Gourjon, M., Grégoire, B., Orlt, M., Paglialonga, C., Porth, L.: Masking in fine-grained leakage models: Construction, implementation and verification. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 189–228 (2021). https://doi.org/10.46586/tches.v2021.i2.189-228, https://tches.iacr.org/index.php/TCHES/article/view/8792
8. Battistello, A., Coron, J.S., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2016. Lecture Notes in Computer Science, vol. 9813, pp. 23–39. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–19, 2016). https://doi.org/10.1007/978-3-662-53140-2_2
9. 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.) Advances in Cryptology – CRYPTO 2020, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 339–368. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56784-2_12
10. 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, San Francisco, CA, USA (May 22–26, 2022). https://doi.org/10.1109/SP46214.2022.9833600
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.) Advances in Cryptology – EUROCRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 321–353. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78375-8_11
12. Bryant, R.E.: Graph-Based Algorithms for Boolean Function Manipulation. IEEE Trans. Computers **35**(8), 677–691 (1986). https://doi.org/10.1109/TC.1986.1676819, https://doi.org/10.1109/TC.1986.1676819

13. Carlet, C.: Boolean Functions for Cryptography and Coding Theory. Cambridge University Press, Cambridge (2021)
14. Cassiers, G., Faust, S., Orlt, M., Standaert, F.X.: Towards tight random probing security. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 185–214. Springer, Heidelberg, Germany, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84252-9_7
15. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.: Hardware Private Circuits: From Trivial Composition to Full Verification. IEEE Trans. Computers **70**(10), 1677–1690 (2021). https://doi.org/10.1109/TC.2020.3022979, https://doi.org/10.1109/TC.2020.3022979
16. Cassiers, G., Standaert, F.: Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. IEEE Trans. Inf. Forensics Secur. **15**, 2542–2555 (2020). https://doi.org/10.1109/TIFS.2020.2971153, https://doi.org/10.1109/TIFS.2020.2971153
17. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener [48], pp. 398–412. https://doi.org/10.1007/3-540-48405-1_26
18. De Meyer, L., Bilgin, B., Reparaz, O.: Consolidating security notions in hardware masking. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(3), 119–147 (2019). https://doi.org/10.13154/tches.v2019.i3.119-147, https://tches.iacr.org/index.php/TCHES/article/view/8291
19. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 423–440. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014). https://doi.org/10.1007/978-3-642-55220-5_24
20. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.X.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 89–120 (2018). https://doi.org/10.13154/tches.v2018.i3.89-120, https://tches.iacr.org/index.php/TCHES/article/view/7270
21. Feldtkeller, J.: Saireda (2022), https://github.com/Chair-for-Security-Engineering/SAIREDA
22. Feldtkeller, J., Knichel, D., Sasdrich, P., Moradi, A., Güneysu, T.: Randomness optimization for gadget compositions in higher-order masking. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(4), 188–227 (2022). https://doi.org/10.46586/tches.v2022.i4.188-227
23. Feldtkeller, J., Richter-Brockmann, J., Sasdrich, P., Güneysu, T.: CINI MINIS: Domain isolation for fault and combined security. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022: 29th Conference on Computer and Communications Security. pp. 1023–1036. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). https://doi.org/10.1145/3548606.3560614
24. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Çetin Kaya., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2001. Lecture Notes in Computer Science, vol. 2162, pp. 251–261. Springer, Heidelberg, Germany, Paris, France (May 14–16, 2001). https://doi.org/10.1007/3-540-44709-1_21
25. 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., Green-

stadt, R. (eds.) USENIX Security 2021: 30th USENIX Security Symposium. pp. 1469–1468. USENIX Association (Aug 11–13, 2021)

26. Gilbert Goodwill, B.J., Jaffe, J., Rohatgi, P., et al.: A testing methodology for side-channel resistance validation. In: NIST non-invasive attack testing workshop. vol. 7, pp. 115–136 (2011)
27. Groß, H., Mangard, S.: A unified masking approach. Journal of Cryptographic Engineering **8**(2), 109–124 (Jun 2018). `https://doi.org/10.1007/s13389-018-0184-y`
28. Groß, H., Mangard, S., Korak, T.: Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In: Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016. p. 3. ACM (2016). `https://doi.org/10.1145/2996366.2996426`
29. Groß, H., Mangard, S., Korak, T.: An efficient side-channel protected AES implementation with arbitrary protection order. In: Handschuh, H. (ed.) Topics in Cryptology – CT-RSA 2017. Lecture Notes in Computer Science, vol. 10159, pp. 95–112. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 14–17, 2017). `https://doi.org/10.1007/978-3-319-52153-4_6`
30. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003). `https://doi.org/10.1007/978-3-540-45146-4_27`
31. Knichel, D., Moradi, A.: Composable gadgets with reused fresh masks first-order probing-secure hardware circuits with only 6 fresh masks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(3), 114–140 (2022). `https://doi.org/10.46586/tches.v2022.i3.114-140`
32. Knichel, D., Sasdrich, P., Moradi, A.: SILVER - statistical independence and leakage verification. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 787–816. Springer, Heidelberg, Germany, Daejeon, South Korea (Dec 7–11, 2020). `https://doi.org/10.1007/978-3-030-64837-4_26`
33. Knichel, D., Sasdrich, P., Moradi, A.: Generic hardware private circuits towards automated generation of composable secure gadgets. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(1), 323–344 (2022). `https://doi.org/10.46586/tches.v2022.i1.323-344`
34. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO'96. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 1996). `https://doi.org/10.1007/3-540-68697-5_9`
35. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener [48], pp. 388–397. `https://doi.org/10.1007/3-540-48405-1_25`
36. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) Topics in Cryptology – CT-RSA 2005. Lecture Notes in Computer Science, vol. 3376, pp. 351–365. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 14–18, 2005). `https://doi.org/10.1007/978-3-540-30574-3_24`
37. Mangard, S., Schramm, K.: Pinpointing the side-channel leakage of masked AES hardware implementations. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2006. Lecture Notes in Computer Science,

vol. 4249, pp. 76–90. Springer, Heidelberg, Germany, Yokohama, Japan (Oct 10–13, 2006). https://doi.org/10.1007/11894063_7

38. Moos, T., Moradi, A., Schneider, T., Standaert, F.X.: Glitch-resistant masking revisited. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(2), 256–292 (2019). https://doi.org/10.13154/tches.v2019.i2.256-292, https://tches.iacr.org/index.php/TCHES/article/view/7392
39. Müller, N., Moradi, A.: PROLEAD A probing-based hardware leakage detection tool. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(4), 311–348 (2022). https://doi.org/10.46586/tches.v2022.i4.311-348
40. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4307, pp. 529–545. Springer (2006). https://doi.org/10.1007/11935308_38, https://doi.org/10.1007/11935308_38
41. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. Journal of Cryptology **24**(2), 292–321 (Apr 2011). https://doi.org/10.1007/s00145-010-9085-7
42. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 142–159. Springer, Heidelberg, Germany, Athens, Greece (May 26–30, 2013). https://doi.org/10.1007/978-3-642-38348-9_9
43. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 764–783. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). https://doi.org/10.1007/978-3-662-47989-6_37
44. Richter-Brockmann, J., Feldtkeller, J., Sasdrich, P., Güneysu, T.: VERICA - verification of combined attacks automated formal verification of security against simultaneous information leakage and tampering. IACR Transactions on Cryptographic Hardware and Embedded Systems **2022**(4), 255–284 (2022). https://doi.org/10.46586/tches.v2022.i4.255-284
45. Richter-Brockmann, J., Shahmirzadi, A.R., Sasdrich, P., Moradi, A., Güneysu, T.: FIVER - robust verification of countermeasures against fault injections. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 447–473 (2021). https://doi.org/10.46586/tches.v2021.i4.447-473, https://tches.iacr.org/index.php/TCHES/article/view/9072
46. Schneider, T., Moradi, A.: Leakage assessment methodology - extended version. Journal of Cryptographic Engineering **6**(2), 85–99 (Jun 2016). https://doi.org/10.1007/s13389-016-0120-y
47. Schneider, T., Moradi, A., Güneysu, T.: ParTI – towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 302–332. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016). https://doi.org/10.1007/978-3-662-53008-5_11
48. Wiener, M.J. (ed.): Advances in Cryptology – CRYPTO'99, Lecture Notes in Computer Science, vol. 1666. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999)