Formal Definition and Verification for Combined Random Fault and Random Probing Security

Sonia Belaïd[1], Jakob Feldtkeller[2], Tim Güneysu[2,3], Anna Guinet[2], Jan Richter-Brockmann[2], Matthieu Rivain[1], Pascal Sasdrich[2], and Abdul Rahman Taleb[1]

[1] CryptoExperts, Paris, France
{sonia.belaid, matthieu.rivain}@cryptoexperts.com
taleb.abdulrahman1@gmail.com
[2] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
{jakob.feldtkeller, tim.gueneysu, anna.guinet, jan.richter-brockmann, pascal.sasdrich}@rub.de
[3] DFKI, Bremen, Germany

Abstract. In our highly digitalized world, an adversary is not constrained to purely digital attacks but can monitor or influence the physical execution environment of a target computing device. Such side-channel or fault-injection analysis poses a significant threat to otherwise secure cryptographic implementations. Hence, it is important to consider additional adversarial capabilities when analyzing the security of cryptographic implementations besides the default black-box model. For side-channel analysis, this is done by providing the adversary with knowledge of some internal values, while for fault-injection analysis the capabilities of the adversaries include manipulation of some internal values.

In this work, we extend probabilistic security models for physical attacks, by introducing a *general random probing model* and a *general random fault model* to capture arbitrary leakage and fault distributions, as well as the combination of these models. Our aim is to enable a more accurate modeling of low-level physical effects. We then analyze important properties, such as the impact of adversarial knowledge on faults and compositions, and provide tool-based formal verification methods that allow the security assessment of design components. These methods are introduced as extension of previous tools VERICA and IronMask which are implemented, evaluated and compared.

Keywords: Physical Security · Random Probing Model · Random Fault Model · Combined Analysis · VERICA · IronMask

1 Introduction

State-of-the-art cryptographic schemes are usually analyzed in the common black-box model, which operates under the assumption that the internal values of the scheme are hidden and protected from the adversary. However, over the last 25 years, physical attacks, which exploit the physical realization and implementation of cryptographic algorithms, have questioned those assumptions. In particular, Side-Channel Analysis (SCA) exploits dependencies between processed values and physical execution characteristics such as timing behavior [32], instantaneous power consumption [33], or electromagnetic emanations [28], to conclude information about a secret. Similarly, Fault Injection Analysis (FIA) manipulates the physical execution environment to create a faulty intermediate state such that the resulting output gives a hint about the processed secret. Common fault injection methods include clock and voltage glitching [47,21], targeted electromagnetic (EM) pules [18,23], or focused laser beams [46]. Both attack vectors question a different assumption of the cryptographic black-box model: SCA that internal values are *confidential* and hidden from the adversary and FIA that the internal values have *integrity* and cannot be manipulated by the adversary. However, in a real attack scenario, the adversary is not restricted to performing only one of the two attack types. Indeed, first practical attack for Combined Analysis (CA), i.e., the combination of FIA and SCA, are emerging [3,15,40,41,42,43].

Constructing effective countermeasures against such physical attacks requires a deep understanding of attack properties and leakage behavior. To this end, the research community endeavors to theoretically model the leakage emanating from the victim device which is susceptible to exploitation by the adversary. In particular for SCA, in the famous d-probing model introduced by Ishai, Sahai, and Wagner [31], the leakage is modeled by the exact values of d internal variables of the adversary's choice. A victim device is then deemed d-probing secure if any such set of d intermediate variables is statistically independent on the processed secret. While this probing model facilitates security proofs, it sometimes fails to closely reflect the reality of embedded devices. For instance, it does not capture horizontal attacks [8], which exploit in particular repeated manipulation of variables within an execution. Consequently, the community is starting to focus on more realistic leakage models, like the ϵ-random probing model [31,2,10]. In the latter, the leakage is assumed to gather the exact value carried by each wire of the circuit with probability p. The security is then determined by the probability ϵ of the adversary to obtain a secret-dependent probe combination. This model tightly reduces to the security in the practical noisy leakage model [35,22], where each variable leaks a noisy function of its value. Nevertheless, the random probing model is still insufficient in modeling low-level physical effects because it assigns the same independent probability to each intermediate variable whereas the underlying noise is likely to be different and the independence assumption is not verified in practice.

For modeling of FIA the adversary is given the ability to manipulate a set of intermediate variables in a predefined way and then their impact on the system output is analyzed. Models from the literature often take inspiration from probing models for SCA. Specifically, the k-fault model [30] allows the adversary to manipulate up to k intermediate values, while the random fault model [19] defines a fault probability q and manipulates each internal values with probability q. In both models, the adversary wins when they get a faulty output that cannot be detected or corrected, whereas the random fault model determines the probability μ with which this happens. There again, while beneficial for a first approximation of physical security, the current models come with significant abstractions. In particular, the deterministic k-fault model cannot capture the imprecise nature of physical attacks due to the probabilistic fault behavior, while the random fault model fails to precisely model low-level physical characteristics of fault injection due to the same independent probability for each fault. Naturally, models for CA combine the capabilities of the individual models. All those models allow the analysis of certain types of secret-dependent leakages and enable pre-silicon evaluation.

Contribution. In this work, we provide a generalization of the probabilistic models for physical attacks by considering arbitrary leakage and fault distributions. In particular, we introduce a *general random probing model*, a *general random fault model*, and a *general random combined model* in Section 3, all of which, but especially our random combined model, are analyzed for interesting properties. Specifically, we analyze the impact of the adversarial knowledge on the injected faults on combined security. Since the resulting analysis complexity is very high, we introduce notions for compositions for all three models in Section 4. For our general random probing model, this is a straightforward extension of an existing notion for the ϵ-random probing model. To the best of our knowledge, for the other two models, this is the first attempt at composition in a probabilistic model. Within these security models, we investigate the formal verification of the proposed combined composition notions for gadgets. Specifically, we introduce two methods of tool-based analysis using VERICA [37] and IronMask [11]. We explain how to extend these tools to verify the proposed notions and present the implementation details in Section 5. In particular, our extension of VERICA can analyze arbitrary circuits with exact precision (up to some threshold) but is comparably slow. In contrast, our extension of IronMask is much faster but has restrictions in the type of designs it can handle and makes some approximations. We finally provide an extensive evaluation and comparison of the tools in Section 6.

2 Preliminaries

In the following, we give a short overview of the used notation and circuit model. Afterwards, we provide the basic concepts required to understand the contribution of this paper. Specifically, we introduce countermeasures for SCA and FIA, and discuss security proofs via simulation.

2.1 Notation

Throughout the paper, we use a sans-serif font for functions (e.g., f) and an upper-case calligraphic font for sets (e.g., $\mathcal{S}$). We designate $\overline{\mathcal{S}}$ the complement of a set $\mathcal{S}$, $|\mathcal{S}|$ its cardinality, and $\mathbb{D}\mathcal{S}$ a discrete probability distribution defined over the event set $\mathcal{S}$. Further, we name $\prod_i \mathbb{D}\mathcal{S}_i$ the joint probability distribution of independent distributions $\mathbb{D}\mathcal{S}_i$ where the joint probability is computed as the multiplication of the individual probabilities, i.e., $\mathsf{Pr}[a \cap b] = \mathsf{Pr}[a] \cdot \mathsf{Pr}[b]$, and with $\equiv$ the equality of distributions. To simplify the notation for n-times replication we denote with $\mathbb{V}_n = \{(0)^n, (1)^n\}$ the set that contains the zero vector $(0)^n$ and the one vector $(1)^n$ of size n. Other notations will be introduced throughout the paper where necessary.

2.2 Circuit Model

In this work, we model a circuit C as a *Direct Acyclic Graph (DAG)* $\mathcal{C} = \{\mathcal{G}, \mathcal{W}\}$, where vertices $g \in \mathcal{G}$ represent logical gates and edges $w \in \mathcal{W}$ represent wires carrying a Boolean value and connecting individual gates. We restrict the set of combinational gates to $\mathcal{G}_{\mathsf{c}} = \{\mathsf{inv}, \mathsf{and}, \mathsf{xor}, \mathsf{or}\}$ and the set of memory gates to $\mathcal{G}_{\mathsf{m}} = \{\mathsf{reg}\}$. Further, we define a set of input and output gates $\mathcal{G}_{\mathsf{io}} = \{\mathsf{in}, \mathsf{out}\}$, where in produces and out absorbs a Boolean value, and a set of probabilistic gates $\mathcal{G}_{\mathsf{rand}} = \{\mathsf{rand}\}$, where rand produces a uniform-random Boolean value. Finally, we define the set of constant gates $\mathcal{G}_{\mathsf{const}} = \{\mathsf{zero}, \mathsf{one}\}$, where each gate produces the respective Boolean value. Hence, each gate is from the set $\mathcal{G}_{\mathsf{all}} = \mathcal{G}_{\mathsf{c}} \cup \mathcal{G}_{\mathsf{m}} \cup \mathcal{G}_{\mathsf{io}} \cup \mathcal{G}_{\mathsf{rand}} \cup \mathcal{G}_{\mathsf{const}}$. Given this, each gate implements a deterministic or probabilistic Boolean function $\mathsf{f_g} : \mathbb{F}_2^h \rightarrow \mathbb{F}_2$, with $0 \leq h \leq 2$, of the respective functionality. Further, let $\mathcal{G}_{\mathsf{f}} = \{\mathbb{F}_2^h \rightarrow \mathbb{F}_2 \mid h \leq 2\}$ be the set of all possible Boolean functions for this purpose. With respect to the fan-out of wires, we consider two different scenarios depending on the analysis of hardware or software. For hardware (cf. Section 5.2), we create a copy of the wire for each gate that is connected, i.e., for a wire with fan-out n we create n copies. Here a gate output can be connected to any number of copies of the same wire. For software (cf. Section 5.3), we say that each gate can have at most one output and introduce a special copy operation, that outputs two times the input. Hence, to represent a wire with fan-out n we require $2n-1$ wires to construct a tree of copys.

2.3 Countermeasures

Masking. A popular countermeasure against SCA is *Boolean masking* [14,29], due to its sound formal foundation. The core idea is to split a secret $x \in \mathbb{F}_2$ into a vector $\langle x_0, \ldots, x_{s-1} \rangle \in \mathbb{F}_2^s$, with $x_i \in \mathbb{F}_2$, such that $x = \bigoplus_{i=0}^{s-1} x_i$ and each subset $\{x_i \mid i \in [0, s-1]\}$ with cardinality smaller than s is statistically independent of x. We refer to a component x_i as a *share* of x with *share index* i. Similarly, a circuit is transformed to a *masked circuit*, which operates over shares of the inputs. In this paper, we assume the initial encoding enc of inputs and the final decoding dec of outputs (generating shares or recreating the secret, respectively) not as part of the masked circuit.

Replication. A popular countermeasure against FIA is the replication of the circuit in combination with a majority function for error correction purposes. In particular, a value $x \in \mathbb{F}_2$ is replicated to a vector $\langle x_0, \ldots, x_{n-1} \rangle \in \mathbb{F}_2^n$ with $n = 2k + 1$, such that $\forall i, j \in [0, n-1] : x_i = x_j$. Then, up to k faults can be corrected with a majority function maj. Likewise, a circuit is replicated n times, where every replication operates on a unique set of value replications.

2.4 Security Proofs via Simulation

Security proofs in the context of SCA are often conducted based on *simulation*. For this, two worlds are introduced. The first world represents a real implementation, while the second world is made trivially secure by removing the secret that an adversary tries to learn. If an adversary is not able to distinguish between the two worlds then the view of the adversary is proven to be independent of the secret. The proof works by construction of a simulator that recreates the distribution of the observed values without access to any secret.

However, in this work, we do not require perfect simulation but allow the simulator to be wrong with a small probability. In particular, we require a simulator to create a distribution that is ϵ close to the observed distribution. We say that any two probability distributions $\mathbb{D}_1$ and $\mathbb{D}_2$ are ϵ-close ($\mathbb{D}_1 \approxeq_\epsilon \mathbb{D}_2$) if their statistical distance is upper-bounded by ϵ, i.e.,

$$\frac{1}{2} \sum_x |\mathsf{Pr}_{\mathbb{D}_1}[x] - \mathsf{Pr}_{\mathbb{D}_2}[x]| \leq \epsilon.$$

3 Security Model

We start our contribution by defining probabilistic security models for SCA, FIA, and CA. For this, we build on existing models but capture a more general attack scenario.

3.1 General Random Probing Security

Adversary Model. We introduce a new generalization of the random probing model [31,2,10] called *general random probing model*. Here, a probing adversary $\mathcal{A}_p$ can invoke a circuit C multiple times and on each invocation, the exact values of a random subset of wires in C are leaked to $\mathcal{A}_p$. We denote the leaking combination of wires, i.e., the subset of the wires of the circuit that is given to $\mathcal{A}_p$, with $\tilde{\mathcal{W}} \subseteq \mathcal{W}$. Let $\mathcal{W}^\infty = \{\tilde{\mathcal{W}} \subseteq \mathcal{W}\}$ be the set of all wire combinations in C and $\mathbb{D}\mathcal{W}^\infty$ an arbitrary discrete probability distribution defined over $\mathcal{W}^\infty$. Further, we define the following two functions to first select a random element from $\mathcal{W}^\infty$ and then determine the values carried by the selected wires for a given input:

$\mathtt{LeakingWires}(C, \mathbb{D}\mathcal{W}^\infty)$: The leaking-wire sampler selects for a given circuit C a wire combination $\tilde{\mathcal{W}}$ with probability $\mathsf{Pr}_{\mathbb{D}\mathcal{W}^\infty}[\tilde{\mathcal{W}}]$.

$\texttt{AssignWires}(C, \tilde{\mathcal{W}}, x)$: The assign-wire sampler takes a fixed input x for C and outputs the values assigned to the wires $w \in \tilde{\mathcal{W}}$ as a tuple in $\mathbb{F}_2^{|\tilde{\mathcal{W}}|}$. If C is probabilistic so is $\texttt{AssignWires}()$.

Then, the view of $\mathcal{A}_p$ is formally defined as the *random probing leakage* $\mathcal{L}_{\tilde{\mathcal{W}}}(C, x)$, which is given by the random experiment

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C, \mathbb{D}\mathcal{W}^{\infty}) \,,$$
$$\mathcal{L}_{\tilde{\mathcal{W}}}(C, x) \leftarrow \texttt{AssignWires}(C, \tilde{\mathcal{W}}, x) \,.$$

Security Definition. Intuitively, a circuit C is secure in our model if the view of $\mathcal{A}_p$ can be simulated with high probability without access to the secret, i.e., there exists a simulator Sim that recreates the distribution of the leaking wires $\tilde{\mathcal{W}}$ without knowledge of the secret, such that the failure probability of Sim is bounded by some (small) ϵ. A more formal definition is given in Definition 1. Throughout this work, we consider enc to be some encoding function, e.g., defined by Boolean masking (cf. Section 2.3).

Definition 1 (General Random Probing Security). *A circuit C is said to be* $(\mathbb{D}\mathcal{W}^{\infty}, \epsilon)$-random probing secure *with respect to an encoding* enc *if there exists a simulator* Sim *such that for all inputs x:*

$$\mathsf{Sim}(C, \tilde{\mathcal{W}}) \approxeq_{\epsilon} \mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathit{enc}(x)) \,.$$

The required simulator Sim can be constructed by returning a *simulation failure* $\perp$ whenever the exact distribution $\mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x))$ cannot be recreated without access to x and ensuring that

$$\mathsf{Pr}[\mathsf{Sim}(C, \tilde{\mathcal{W}}) = \perp] = \epsilon \,,$$

and, conditionned to the event $\mathsf{Sim}(C, \tilde{\mathcal{W}}) \neq \perp$,

$$\mathsf{Sim}(C, \tilde{\mathcal{W}}) \equiv \mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x)) \,.$$

Relation to Random Probing Model. The *random probing model* [31,2,10] is a specific instance of the above-defined general random probing model. Specifically, the probability distribution $\mathbb{D}\mathcal{W}^{\infty}$ is selected such that each wire $w \in \mathcal{W}$ leaks with the same probability p independent of all other wires.

As shown by Duc et *al.* [22], the random probing model can be seen as an intermediate model between the noisy leakage model [35] and the d-threshold probing model [31]. Here, the assumption of the mutually independent leakage probability p for all wires can be traced to the assumption of equal and mutually independent noise in the noisy leakage model. However, the latter does not necessarily hold in practice, neither in hardware nor in software [9]. By defining the leakage over an arbitrary discrete probability distribution $\mathbb{D}\mathcal{W}^{\infty}$, we can model scenarios where the noise of two wires is not independent (e.g., because they operate in parallel with the same background computation) and, hence, the leakage probability of the two wires is dependent. Similarly, dependencies in the leakage probability can also be caused by physical defaults such as glitches [34,36] and couplings [16,36] or by shared structures like the Power Distribution Network (PDN) [44].

The introduction of the arbitrary discrete probability distribution $\mathbb{D}\mathcal{W}^{\infty}$ also allows two different wire combinations of one wire $\{w_i\}$ and $\{w_j\}$ to have different leakage probabilities p_i and p_j, respectively. With that, it enables more fine-grained modeling for the contribution of individual wires to the occurring leakage of wire combinations (by a different weight for the wires w_i and w_j to occur). For example, an EM probe usually does not capture the entire circuit but only a subset of neighboring circuit elements. Hence, a subset of wires does not contribute to the leakage at all and can be modeled by wire combinations with zero probability. Modeling this correctly can lead to place-and-route algorithms that take EM probing into account and minimize the observable leakage. Other differences in the leakage probability may be caused by the difference in the driver strengths of individual gates, or even by the type of operation a wire is used for.

By introducing this general model, we hope to fuel research into the nature of physical leakage by looking specifically into the leakage dependencies and contributions of different circuit structures. However, due to the computational blow-up, we will stick to mutually independent leakage probabilities in the practical implementation of our evaluation tools (cf. Section 5).

3.2 General Random Fault Security

Adversary Model. We use the adversary model proposed by Feldtkeller et *al.* [26] that is based on a fault model from Richter-Brockmann et *al.* [39]. Here, a faulting adversary can invoke a circuit C multiple times, where on each invocation, a random subset of gates are manipulated according to a specified fault transformation. More specifically, a fault consists of a *fault location* $g \in \mathcal{G}$, i.e., a gate of the circuit, and a *fault transformation* $\tau : \mathcal{G}_{\mathsf{f}} \to \mathcal{G}_{\mathsf{f}}$, i.e., a transformation of the Boolean function a gate implements, where the fault model restricts the allowed transformations. Popular fault transformations are, e.g., $\tau_{\mathsf{set}}(g) = \mathsf{one}$, $\tau_{\mathsf{reset}}(g) = \mathsf{zero}$, or $\tau_{\mathsf{flip}}(g) = \mathsf{inv}(g)$. Hence, a fault is a pair $f = (g, \tau)$ and we denote by $\mathcal{F}$ the set of all possible faults, by $\mathcal{F}^{\infty} = \{\tilde{\mathcal{F}} \subseteq \mathcal{F}\}$ the set of all possible fault combinations $\tilde{\mathcal{F}}$ with distinct locations (i.e., each $g \in \mathcal{G}$ occurs at most once in $\tilde{\mathcal{F}}$), and by $\mathbb{D}\mathcal{F}^{\infty}$ an arbitrary distribution defined over $\mathcal{F}^{\infty}$. We define the following function to select a fault combination for a single circuit invocation:

$\texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^{\infty})$: For a given circuit C, the faulty-gate sampler selects a fault combination $\tilde{\mathcal{F}}$ with probability $\mathsf{Pr}_{\mathbb{D}\mathcal{F}^{\infty}}[\tilde{\mathcal{F}}]$ and outputs the modified circuit $C^{\tilde{\mathcal{F}}}$ that we refer to as a *faulty circuit.*

Note that $\tilde{\mathcal{F}}$ can be empty and, if $\mathsf{Pr}_{\mathbb{D}\mathcal{F}^{\infty}}[\emptyset] > 0$, then the sampler can output the original circuit $C = C^{\emptyset}$. For the sake of simplicity, it still falls within the definition of a faulty circuit.

The original adversary model introduced by Feldtkeller et *al.* provides the adversary with a correct and a faulty output of the circuit C. For our purposes, we define the leakage of faults by the correctness of the output, which is a conservative but popular choice for fault security [5,20,37,38]. For that, we define a decoding or correction gadget for the context of faulty circuits:

G^D : The decoding gadget realizes a function such that, given an input with at most k bit faults, outputs a corrected result.

Then, we define the leakage by the correctness of the output of the faulty circuit. More formally, we define the *random fault leakage* $\mathcal{L}_{\tilde{\mathcal{F}}}(C, x)$ as the output of the random experiment, with

$$C^{\tilde{\mathcal{F}}} \leftarrow \mathtt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^{\infty}),$$

$$\mathcal{L}_{\tilde{\mathcal{F}}}(C) \leftarrow \begin{cases} 0 & \text{if } \forall x : C(x) = G^D(C^{\tilde{\mathcal{F}}}(x)), \\ 1 & \text{else.} \end{cases}$$

In contrast to $\mathcal{L}_{\tilde{\mathcal{W}}}(C, \mathsf{enc}(x))$, the random fault leakage is not a distribution but, for each fault combination $\tilde{\mathcal{F}}$, a constant Boolean value. With this, we define the view of the adversary $\mathcal{A}_f$, after injecting a random fault into a circuit C, as $\mathcal{L}_{\tilde{\mathcal{F}}}(C)$. Note, that we focus on correction-based countermeasures here. However, detection-based countermeasures can be treated analogously by considering the result secure if a fault was detected correctly.

As with the general random probing model, this adversary model allows the modeling of a wide range of different adversarial capabilities [26]. For example, the set of fault combinations $\mathcal{F}^{\infty}$ can be set to register combinations with long computation paths to model clock glitches. Or it can be set to a set of adjacent gates to model a laser attack. Similarly, a distribution of faults $\mathbb{D}\mathcal{F}^{\infty}$ with a small variance can be used to model an adversary with precise faulting capabilities, while a broader distribution can be used for a more dispersed fault behavior.

Security Definition. We extend the above adversary models by providing an appropriate definition for security. Intuitively, we say a fault combination leads to an insecure circuit behavior if there exists some input assignment that cannot be corrected at the output. This is a very conservative assumption, in that it assumes an adversary who can exploit every effective fault at the output to gain full knowledge of the secret, and is popular in the literature [5,20,37,38]. To account for the random behavior of our adversary model, we say a circuit is *random fault secure* if the probability that the adversary will get exploitable information is bounded by some (small) μ.

Definition 2 (General Random Fault Security). *A circuit C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure with respect to a decoding G^D if:*

$$\mathsf{Pr}[\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1] \leq \mu,$$

where $\mathcal{L}_{\tilde{\mathcal{F}}}()$ is computed from the random experiment

$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^{\infty}).$$

In this definition, the decoding gadget required for the computation of the fault leakage $\mathcal{L}_{\tilde{\mathcal{F}}}(C)$ cannot be faulted by $\mathcal{A}_f$. This is symmetric to the encoding enc in random probing security, which is not probed, and can be justified by the fact that a fault in a final correction can only leak the output of the circuit. Note, however, that any correction implemented within the circuit is subject to faults in our model.

Relation to Random Fault Model. Dhooghe and Nikova already proposed a *random fault model* [19], which is inspired by the random probing model. However, their model differs from our proposal in two key features: (i) They consider faults in wires allowing only the fault transformations set, reset, and flip. In contrast, we model faults by a transformation of gate functions which allows a wide range of possible fault scenarios, including set, reset, and flip [39]. (ii) They consider an adversary that has precise control over the location of the fault where the occurrence of the fault is randomly determined by an independent probability κ. In contrast, we allow an arbitrary distribution over fault combinations. Hence, different fault locations can be dependent and the fault type can be uncertain as well. This allows a wide range of possible adversarial scenarios [26]. Therefore, our new model is a generalization of the previously proposed random fault model.

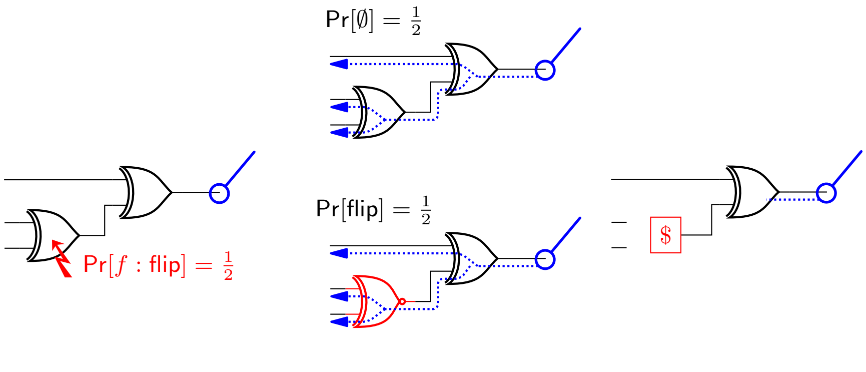
3.3 General Random Combined Security

Adversary Model. We now introduce a model for an adversary that can both inject faults and place probes simultaneously. As such, the resulting *combined* adversary will have the capability to manipulate a random set of gates of a circuit. Then, both the exact values of a random subset of wires and the correctness of the circuit are leaked to the adversary. To formally capture the view of the adversary, we define the *random combined leakage* $\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x)$ as the output of the random experiment

$$\begin{aligned} C^{\tilde{\mathcal{F}}} &\leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{F}^{\infty})\,, \\ \tilde{\mathcal{W}} &\leftarrow \texttt{LeakingWires}(C^{\tilde{\mathcal{F}}}, \mathbb{D}\mathcal{W}^{\infty})\,, \\ \mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x) &\leftarrow \texttt{AssignWires}(C^{\tilde{\mathcal{F}}}, \tilde{\mathcal{W}}, x) \parallel \mathcal{L}_{\tilde{\mathcal{F}}}(C)\,. \end{aligned}$$

Again, we assume a correction-based countermeasure for simplicity. When detection is used, it is important to include the detection flag in the random combined leakage (which then need to be simulated alongside the leaking wires).

Considering the SCA aspect of CA, we require for security that the leaking wires $\tilde{\mathcal{W}}$ can be simulated without knowledge of the secret. However, for simulation, the adversarial knowledge of the faulty circuit makes a difference. Imagine



(a) Fault scenario with a probabilistic flip fault.

(b) View of the simulator with known faulty circuit $C^{\tilde{\mathcal{F}}}$.

(c) View of the simulator with known fault distribution $\mathbb{D}\mathcal{F}^{\infty}$.

Fig. 1. Difference when knowing the injected fault versus knowing only the fault distribution. When knowing the fault the circuit is deterministic (for deterministic inputs) and a probe on the output propagates to all inputs. When only the fault distribution is known, the circuit can get probabilistic (even with deterministic inputs) potentially stopping the propagation of probes.

the scenario in Figure 1, where a flip fault is injected into a xor chain with probability $\frac{1}{2}$. If the adversary knows the faulty circuit $C^{\tilde{\mathcal{F}}}$ (and, hence, $C^{\tilde{\mathcal{F}}}$ is given to the simulator) then all deterministic inputs to the xor chain are required to simulate the probe at the end in both cases (non-faulty/faulty). In contrast, if the adversary does not know $C^{\tilde{\mathcal{F}}}$ (and only the fault distribution $\mathbb{D}\tilde{\mathcal{F}}^{\infty}$ is given to the simulator) then the fault randomizes the intermediate value and the output of the final xor can be simulated by a uniform random value. Hence, the fault effectively works as a mask refreshing. For this, we introduce two different combined adversaries: one without and one with knowledge of the faulty circuit.

i) Unknown-Fault Random Combined Adversary. Our first adversary $\mathcal{A}_{uc}$ is the combination of $\mathcal{A}_p$ and $\mathcal{A}_f$ without knowledge of the randomly chosen fault combination $\tilde{\mathcal{F}}$. Specifically, the adversary gets no access to the faulty circuit $C^{\tilde{\mathcal{F}}}$ and the corresponding simulator gets only the circuit C and the fault distribution $\mathbb{D}\mathcal{F}^{\infty}$ as input. Hence, the view of $\mathcal{A}_{uc}$ is defined by $\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x)$ and the effects of fault injection and probing is interleaved, i.e., we analyze the circuit in Figure 1c.

ii) Known-Fault Random Combined Adversary. The second adversary $\mathcal{A}_{kc}$ is the combination of $\mathcal{A}_p$ and $\mathcal{A}_f$ with additional knowledge of the faulty circuit $C^{\tilde{\mathcal{F}}}$ (or equivalently, the selected fault combination $\tilde{\mathcal{F}}$). For this, the corresponding simulator has access to $C^{\tilde{\mathcal{F}}}$ for the simulation of the leaking wires. With this, the view of $\mathcal{A}_{kc}$ is defined by $\mathcal{L}_{\tilde{\mathcal{W}},\tilde{\mathcal{F}}}(C, x) \parallel \tilde{\mathcal{F}}$. Since the faulty circuit is known, we can analyze $\mathcal{L}_{\tilde{\mathcal{W}}}(C^{\tilde{\mathcal{F}}}, x)$ and $\mathcal{L}_{\tilde{\mathcal{F}}}(C)$ independently, i.e., after the fault injection, we only consider one of the two circuits in Figure 1b.

When comparing the two adversaries, it becomes apparent that $\mathcal{A}_{uc}$ is the more realistic adversary model for CA because, in a real-world circuit, the adversary usually does not know the exact effect of an injected fault. However, the uncertainty about the faults makes the analysis of combined security much more complex, due to the reciprocal effects of faults and probes. Fortunately, we can show that any circuit secure against $\mathcal{A}_{kc}$ is also secure against $\mathcal{A}_{uc}$. For this, we see $\mathcal{A}_{kc}$ mostly as a useful abstraction for analysis, allowing a clear path to security verification.

In this sense, we can make a further distinction in the knowledge an adversary has about the effect of fault injection. Specifically, we can separate the knowledge about the fault location and its effect. For this, we use the known-fault adversary $\mathcal{A}_{kc}$ in combination with a fault transformation to a probabilistic gate function $\mathsf{f_g}$, i.e., $\mathsf{f_g}$ has an internal random tape that impacts the output of the gate. While this pushes the analysis closer to $\mathcal{A}_{uc}$, it is important to evaluate the correctness of the circuit for all values of the random tape, i.e., $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0$ if the output can be corrected for all inputs and random values of probabilistic gate functions.

Our model also allows to go in the opposite direction by empowering $\mathcal{A}_{kc}$ with direct control over the injected fault. Hence, we can model a *chosen-fault* combined adversary as the adversary $\mathcal{A}_{kc}$ that can freely choose the fault distribution $\mathbb{D}\mathcal{F}^{\infty}$ of a (restrict) set of fault combinations $\mathcal{F}^{\infty}$. In most cases, this will lead to an adversary that directly chooses the location and transformation of the injected fault, since any uncertainty will reduce the advantage of the adversary, which removes the random nature of the model. Note, if the set $\mathcal{F}^{\infty}$ is restricted to the set of all gate combinations with up to k faults, this model is equivalent to the popular threshold fault model [1,5,30,38,45]. However, since the semantics of the security definition do not change (we do not care how the fault distribution is generated), we do not introduce a specific adversary model for this case.

Security Definition. We provide security definitions for the setting of combined adversaries. Intuitively, we say that a circuit is combined-secure if the view of the corresponding adversary can be simulated and the output can be corrected with high probability. For this, we introduce a new security parameter γ that represents the advantage of the adversary, i.e., the probability that the adversary gains some knowledge about the secret. In the combined setting, the adversary wins if either there exists some fault leakage ($\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1$) or the leaking wires in the faulty circuit cannot be simulated without knowledge of the secret. To overcome the dependencies between those two events, we only conduct the simulation of the leaking wires if there is no fault leakage. Hence, we get the following two parameters (where we usually choose the lowest possible value):

$$\mu \geq \mathsf{Pr}[\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1]\,,$$

$$\epsilon \geq \mathsf{Pr}[\mathsf{Sim}(C, \tilde{\mathcal{W}}) \equiv \perp \mid \mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0]\,.$$

Here, μ is defined exactly as in Definition 2, while ϵ is the probability that the simulation fails knowing that $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0$. We then express the advantage of the combined adversary by

$$\gamma \geq \mu + (1-\mu)\epsilon \,.$$

To compute ϵ, we introduce a new subset of the fault combination $\mathcal{B} \subseteq \mathcal{F}^{\infty}$, such that $\mathcal{B}$ captures all fault combinations that always yield a result that can be corrected, i.e., $\mathcal{B} = \{\tilde{\mathcal{F}} \in \mathcal{F}^{\infty} \mid \mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0\}$. Then, we define the distribution $\mathbb{D}\mathcal{B}$ as the scaled distribution $\mathbb{D}\mathcal{F}^{\infty}$ conditioned to the event $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 0$.

We start with the security definition for the adversary $\mathcal{A}_{uc}$, where the random combined leakage needs to be simulated. In a sense, this is the most natural definition as the adversary has no knowledge about the circuit transformation caused by the injected fault.

Definition 3 (Unknown-Fault Random Combined Security). *A circuit C is* $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma)$-Unknown-Fault Random Combined Secure ($\mathrm{RCS}_{\mathrm{UF}}$) *with respect to an value encoding* enc *and a error decoding G^D if there exists some $\mu, \epsilon \leq 1$ such that C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure with respect to G^D, there exists a simulator* Sim *such that for all inputs x:*

$$\mathsf{Sim}(C, \mathbb{D}\mathcal{F}^{\infty}) \approxeq_{\epsilon} \mathcal{L}_{\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{B}}(C, \mathsf{enc}(x))\,,$$

and

$$\mu + (1-\mu)\epsilon \leq \gamma\,,$$

where $\mathcal{L}_{\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{B}}()$ is computed from the random experiment

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C, \mathbb{D}\mathcal{W}^{\infty})\,,$$
$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{B})\,.$$

While simple, this definition is difficult to analyze due to the interleaving of probes and faults within the simulator. Therefore, we provide a second security definition tailored to $\mathcal{A}_{kc}$. Because this adversary knows the faulty circuit $C^{\tilde{\mathcal{F}}}$, the interleaving of probes and faults is eliminated. Hence, the analysis gets simpler and we will later show that security in this model implies security with unknown faults.

Definition 4 (Known-Fault Random Combined Security). *A circuit C is* $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma)$-Known-Fault Random Combined Secure ($\mathrm{RCS}_{\mathrm{KF}}$) *with respect to an value encoding* enc *and an error decoding G^D if there exists some $\mu, \epsilon \leq 1$ such that C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure with respect to G^D, there exists a simulator* Sim *such that for all inputs x it holds that*

$$C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{B})\,,$$
$$\mathsf{Sim}(C^{\tilde{\mathcal{F}}}) \approxeq_{\epsilon} \mathcal{L}_{\tilde{\mathcal{W}}}(C^{\tilde{\mathcal{F}}}, x),$$

and

$$\mu + (1-\mu)\epsilon \leq \gamma\,,$$

where $\mathcal{L}_{\tilde{\mathcal{W}}}()$ is computed from the random experiment

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(C^{\tilde{\mathcal{F}}}, \mathbb{D}\mathcal{W}^{\infty})\,.$$

Reduction between Security Definitions. It is evident that the adversary $\mathcal{A}_{kc}$ has more knowledge about the probed circuit structure than $\mathcal{A}_{uc}$. Hence, it seems reasonable that $\mathrm{RCS}_{\mathrm{UF}}$ is the more general security notion and we can reduce its security to $\mathrm{RCS}_{\mathrm{KF}}$. In other words, if a circuit is $\mathrm{RCS}_{\mathrm{KF}}$, then it is always $\mathrm{RCS}_{\mathrm{UF}}$.

In particular, we can show that if there exists a simulator Sim_{KF} for the adversary $\mathcal{A}_{kc}$ then we can always construct a simulator Sim_{UF} for the adversary $\mathcal{A}_{uc}$ with at most the failure probability of Sim_{KF}. For that, consider that $\mathrm{RCS}_{\mathrm{KF}}$ requires the simulation for a given but randomly chosen faulty circuit $C^{\tilde{\mathcal{F}}}$. Hence, we can compute the failure probability of Sim_{KF} as the sum of the failure probability for each faulty circuit weighted by the probability of that circuit to occur, i.e.,

$$\epsilon_{kf} = \sum_{\tilde{\mathcal{F}}} \mathsf{Pr}[\tilde{\mathcal{F}}]\epsilon_{kc}^{\tilde{\mathcal{F}}}\,, \tag{1}$$

where $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ is the failure probability of Sim_{KF} given the faulty circuit $C^{\tilde{\mathcal{F}}}$. We can then construct a simulator that randomly selects a faulty circuit and calls Sim_{KF} for the wire simulation. Note, that standalone fault security is not affected by the choice of the adversary. We show a more formal argumentation below.

Theorem 1. *Let C be a circuit that is $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{kc})$-$RCS_{KF}$. Then C is $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{uc})$-$RCS_{UF}$ with $\gamma_{uc} \leq \gamma_{kc}$.*

Proof. Let C be a $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{kc})$-$\mathrm{RCS}_{\mathrm{KF}}$ circuit. Then, by definition of $\mathrm{RCS}_{\mathrm{KF}}$, C is $(\mathbb{D}\mathcal{F}^{\infty}, \mu)$-random fault secure and there exists a simulator Sim_{KF} with failure probability ϵ_{kc} for the probing leakage $\mathcal{L}_{\tilde{\mathcal{W}}}(C^{\tilde{\mathcal{F}}}, x)$ such that

$$\gamma_{kc} \geq \mu + (1-\mu)\epsilon_{kc}\,.$$

Now we construct a simulator Sim_{UF} for $\mathrm{RCS}_{\mathrm{UF}}$ out of Sim_{KF}. This is possible because Sim_{UF} has to simulate the same events as Sim_{KF} since the set of fault combinations with $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1$ remains untouched by the knowledge of the adversary. The simulator Sim_{UF} is constructed by the following two steps:

1. Call $C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \mathbb{D}\mathcal{B})$,
2. Return $\mathsf{Sim}_{KF}(C^{\tilde{\mathcal{F}}}, \tilde{\mathcal{W}})$.

When this simulator does not fail it is a perfect simulation of the leaking wires because it draws $\tilde{\mathcal{F}}$ from the correct distribution and Sim_{KF} produces a perfect simulation for any faulty circuit $C^{\tilde{\mathcal{F}}}$ (if it does not fail). Given this, the failure probability of Sim_{UF} is given by

$$\epsilon_{uf} = \sum_{\tilde{\mathcal{F}}} \mathsf{Pr}[\tilde{\mathcal{F}}]\epsilon_{kc}^{\tilde{\mathcal{F}}}\,,$$

where $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ is the failure probability of Sim_{KF} given the faulty circuit $C^{\tilde{\mathcal{F}}}$. Hence, we have $\epsilon_{uc} = \epsilon_{kc}$ (see Equation 1) for this simulator (however, there could be a simulator with a smaller failure probability). Together with $(\mathcal{F}^{\infty}, \mu)$-random fault security of C, it follows that C is $(\mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma_{uc})$-$\mathrm{RCS}_{\mathrm{UF}}$ with $\gamma_{uc} \leq \gamma_{kc}$. □

In the remainder of this paper, we will mostly focus on $\mathrm{RCS_{KF}}$ for the sake of simplicity.

Relation Random Combined Model. In addition to the random fault model, Dhooghe and Nikova also propose a combined version of the random probing and random fault model [19]. Their model does not leak the exact occurring fault to the adversary (similar to $\mathrm{RCS_{UF}}$). Since they use their version of the random fault model and the traditional random probing model, the combination has the same limitations as the individual models. In addition, our model introduces a new security parameter γ that captures the overall advantage of the combined adversary. As with the individual models, the parameters of our combined model can be chosen such that it is equal to the definition from Dhooghe and Nikova. While Dhooghe and Nikova propose this model in an appendix of their work, they do not conduct any analysis or further investigations of it.

4 Compositional Notions

Analyzing entire circuits for their security is often prohibitively complex. As a result, the research community focuses on the construction of so-called *gadgets*, i.e., small circuits implementing a small function (often single binary operations) in a secure manner such that security is guaranteed even under composition [7,10,13,20,27,37]. To abstractly argue about the security via composition, we first define a *composability notion*, which defines the properties a gadget must fulfill, usually, by restricting the propagation of leakage or faults. Second, we outline and prove the conditions of composition, i.e., how the gadgets can be securely combined, in a *composition theorem*.

4.1 Composition in the Random Probing Model

For the general random probing model, we can build upon the notion of Random Probing Composability (RPC) proposed by Belaïd et *al.* [10] and extend it for general distributions. At a high level, RPC provides $\mathcal{A}_p$ with the usual random probing leakage for the gadget and additionally with a tuple $\mathcal{O}$ of arbitrary but bounded sets of probes on shares of each output. More specifically, for each unshared output with index i, there is a bounded set $\mathcal{O}_i$ that contains the share indices of the probed output shares. Then, a gadget is composable if the view of $\mathcal{A}_p$ can be simulated with high probability using only bounded sets of input shares, i.e., for each unshared input i the simulator requires only a bounded set of share indices $\mathcal{I}_i$. Here, the set of input and output shares are bounded by the same parameter d, i.e., $|\mathcal{O}_i| \leq d$ and $|\mathcal{I}_i| \leq d$. The extension to arbitrary probability distributions is straightforward: the internal wire selection uses $\mathbb{D}\mathcal{W}^\infty$ instead of a wire-independent leakage probability.

Definition 5 (General Random Probing Composability (GRPC)). *A gadget $G : (\mathbb{F}_2^s)^h \to (\mathbb{F}_2^s)^m$ is $(d, \mathbb{D}\mathcal{W}^\infty, \epsilon)$-GRPC if there exists a deterministic* *algorithm* ShareSelect *and a probabilistic simulator* Sim *such that for every input x and for every tuple of sets $\mathcal{O} = (\mathcal{O}_i)_{i\in[m]}$, with $\forall i : \mathcal{O}_i \subseteq [s]$ and $|\mathcal{O}_i| \leq d$, there exists a $\epsilon \leq 1$ such that the random experiment*

$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(G, \mathbb{D}\mathcal{W}^{\infty})$$
$$\mathcal{I} := (\mathcal{I}_1 \ldots \mathcal{I}_h) \leftarrow \texttt{ShareSelect}(\tilde{\mathcal{W}}, \mathcal{O})$$
$$out \leftarrow \mathsf{Sim}(x|_{\mathcal{I}}, \mathcal{I}, \mathcal{O}, \tilde{\mathcal{W}})$$

yields

$$\mathsf{Pr}[(|\mathcal{I}_1| > d) \vee \ldots \vee (|\mathcal{I}_h| > d)] \leq \epsilon$$
$$out \equiv (\texttt{AssignWires}(G, \tilde{\mathcal{W}}, x), y|_{\mathcal{O}}),$$

with $y \leftarrow G(x)$.

Arguing about the composition of gadgets requires knowledge of the joint probability distribution for the leakage in multiple gadgets. Using a complex (and for some scenarios more realistic) joint probability distribution, where the leakage of different gadgets is mutually dependent, contradicts the core idea of composition, i.e., the independent analysis of individual gadgets. Hence, in the following, we assume mutually independent leakage distributions for each gadget. While this is certainly a restriction in the generality of our notion, we think it is justified by the significant gains of the gadget-based approach.

Gadgets supporting GRPC can be composed in an arbitrary way as long as each output of a gadget is used only once as input to another gadget. However, the failure probability increases linearly with the number of used gadgets. Our theorem is actually a generalization of Theorem 1 from Belaïd et *al.* [10], and the proof is similar except for the computation of the global failure probability. Instead of being $1 - (1-\epsilon)^{|C|}$, bounded by $|C| \cdot \epsilon$, the latter naturally becomes $1 - \Pi_{i=1}^{|C|}(1-\epsilon_i)$ to reflect more tightly the individual gadget-simulation failures.

Theorem 2. *Let C be a circuit constructed by composition of $(d, \mathbb{D}\mathcal{W}_i^{\infty}, \epsilon_i)$-GRPC gadgets G_i, for $i \in \{1, \ldots, |C|\}$, such that each output of G_i is used as input of at most one other gadget G_j or as output of C. Given this, C is $(\prod_{i=1}^{|C|} \mathbb{D}\mathcal{W}_i^{\infty}, 1 - \Pi_{i=1}^{|C|}(1-\epsilon_i))$-random probing secure.*

Relation to (Random) Probing-Secure Composition. As mentioned, our notion is a direct extension of RPC [10] to arbitrary probe distributions. Another concept for the construction of random probing secure circuits is *expansion* [4,10]. For this, a circuit compiler, which replaces each gate with a respective secure gadget, is applied e-times to a circuit. Hence, for $e = 1$, we get the normal gadget-based approach. However, for $e = 2$, each share in the previous masked circuit is shared again and each gate of a gadget is replaced with the respective gadget of the gate again. In essence, this reduces the leakage probability of wires as now the simulation of entire gadgets needs to fail in order to leak a single share (of the originally masked circuit). Hence, the leakage probability

p is replaced by the simulation-failure probability ϵ. Using the right gadgets can result in an exponential (in e) security increase, but also an exponential increase in implementation costs. We leave the generalization to arbitrary probe distributions for future work.

In the standard Ishai-Sahai-Wagner (ISW) probing model [31], where an adversary can choose up to d wires to be probed, the requirements for compositions are now well understood. In particular, it requires to restrict and guide the flow of probe propagation through the circuit. While (Strong) Non-Interference ((S)NI) [6,7] restricts the number of input shares that can be used for the simulation of probes, Probe-Isolating Non-Interference (PINI) [13] restricts the location (in terms of share domains) of input shares the simulator can use. Hence, (G)RPC has some similarity with (S)NI, in that it restricts the number of input shares a simulator can access. However, in contrast to (S)NI, the number of shares the simulator can access is a fixed parameter that is independent of the amount of leakage within the gadget.

4.2 Composition in the Random Fault Model

We continue with compositional statements for the random fault model. In contrast to the random probing model, to the best of our knowledge, there exists no prior work that looks into the composition of the random fault model. However, because of the duality of probes and faults, we can adapt the above notation from the random probing to the random fault case. However, instead of adding probes on outputs we now consider additional faults on inputs and check for correction instead of simulation.

To properly argue about the composition we need to specify the used countermeasure (similar to masking for probing). Hence, in the following, we will only consider circuits secured by simple repetition against a fault adversary. In particular, we will use $2k + 1$ repetitions such that a majority vote can be used for the correction of up to k faults. Then, for composition, we consider an adversary that can inject a random fault combination into the gadget where additionally a tuple $\mathcal{I}'$ of arbitrarily but bounded sets of inputs are potentially manipulated by faults. More specifically, for each input with index i, there is a bounded set $\mathcal{I}'_i$ that contains the replication indices of inputs that may be affected by a fault. Here, we consider each replication in $\mathcal{I}'_i$ to be a unique and independent input, to account for all possible distribution changes due to a fault in previous parts of the composed circuit. Hence, a fault on an input wire can be seen as an arbitrary change in the value distribution over $\mathbb{F}_2$. Then, a gadget is composable, if the output can be corrected with high probability.

To formally define our notion of composition in the presence of a random fault adversary, we start by defining Restricted Random Fault Composability (RRFC), which states the required property for a fixed tuple of input faults $\mathcal{I}'$.

Definition 6 (Restricted Random Fault Composability). *Let $n = 2k+1$ and $\mathcal{I}' = (\mathcal{I}'_i)_{i\in[h]}$ be a tuple of sets such that $\mathcal{I}'_i \subseteq [n]$, for all i: $|\mathcal{I}'_i| \leq k$. A gadget*

$G : (\mathbb{F}_2^n)^h \to (\mathbb{F}_2^n)^m$ is $(\mathcal{I}', k, \mathbb{DF}^\infty, \mu)$-RRFC if there exists a deterministic algorithm `ReplicationSelect` *such that the random experiment*

$$G^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(G, \mathbb{DF}^\infty)\,,$$
$$\mathcal{O}' := (\mathcal{O}'_1, \ldots, \mathcal{O}'_m) \leftarrow \texttt{ReplicationSelect}(G^{\tilde{\mathcal{F}}}, \mathcal{I}')$$
$$\mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) \leftarrow \begin{cases} 0 & \textit{if } \forall i \le m : |\mathcal{O}'_i| \le k\,, \\ 1 & \textit{else.} \end{cases}$$

yields

$$\mathsf{Pr}[\mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) = 1] \le \mu$$
$$G^{\tilde{\mathcal{F}}}(x')|_{\overline{\mathcal{O}'}} \equiv G(x)|_{\overline{\mathcal{O}'}}$$

for all inputs $x \in \mathbb{V}_n^h$ and faulty inputs $x' \in (\mathbb{F}_2^n)^h$ with $x|_{\overline{\mathcal{I}'}} = x'|_{\overline{\mathcal{I}'}}$.

Then, a gadget supports Random Fault Composability (RFC) with some bound μ if for all possible tuples $\mathcal{I}'$ the RRFC failure probability is bounded by μ.

Definition 7 (Random Fault Composability). *A gadget $G : (\mathbb{F}_2^n)^h \to (\mathbb{F}_2^n)^m$ with $n = 2k+1$ is $(k, \mathbb{DF}^\infty, \mu)$-RFC if for all tuples of sets $\mathcal{I}' = (\mathcal{I}'_i)_{i\in[h]}$ with $\forall i : \mathcal{I}'_i \subseteq [n]$ and $|\mathcal{I}'_i| \le k$ the gadget G is $(\mathcal{I}', k, \mathbb{DF}^\infty, \mu_{\mathcal{I}'})$-RRFC and* $\max_{\mathcal{I}'}\{\mu_{\mathcal{I}'}\} \le \mu$.

While we split the definition of RFC into two parts (for reasons that become apparent when we look at composition under a combined adversary) it is easy to see that there is a close symmetry with the definition of GRPC. Where GRPC goes through all possible tuples of output probes $\mathcal{O}$, RFC goes through all possible tuples of input faults. Where GRPC restricts the number of shares per input the simulator can use for a successful simulation, RFC restricts the number of replications per output that can be affected by a fault.

We now show that gadgets supporting RFC can be composed arbitrarily. Under the assumption of mutually independent fault distributions for each gadget, the failure probability increases linearly with the number of gadgets in the circuit.

Theorem 3. *Let C be a circuit constructed by composition of $(k, \mathbb{DF}_i^\infty, \mu_i)$-RFC gadgets G_i, for $i \in \{1, \ldots, |C|\}$. Then, C is $(\prod_{i=1}^{|C|} \mathbb{DF}_i^\infty, 1 - \prod_{i=1}^{|C|}(1-\mu_i))$-random fault secure.*

Proof. Let the faulty circuit be $C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \prod_{i=1}^{|C|} \mathbb{DF}_i^\infty)$, with $\tilde{\mathcal{F}}$ the set of faults selected with $\mathsf{Pr}_{\prod_{i=1}^{|C|} \mathbb{DF}_i^\infty}[\tilde{\mathcal{F}}]$. We can divide $\tilde{\mathcal{F}}$ into $|C|$ disjoint fault sets $\tilde{\mathcal{F}}_i \subseteq \tilde{\mathcal{F}}$ such that $\tilde{\mathcal{F}}_i$ belongs to G_i and is selected with $\mathsf{Pr}_{\mathbb{DF}_i^\infty}[\tilde{\mathcal{F}}_i]$.

We go iteratively through the gadgets, starting with the gadgets only connected to the inputs of C. Let G_i be such a gadget. Then, by definition of $(k, \mathbb{DF}_i^\infty, \mu_i)$-RFC, the gadget G_i is $(\mathcal{I}'_{G_i}, k, \mathbb{DF}_i^\infty, \mu_i)$-RRFC for the tuple of empty sets $\mathcal{I}'_{G_i} = (\mathcal{I}'_{G_i,j} = \emptyset)_{j\in[h]}$. Hence, there exists a tuple of sets $\mathcal{O}'_{G_i}$ such that $\mathsf{Pr}[\mathcal{L}_{\mathcal{I}'_{G_i},\tilde{\mathcal{F}}}(G) = 1] \leq \mu$ and $G_i^{\tilde{\mathcal{F}}_i}(x)|_{\overline{\mathcal{O}'_{G_i}}} \equiv G_i(x)|_{\overline{\mathcal{O}'_{G_i}}}$.

We continue with the child gadgets, i.e., gadgets that have inputs connected to outputs of the just handled gadgets and (potentially) inputs of the circuit. Let G_j be such a gadget. Again, we can create a tuple of sets $\mathcal{I}'_{G_j}$ out of the respective tuples $\mathcal{O}'_{G_i}$ of the parent gadgets G_i. Because of $(k, \mathbb{D}\mathcal{F}_j^{\infty}, \mu_j)$-RFC there exists tuple of sets $\mathcal{O}'_{G_j}$ such that $\mathsf{Pr}[\mathcal{L}_{\mathcal{I}'_{G_j},\tilde{\mathcal{F}}}(G) = 1] \leq \mu$ and $G_j^{\tilde{\mathcal{F}}_j}(x)|_{\overline{\mathcal{O}'_{G_j}}} \equiv G_j(x)|_{\overline{\mathcal{O}'_{G_j}}}$. We repeat this process until we reach the outputs of C.

Since we have $n = 2k{+}1$ replications, we can construct a decoding gadget G^D for C by computing the majority of the output wires $w_i^1, \dots, w_i^n$ for all $i \in [m]$. This decoding gadget will correct an output value as long as the number of faulty replications is smaller or equal to k, which is always true if none of the gadgets G_i, for $i = 1, \dots |C|$, fail with respect to RFC. The probability that at least one of $|C|$ gadgets fail is $1{-}\prod_{i=1}^{|C|}(1{-}\mu_i)$. Hence, $\mathsf{Pr}[C(x) \equiv G^D(C^{\tilde{\mathcal{F}}}(x))] \leq 1{-}\prod_{i=1}^{|C|}(1{-}\mu_i)$, which shows random fault security of C. □

Note, that the above composition loses some tightness in μ because a failure of RFC in one gadget does not necessarily mean that the entire circuit is insecure, e.g., if some faults cancel each other out in a later gadget.

Relation to Fault-Secure Composition. To the best of our knowledge, this is the first work establishing a compositional property in the random fault model. However, in the threshold fault model [30], where an adversary can place up to k faults, composition is already discussed and notions usually have a strong symmetry to notions in the ISW probing model. In particular, (Strong) Non-Accumulation ((S)NA) [20] (later refined to Fault (Strong) Non-Interference (F-(S)NI) [37]) restricts the number of faults that can propagate to the output of a gadget, while Fault-Isolating Non-Interference (FINI) [27] limits fault propagation within so-called redundancy domains. Both variants ensure that the number of faults in the replication of a single value does not exceed the threshold that allows correction (detection) of faults. In this sense, our proposed notion has some similarity with (S)NA/F-(S)NI, in that it restricts the number of output replications that are allowed to be affected by a fault. However, similar to the contrast between (G)RPC and (S)NI, the number of allowed faulty outputs is not dependent on the amount of injected faults. Also, we use the same symmetry between faults and probes for the definition of our notion of composition.

4.3 Composition in the Random Combined Model

Finally, we provide a compositional statement for the random combined model, i.e., under an adversary with both faulting and probing capabilities. Here, we consider composition under an adversary that knows the injected fault and use the reduction from Section 3.3 for the adversary with unknown faults. We leave the tighter compositional statement in the setting with unknown faults for future work. In principle, the following notion is a combination of the two previous notions for the individual cases, however, with subtle differences.

First, we count faults in inputs and outputs per input and output share. More specifically, the tuple of indices for potentially faulty inputs has now a set for each share of each input, i.e., $\mathcal{I}' = ((\mathcal{I}'_{i,j})_{i\in[s]})_{j\in[h]}$, where j is the index of the input and i the share index. The same holds for the tuple of potential faulty outputs $\mathcal{O}' = ((\mathcal{O}'_{i,j})_{i\in[s]})_{j\in[m]}$, which is constructed by `ReplicationSelect`. Hence, we bound the number of allowed faults per input and output share by k.

Second, and in contrast, probes on outputs are extended to all replications of the probed value[4]. Hence, the tuple $\mathcal{O} = (\mathcal{O}_i)_{i\in[m]}$, where each $\mathcal{O}_i$ contains the probed share indices of the i'th output, does not change. However, the meaning of $j \in \mathcal{O}_i$ changes in so far as now all replications of the j'th share of the i'th output need to be simulated. This is necessary to allow for a wide range of gadget implementations where there are potential interdependencies between different replications, e.g., via a correction module, that allows probe propagation across replications [25]. Similarly, the simulator gets access to all replications of the shares indicated in the tuple $\mathcal{I} = (\mathcal{I}_i)_{i\in[h]}$.

Third, we define the failure probability for probe simulation $\epsilon_{\mathcal{I}'}$ in dependence on the tuple of faulty-input indices $\mathcal{I}'$. This is analog to the definition of RRFC (Definition 6) and allows us to iterate over all possible tuples $\mathcal{I}'$ for our combined composition. In particular, this enables a precise definition of the conditions for Random Probing Composition under Faults (RPCUF), which is essentially Definition 5 under a given tuple of potentially faulty inputs $\mathcal{I}'$ and a random fault combination $\tilde{\mathcal{F}} \in \mathcal{F}^\infty$.

Definition 8 (Random Probing Composition under Faults). *Let $\mathcal{I}' = ((\mathcal{I}'_{i,j})_{i\in[s]})_{j\in[h]}$ be a tuple of sets such that $\forall i,j : \mathcal{I}'_{i,j} \subseteq [n]$ and $|\mathcal{I}'_{i,j}| \leq k$. A gadget $G : ((\mathbb{F}_2^n)^s)^h \to ((\mathbb{F}_2^n)^s)^m$ with $n = 2k+1$ is $(\mathcal{I}', \mathbb{D}\mathcal{F}^\infty, d, \mathbb{D}\mathcal{W}^\infty, \epsilon)$-RPCUF if there exists a deterministic algorithm* `ShareSelect` *and a probabilistic simulator* Sim *such that for all faulty inputs $x' \in ((\mathbb{F}_2^n)^s)^h$, for which there exists an $x \in (\mathbb{V}_n^s)^h$ with $x|_{\overline{\mathcal{I}'}} = x'|_{\overline{\mathcal{I}'}}$, and every tuple of sets $\mathcal{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_m)$, with $\forall i : \mathcal{O}_i \subseteq [s]$ and $|\mathcal{O}_i| \leq d$, the random experiment*

$$G^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(G, \mathbb{D}\mathcal{F}^\infty)$$
$$\tilde{\mathcal{W}} \leftarrow \texttt{LeakingWires}(G^{\tilde{\mathcal{F}}}, \mathbb{D}\mathcal{W}^\infty)$$
$$\mathcal{I} := (\mathcal{I}_1 \ldots \mathcal{I}_h) \leftarrow \texttt{ShareSelect}(\tilde{\mathcal{W}}, \mathcal{O})$$
$$out \leftarrow \mathsf{Sim}(x'|_{\mathcal{I}}, \mathcal{I}, \mathcal{I}', \mathcal{O}, \tilde{\mathcal{W}})$$

yields

$$\mathsf{Pr}[(|\mathcal{I}_1| > d) \vee \ldots \vee (|\mathcal{I}_h| > d)] \leq \epsilon$$
$$out \equiv (\texttt{AssignWires}(G^{\tilde{\mathcal{F}}}, \tilde{\mathcal{W}}, x'), y'|_{\mathcal{O}})$$

with $y' \leftarrow G^{\tilde{\mathcal{F}}}(x')$.

[4] The same procedure should be used when analyzing replicated circuits for stand-alone GRPC.

Fourth, and similar to Section 3.3, we only check for side-channel security if the gadget is fault secure to keep the failure probabilities for probe simulation $\epsilon_{\mathcal{I}'}$ and correctness $\mu_{\mathcal{I}'}$ independent. For this, we again introduce a subset of fault combinations $\mathcal{B}_{\mathcal{I}'} \subseteq \mathcal{F}^{\infty}$, such that $\mathcal{B}_{\mathcal{I}'}$ captures all fault combinations that, in combination with potential faults on the inputs with indices in $\mathcal{I}'$, lead to a gadget output that can be corrected, i.e., $\mathcal{B}_{\mathcal{I}'} = \{\tilde{\mathcal{F}} \in \mathcal{F}^{\infty} \mid \mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) = 0\}$. Note, that we define this set to be dependent on the tuple $\mathcal{I}'$. The corresponding distribution $\mathbb{D}\mathcal{B}_{\mathcal{I}'}$ is defined as the scaled distribution $\mathbb{D}\mathcal{F}^{\infty}$ conditioned on the event $\mathcal{L}_{\mathcal{I}',\tilde{\mathcal{F}}}(G) = 0$.

With this, we say that a gadget supports Known-Fault Random Combined Composability (RCC_{KF}) if, for any tuple $\mathcal{I}'$, the gadget supports RRFC and RPCUF with negligible advantage for the adversary. To again compute a unified failure probability we chose the maximum combined failure probability (i.e., the probability that either RRFC fails or RPCUF fails under the condition that RRFC holds) over all tuples $\mathcal{I}'$.

Definition 9 (Known-Fault Random Combined Composability). *A gadget $G : ((\mathbb{F}_2^n)^s)^h \to ((\mathbb{F}_2^n)^s)^m$ is $(d, k, \mathbb{D}\mathcal{W}^{\infty}, \mathbb{D}\mathcal{F}^{\infty}, \gamma)$-$RCC_{KF}$ if for all tuples of sets $\mathcal{I}' = ((\mathcal{I}'_{i,j})_{i\in[s]})_{j\in[h]}$, such that $\forall i, j : \mathcal{I}'_{i,j} \subseteq [n]$ and $|\mathcal{I}'_{i,j}| \leq k$, there exists some $\mu_{\mathcal{I}'}, \epsilon_{\mathcal{I}'} \leq 1$ such that the gadget G is $(\mathcal{I}', k, \mathbb{D}\mathcal{F}^{\infty}, \mu_{\mathcal{I}'})$-RRFC and $(\mathcal{I}', \mathbb{D}\mathcal{B}_{\mathcal{I}'}, d, \mathbb{D}\mathcal{W}^{\infty}, \epsilon_{\mathcal{I}'})$-RPCUF and it holds that* $\max_{\mathcal{I}'}\{\mu_{\mathcal{I}'} + (1-\mu_{\mathcal{I}'})\epsilon_{\mathcal{I}'}\} \leq \gamma$.

Under this definition, we can arbitrarily compose any RCC_{KF} gadgets as long as each output of a gadget is only used once as input to another gadget. This restriction comes from the composition under probes (cf. Section 4.1). Then, under the assumption of mutually independent fault and probing distributions per gadget, the combined failure probability increases linearly in the number of gadgets in the circuit.

Theorem 4. *Let C be a circuit constructed by composition of gadgets G that are $(d, k, \mathbb{D}\mathcal{W}_i^{\infty}, \mathbb{D}\mathcal{F}_i^{\infty}, \gamma_i)$-$RCC_{KF}$, for $i \in \{1, \ldots, |C|\}$, such that each output of G_i is used as input of at most one other gadget G_j or as output of C. Then, C is*

$$(\prod_{i=1}^{|C|} \mathbb{D}\mathcal{W}_i^{\infty}, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i^{\infty}, 1 - \prod_{i=1}^{|C|}(1-\gamma_i))\text{-}RCS_{KF}.$$

Intuitively, the proof follows the lines of the compositional statements for standalone probing and faulting. Specifically, we first go from inputs to outputs through the circuit to construct the respective tuples of potentially faulty inputs to each gadget to show composition under faults. Then, we go backward, i.e., from outputs to inputs, through the gadgets and use the fact that the set of faulty inputs is bounded in case of a fault combination that can be corrected to show random probing security under faults.

Proof. Let the faulty circuit be $C^{\tilde{\mathcal{F}}} \leftarrow \texttt{AssignFaultGates}(C, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i^{\infty})$, with $\tilde{\mathcal{F}}$ the set of faults selected with $\mathsf{Pr}_{\prod_i \mathbb{D}\mathcal{F}_i^{\infty}}[\tilde{\mathcal{F}}]$. We can divide $\tilde{\mathcal{F}}$ into $|C|$ disjoint fault sets $\tilde{\mathcal{F}}_i \subseteq \tilde{\mathcal{F}}$ such that $\tilde{\mathcal{F}}_i$ belongs to G_i and is selected with $\mathsf{Pr}_{\mathbb{D}\mathcal{F}_i^\infty}[\tilde{\mathcal{F}}_i]$. Further, let $\tilde{\mathcal{W}}$ be the set of leaking wires of C selected with $\mathsf{Pr}_{\prod_i \mathbb{D}\mathcal{W}_i^\infty}[\tilde{\mathcal{W}}]$. We can divide $\tilde{\mathcal{W}}$ into $|C|$ disjoint parts $\tilde{\mathcal{W}}_i \subseteq \tilde{\mathcal{W}}$, each belonging to the gadget G_i such that each $\tilde{\mathcal{W}}_i$ was selected with $\mathsf{Pr}_{\mathbb{D}\mathcal{W}_i^\infty}[\tilde{\mathcal{W}}_i]$.

Since each gadget G_i is $(\mathcal{I}', k, \mathbb{D}\mathcal{F}^\infty, \mu_{\mathcal{I}'})$-RRFC for all tuples $\mathcal{I}'$ it follows with Theorem 3 that the circuit random fault secure with

$$\mu = 1 - \prod_{i=1}^{|C|}(1 - \mu_i)\,. \tag{2}$$

Let $\mathcal{B} = \{\tilde{\mathcal{F}} \mid \tilde{\mathcal{F}} = \bigcup_{i=1}^{|C|} \tilde{\mathcal{F}}_i, \forall \mathcal{I}'_{G_i} : \mathcal{L}_{\mathcal{I}'_{G_i},\tilde{\mathcal{F}}_i}(G_i) = 0\}$ be the set of faults considered secure in Theorem 3, i.e., all fault combinations that can be corrected at the output of the respective gadgets. We denote by

$$\mathcal{B}_{\mathcal{I}'_{G_i}} = \{\tilde{\mathcal{F}}_i \in \mathcal{F}^\infty \mid \mathcal{L}_{\mathcal{I}'_{G_i},\tilde{\mathcal{F}}_i}(G_i) = 0\} \subseteq \mathcal{B}$$

the set of fault combinations that can be corrected in a gadget G_i under faults in the input indices in $\mathcal{I}'_{G_i}$, with $\forall j : |\mathcal{I}'_{G_i,j}| \leq k$.

We now go backward through the circuit, starting with gadgets connected to the outputs of C, considering only faults $\tilde{\mathcal{F}} \in \mathcal{B}$. Let G_i be a gadget only connected to outputs of C. By RPCUF of G_i, we can construct a simulator Sim_{G_i} that requires a subset of inputs (defined by the tuple $\mathcal{I}_{G_i}$) for the simulation of the wires in $\tilde{\mathcal{W}}_i$ under the faults $\tilde{\mathcal{F}}_i$ and with faulty inputs with indices in $\mathcal{I}'_{G_i}$. Let the failure probability of Sim_{G_i} be ϵ_i.

We continue with the parent gadgets, i.e., gadgets that have outputs connected to the inputs of just handled gadgets and (potentially) outputs of the circuit. Let G_j be such a gadget. By RPCUF of G_j, we can construct a simulator Sim_{G_j} that requires a subset of inputs (defined by the tuple $\mathcal{I}_{G_j}$) for the simulation of the wires in $\tilde{\mathcal{W}}_j$ and the output wires defined by the tuples $\mathcal{I}_{G_i}$ of the child gadgets G_i under the faults $\tilde{\mathcal{F}}_j$ and with faulty inputs with indices in $\mathcal{I}'_{G_j}$. In particular, each gadget output with index ℓ is only used once in the circuit and we use the corresponding set $\mathcal{I}_{G_i,\ell'}$ of the child gadget as $\mathcal{O}_{G_j,\ell}$. Again, we denote the failure probability of Sim_{G_i} by ϵ_j. We repeat this process until we reach the inputs of C.

Given this, we can construct the simulator Sim for C by composition of the gadget simulators Sim_{G_i}. The failure probability ϵ of Sim is the probability that at least one simulators Sim_{G_i} fails, i.e.,

$$\epsilon = 1 - \prod_{i=1}^{|C|}(1 - \epsilon_i)\,. \tag{3}$$

Therefore, with Equation 2 and 3 we have

$$\begin{aligned}\mu + (1-\mu)\epsilon &= 1 - \prod_{i=1}^{|C|}(1-\mu_i) + \prod_{i=1}^{|C|}(1-\mu_i)(1-\prod_{i=1}^{|C|}(1-\epsilon_i)) \\ &= 1 - \prod_{i=1}^{|C|}(1-\mu_i)(1-\epsilon_i) \\ &\leq 1 - \prod_{i=1}^{|C|}(1-\gamma_i),\end{aligned}$$

with $\gamma_i = \mu_i + (1-\mu_i)\epsilon_i$ being the combined failure probability of each gadget G_i. It follows $(\prod_{i=1}^{|C|} \mathbb{D}\mathcal{W}_i^{\infty}, \prod_{i=1}^{|C|} \mathbb{D}\mathcal{F}_i, 1 - \prod_{i=1}^{|C|}(1-\gamma_i))$-RCS$_{\text{KF}}$ of C. □

Similar to Theorem 3, the above composition loses some tightness by only considering fault combinations that can be corrected after each gadget. While the set $\mathcal{B}$ gets bigger, and hence ϵ increases, when considering all fault combinations that can be corrected at the output of C, the overall γ gets smaller. The reason is that the respective fault combination is entirely captured in μ of the above argument, however, it would only be partially captured (multiplied by some $\epsilon \leq 1$) when considering the tighter definition of $\mathcal{B}$. Hence, the provided γ is indeed a upper bound of the combined failure probability.

Relation to Combined-Secure Composition. As with RFC, to the best of our knowledge, this is the first work discussing compositional properties in the context of the random combined model. However, again in the threshold model, several compositional notions were discussed. Most of them are a combination of compositional notions for probing and faulting, respectively. In this respect, (Strong) Non-Interference Non-Accumulation ((S)NINA) [20] (later refined to Combined (Strong) Non-Interference (C-(S)NI) [37]) is the combination of (S)NA with (S)NI, and Combined-Isolating Non-Interference (CINI) [27] is the combination of FINI and PINI, respectively. In the context of polynomial masking, Berndt et *al.* [12] coined the notion of fault-resilient (S)NI, which is the usual (S)NI notion that is invariant to fault injection. Our proposed compositional notation has some similarities with (S)NINA/C-(S)NI in that it is a combination of stand-alone faulting and probing composition and the underlying stand-alone notions relate to the respective stand-alone notions of (S)NA and (S)NI.

An interesting direction for future research into the composition in the random combined model is the notion of expansion (as considered in the random probing model) and the investigation of gadgets where the random probing security is invariant to faults.

5 Automatic Verification of Protected Implementations

We implement the verification of the new notion of random combined security (and its composability variants) for cryptographic circuits by extending the verification tools VERICA [37] and IronMask [11]. At the state of the art, VERICA can be employed for combined hardware security verification considering the glitch-extended probing model [24] and the zeta fault-injection model [39], while IronMask is able to verify various (random, glitch-extended) probing security notions in an efficient way by relying on an algebraic characterization for specific gadgets.

In our work, we first aim to establish a common foundation for the practical verification of circuits under the new security models. Subsequently, we provide detailed insights into our extensions, denoted as VERICA$^+$ and IronMask$^+$, enabling the verification of combined security properties in cryptographic circuits.

5.1 Verification of the Generalized Security Models

We provide explicit formulas and practical verification choices for the computation of the simulation and correction failure probabilities in the general random probing model, the general random fault model, and the known-fault random combined security. These then serve as a basis for the extensions VERICA$^+$ and IronMask$^+$.

General Random Probing Security. For the general random probing security (cf. Definition 1), we now assume independent and different leakage probabilities p_w for each wire w in $\mathcal{W}$, to ease the computation for the verification. Then, the probability of a leaking wire combination $\tilde{\mathcal{W}}$ is the product of leakage probabilities of each wire $w \in \tilde{\mathcal{W}}$ times the product of $(1 - p_{w'})$ of the remaining wires $w' \in \mathcal{W} \setminus \tilde{\mathcal{W}}$. Consequently, let ϵ be the simulation failure probability such that

$$\epsilon = \sum_{i=1}^{|\mathcal{W}|} \sum_{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty}_{\#i}} \prod_{w \in \tilde{\mathcal{W}}} p_w \prod_{w' \in \mathcal{W} \setminus \tilde{\mathcal{W}}} (1 - p_{w'}) \,, \tag{4}$$

where $\mathcal{W}^{\infty}_{\#i} \subseteq \mathcal{W}^{\infty}$ is the set of wire combinations of exactly i wires that lead to a simulation failure, i.e., $\tilde{\mathcal{W}}^{\infty}_{\#i} = \{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty} \mid |\tilde{\mathcal{W}}| = i \text{ and } \mathsf{Sim}(C, \tilde{\mathcal{W}}) = \perp\}$. The above equation generalizes the computation of the simulation failure probability in the random probing security from Belaïd et *al.* [10]. Precisely, Equation 4 is only equivalent to the latter if we consider that all wires are leaked with the same probability p.

Practically, a circuit may be too large for exhaustively checking all wire combinations. Therefore, VERICA$^+$ and IronMask$^+$ compute the outer sum of Equation 4 only up to a threshold α, i.e., for $i \in [1, \alpha]$. For any combinations of more than α wires, we can either consider that (1) they do not result in a simulation failure ($\forall i \in \,]\alpha, |\mathcal{W}|] \,, \mathcal{W}^{\infty}_{\#i} = \emptyset$) to obtain a lower bound $\epsilon_{\min}$, or (2) they all lead to a simulation failure ($\forall i \in \,]\alpha, |\mathcal{W}|]\,, \tilde{\mathcal{W}}^{\infty}_{\#i} = \{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty} \mid |\tilde{\mathcal{W}}| = i\}$) to obtain an upper bound $\epsilon_{\max}$. Those lower and upper bound computations follow the same method used for the random probing security by Belaïd et *al.* in [10].

General Random Fault Security. For the general random fault model (cf. Definition 2), again for usability purposes of the verification process, we now assume independent and different fault probabilities q_f for each fault $f = (g, \tau)$. The probability of a fault combination $\tilde{\mathcal{F}}$ is thus the product of the individual fault probabilities of the said combination times the product of $(1 - q_{f'})$ for all faults f' not present in it. Let μ be the correction failure probability defined by

$$\mu = \sum_{i=1}^{|\mathcal{F}|} \sum_{\tilde{\mathcal{F}} \in \mathcal{F}^{\infty}_{\#i}} \prod_{f \in \tilde{\mathcal{F}}} q_f \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - q_{f'})\,, \tag{5}$$

where $\mathcal{F}^{\infty}_{\#i} \subseteq \mathcal{F}^{\infty}$ is the set of fault combinations of exactly i faults that cannot be corrected (such that $\tilde{\mathcal{F}} \in \mathcal{F}^{\infty}_{\#i}$ iff $\mathcal{L}_{\tilde{\mathcal{F}}}(C) = 1$).

For practical reasons, Equation 5 is computed for up to β faults during security verification; the outer sum thus reduces to $\sum_{i=1}^{\beta}$. Similarly to the previous models, we derive the lower bound $\mu_{\min}$ and the upper bound $\mu_{\max}$ of μ by assuming that any combination of more than β faults can be, respectively, corrected or not.

Known-Fault Random Combined Security. For simplicity, we restrict ourselves to $\mathrm{RCS}_{\mathrm{KF}}$ (cf. Definition 4) in the practical tool implementations. We then rely on the discussed security reduction for $\mathrm{RCS}_{\mathrm{UF}}$ (cf. Section 3.3) and leave room for a tighter $\mathrm{RCS}_{\mathrm{UF}}$ security analysis for future work. That said, three parameters are reported for $\mathrm{RCS}_{\mathrm{KF}}$:

1. the correction failure probability μ as described in Equation 5,
2. the known-fault simulation failure probability ϵ_{kf} if fault combinations can be corrected, such that

$$\epsilon_{kf} = \frac{1}{1-\mu} \sum_{i=1}^{|\mathcal{F}|} \sum_{\tilde{\mathcal{F}} \in \mathcal{B}_{\#i}} \epsilon^{\tilde{\mathcal{F}}}_{kc} \prod_{f \in \tilde{\mathcal{F}}} q_f \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - q_{f'})\,, \tag{6}$$

   where $\mathcal{B}_{\#i}$ is the set of fault combinations of i faults that can be corrected and $\epsilon^{\tilde{\mathcal{F}}}_{kc}$ is the known-corrected-fault simulation failure probability (Equation 4). The above formula is obtained by observing that for any $\tilde{\mathcal{F}} \in \mathcal{B}$, we have $\Pr_{\mathbb{D}_{\mathcal{B}}}[\tilde{\mathcal{F}}] = \frac{1}{1-\mu} \Pr_{\mathbb{D}_{\mathcal{F}^{\infty}}}[\tilde{\mathcal{F}}] = \frac{1}{1-\mu} \prod_{f \in \tilde{\mathcal{F}}} q_f \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - q_{f'})$.
3. and the advantage of the combined adversary γ_{kc}:

$$\gamma_{kc} = \mu + \epsilon_{kf} \cdot (1 - \mu)\,. \tag{7}$$

Again, due to practical limitations, we compute the lower and upper bounds of μ up to β faults and the lower and upper bounds of $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ up to α wires, if fault combinations can be corrected. Then, the lower and upper bounds of ϵ_{kf} are derived from the ones of $\epsilon_{kc}^{\tilde{\mathcal{F}}}$. Note, however, that those bounds on $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ are related to the upper bound of μ. The reason is, that we cannot compute the respective $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ for $\tilde{\mathcal{F}} \in \mathcal{B}_{\#i}$ with $i > \beta$, since this would mean to iterate over all faulty circuits with more than β faults. Then, the lower and upper bounds of γ_{kc} are computed from the corresponding bounds of both ϵ_{kf} and μ. The respective bounds of those three parameters are then returned.

Compositional Notions. We provide algorithms in Appendix A to describe how to perform the security verification of the compositional notions from Section 4, which have been implemented in the extensions of IronMask and VERICA.

5.2 Extension of VERICA Verification

In this section, we first discuss the implementation of the (general) random probing model in VERICA$^+$, with an optimization to count leaking wire combinations that fails simulations, resulting in a tighter lower bound for the simulation failure probability. Then, we present additional considerations for the computation of the fault probability to assess the general random fault and known-fault random combined security in VERICA$^+$.

A Tighter Lower Bound in the (General) Random Probing Model. We first present a novel approach to compute a tighter lower bound of the simulation failure probability in the random and general random probing models, than the one mentioned in Section 5.1 and in [10]. Concurrently, VERICA$^+$ provides an upper bound of the simulation failure probability in both models, which relies on this new method and is described in Section B.3.

The core insight is that adding wires to a leaking wire combination that fails simulation $\tilde{\mathcal{W}}_{\text{fail}}$ will result in another simulation failure. The idea is thus to derive the set of larger wire combinations containing this leaking wire combination that is found to fail simulation by VERICA$^+$. In addition, we want to build this set without including redundant combinations or using additional memory[5], in a deterministic way.

More precisely, VERICA$^+$ considers only the intermediate wires of a given circuit C, and labels them in topological order, i.e., the wire w_1 is close to an input of C and $w_{|\mathcal{W}|}$ to an output. However, finding the sets $\mathcal{W}_{\#i}^{\infty}$ from Equation 4 is infeasible for a large number of wires i. Contrary to the naive breadth-first approach that is implemented in the original version of VERICA [37], the idea now is to find failing leaking wire combinations $\tilde{\mathcal{W}}_{\text{fail}}$ in a depth-first manner for up to α wires, $\alpha \in [1, |\mathcal{W}|]$, and deriving the larger wire combinations containing them (up to $|\mathcal{W}|$ wires), by incrementally walking through a binomial tree $B_{|\mathcal{W}|}$.

[5] It is not desirable to store all the found wire combinations that lead to a simulation failure, due to the computational blow-up.

Definition 10 (Binomial tree [17]). *Let $l \in \mathbb{N}$ be an index, $l > 0$. A binomial tree is defined recursively as an ordered tree such that a binomial tree of order* 0 *is a single node, and a binomial tree B_l of order l has a root node whose children are root nodes of binomial (sub)trees of descending orders from* $(l-1)$ *to* 1.

Definition 10 is illustrated in Figure 5 in Appendix B.1. For VERICA$^+$, we define the binomial tree $B_{|\mathcal{W}|}$ of order $|\mathcal{W}|$ that has a height $|\mathcal{W}|$ and $2^{|\mathcal{W}|}$ nodes. By definition, the number of nodes at depth l equals the binomial coefficient $\binom{|\mathcal{W}|}{l}$. The nodes of $B_{|\mathcal{W}|}$ are labeled by the collection of indices of the wires in $\mathcal{W}$, i.e., each label represents a unique wire $w \in \mathcal{W}$, except the root node. At depth l, the nodes sharing a parent are labeled in ascending order from l to $|\mathcal{W}|$, from leftmost to rightmost node.

VERICA$^+$ traverses $B_{|\mathcal{W}|}$ in pre-order fashion[6], from the root to a chosen depth α, which is the threshold number of wires to verify (cf. Section 5.1). During the traversal, if VERICA$^+$ visits the child of a node, it adds the wire labeled by the said child to a set of leaking wires, which represents the currently considered wire combination $\tilde{\mathcal{W}}$. VERICA$^+$ then runs the verification on the latter. If it fails, VERICA$^+$ updates the computation of the lower and upper bound of the simulation failure probability, in the considered security model, as detailed in the following. Directly after, VERICA$^+$ goes back to the parent node by removing the wire indexed by the current node from the set of leaking wires; the children nodes are not visited even if the depth α has not been reached. However, if the simulation does not fail, and the depth α is not attained, the tool continues the procedure on the children nodes. The root of $B_{|\mathcal{W}|}$ represents the empty set $\emptyset$; thus, VERICA$^+$ verifies l wires at depth l. At the end of the tree traversal, the lower and upper bounds of the simulation failure probability are returned.

If the set of currently considered leaking wires fails at a depth level in $[1, \alpha]$, VERICA$^+$ updates the computation of the lower bound of the simulation failure probability $\epsilon_{\min}$ in the general random probing security with

$$\prod_{w \in \tilde{\mathcal{W}}_{\text{fail}}} p_w \cdot \prod_{\substack{w' \in \mathcal{W} \\ w' \notin \tilde{\mathcal{W}}_{\text{fail}} \\ w' \notin \mathcal{W}_{\text{sub}}}} (1 - p_{w'}) \cdot \overbrace{\sum_{k=0}^{|\mathcal{W}_{\text{sub}}|} \sum_{\substack{\tilde{\mathcal{W}} \in \mathcal{W}^{\infty}_{\#k} \\ \tilde{\mathcal{W}} \subseteq \mathcal{W}_{\text{sub}}}} \prod_{w \in \tilde{\mathcal{W}}} p_w \prod_{\substack{w' \in \mathcal{W}_{\text{sub}} \\ w' \notin \tilde{\mathcal{W}}}} (1 - p_{w'})}^{\text{subtree cut (Poisson binomial CDF)} = 1} \tag{8}$$

$$\Leftrightarrow \prod_{w \in \tilde{\mathcal{W}}_{\text{fail}}} p_w \cdot \prod_{\substack{w' \in \mathcal{W} \\ w' \notin \tilde{\mathcal{W}}_{\text{fail}} \\ w' \notin \mathcal{W}_{\text{sub}}}} (1 - p_{w'}),$$

where $\mathcal{W}_{\text{sub}}$ is the set of wires in the children nodes (subtree). Then, VERICA$^+$ continues the walk back to the parent node. It thus ‘cuts’ the subtree of the current node to include it entirely in the computation of $\epsilon_{\min}$ with the Poisson binomial Cumulative Distribution Function (CDF) for a sample of size $|\mathcal{W}_{\text{sub}}|$.

[6] In pre-order traversal, the root of the tree is first visited, then the leftmost subtree is recursively traversed, and afterward, the right subtree up to the rightmost one.

Indeed, with the double summation, we construct all wire combinations from the wires in $\mathcal{W}_{\text{sub}}$ that fail the simulation along with the wires in $\tilde{\mathcal{W}}_{\text{fail}}$. And since the Poisson binomial CDF is evaluated for $|\mathcal{W}_{\text{sub}}|$, it equals one. Therefore, Equation 8 is simplified to the product of wire probabilities in the failing wire combination $\tilde{\mathcal{W}}_{\text{fail}}$ times the product of probabilities of wires that are neither in the failing combination nor in the subtree of the current node.

The same can be calculated for the random probing security by considering each wire to leak with probability p. The exact formula is provided in Appendix B.2. A pseudo-code of the binomial tree traversal with the calculation of the bounds is provided in Algorithm 4 in Appendix B.4.

Example. Figure 2 depicts an example of the binomial tree traversal for $\alpha = 3$ and $|\mathcal{W}| = 4$. In the general random probing model, VERICA$^+$ updates $\epsilon_{\min}$ with $p_{w_1} p_{w_2}$ for the first product of Equation 8, times 1 for the second product (all wires are already included in the failing wire combination or the children nodes).

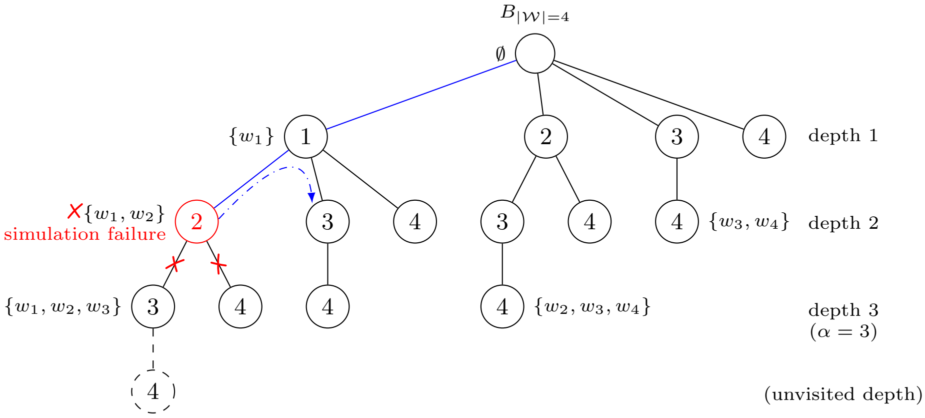


Fig. 2. Example of a binomial tree $B_{|\mathcal{W}|=4}$ to find the exact failing leaking combinations up to $\alpha = 3$ wires (depth 3). The dashed node will not be visited in any case. The blue edges denote the path walked by VERICA$^+$. At the node representing w_2 at depth 2 (in red), the leaking set is $\{w_1, w_2\}$ and fails simulation. Thus, VERICA$^+$ does not visit the children nodes and updates the computation of $\epsilon_{\min}$. It continues by verifying the set $\{w_1, w_3\}$ (blue dashed arrow).

If no subtree can be cut for optimization ($\mathcal{W}_{\text{sub}} = \emptyset$ in Equation 8), the verification is equivalent to iteratively testing all combinations of $1, \ldots, \alpha$ wires. This is only the case if $w_{|\mathcal{W}|}$ leaks or there is no leakage. In general, the more the failing wires or wire combinations are located to the rightmost part of the tree, the less tight the lower bound is or the closer this approach is to the one presented in Section 5.1. That is because the binomial tree is not symmetric. However, if $\alpha = |\mathcal{W}|$, we ensure to find the exact value of the simulation failure probability with this method, since failing wire combination are picked in a deterministic way without redundancy.

Remark 1. The approach in this section is not applicable to injected faults, since adding faults to a non-correctable fault combination does not necessarily result in a failed correction. Thus, it is not used for the general random fault model.

Probability of a Fault Combination. In this section, we explain how VERICA$^+$ exactly computes the probability of a fault combination $\tilde{\mathcal{F}}$ which is required for the evaluation of the correction failure probability μ (cf. Section 5.1), if the said fault combination can not be corrected.

We recall that each fault f is a pair (g, τ), where g is a gate and $\tau : \mathcal{G}_\mathsf{f} \to \mathcal{G}_\mathsf{f}$ is the transformation of the Boolean function encoded by the gate. As explained in Section 5.1, for practicality reasons, VERICA$^+$ considers that each fault f has a probability q_f and the probability of a fault combination is:

$$\Pr[\tilde{\mathcal{F}}] = \prod_{f \in \tilde{\mathcal{F}}} q_f \cdot \prod_{f' \in \mathcal{F} \setminus \tilde{\mathcal{F}}} q_{f'} \,. \tag{9}$$

For each fault, we can further write:

$$q_f = \Pr[\tau \cap g] = \Pr[\tau \mid g] \Pr[g] \,, \tag{10}$$

where $\Pr[g]$ is the probability that a gate $g \in \mathcal{G}$ is targeted and then a specific fault transformation τ occurs with probability $\Pr[\tau \mid g]$.

Furthermore, VERICA$^+$ allows a different number of fault transformations per gate. Let T_g be the set of all possible fault transformations of a specific gate g, which are considered to be equally likely to occur in VERICA$^+$ (Assumption 1). Hence, the probability of a particular fault transformation τ applied to the gate g is:

$$\Pr[\tau \mid g] = |T_g|^{-1} \,,$$

which we substitute into Equation 10 to have $q_f = \Pr[g] \cdot |T_g|^{-1}$ for VERICA$^+$. Moreover, the latter also assumes that up to one fault transformation τ is injected at a gate location, per verification analysis (Assumption 2). As a result, we can rewrite Equation 9 for VERICA$^+$ by

$$\Pr[\tilde{\mathcal{F}}] = \prod_{(g,\tau) \in \tilde{\mathcal{F}}} \frac{\Pr[g]}{|T_g|} \cdot \prod_{(g',\tau') \in \mathcal{F} \setminus \tilde{\mathcal{F}}} (1 - \Pr[g']) \,, \tag{11}$$

where we consider the product of probabilities to inject a fault transformation into some gates times the product of probabilities to not inject any fault into the remaining gates. Therefore, to compute the probability of a fault combination, VERICA$^+$ simply iterates over the gates in the circuit and differentiates the faulted gates from the non-faulted ones.

5.3 Extension of IronMask Verification

In this section, we commence by revisiting the core principles of IronMask. Next, we expound on the extension of IronMask's random probing security verification for C(N)LR gadgets, defined to depict faulty circuits. Following this, we introduce our novel Python script designed to evaluate the random fault security. Finally, we comment on the limitation of our tool regarding general models and its application to the verification of combined models and compositional notions. Henceforth, we simply denote the extended version of IronMask as IronMask$^+$, encompassing the evolutions outlined in this section.

Main Principles of IronMask. In [11], the authors introduce an exact verification procedure of the random probing security of masked gadgets. Their procedure is implemented in a tool referred to as IronMask.[7] This verification tool supports any masking gadgets with linear randomness (i.e., LR-gadgets) and also the so-called NLR-gadgets that are quadratic gadgets which might include non-linear randomness (e.g., by refreshing their inputs). Specifically, an ℓ-to-m NLR-gadget is a gadget G with ℓ s-share inputs and m s-share outputs such that:

$$\begin{aligned} G &: (\mathbb{K}^s)^\ell \to (\mathbb{K}^s)^m \\ (x_1, \ldots, x_\ell) &\mapsto (y_1, \ldots, y_m) = R_{\ell+1}(F(R_1(x_1, r_1), \ldots, R_\ell(x_l, r_l)), r_{\ell+1}) \end{aligned}$$

where F is any arithmetic circuit which computes a homogeneous multi-linear form, the R_i are linear arithmetic circuits and the r_i are vectors of random values uniformly drawn from a finite field $\mathbb{K}$.

The (N)LR-gadgets are sufficient for capturing most basic operations of masked implementations. However, to achieve combined security, additional intermediate correction blocks, such as majority votes, may be required. While the primary purpose of these correction blocks is to provide fault resistance, they often introduce highly non-linear behavior as a side effect.

Furthermore, when considering faults, they can alter the initial gadget structure. In this work, we focus solely on set or reset faults. Specifically, a gate g might be faulted to output either the constant one or the constant zero, i.e., $\tau_{\mathsf{set}}(g) = \mathsf{one}$, $\tau_{\mathsf{reset}}(g) = \mathsf{zero}$ (see Section 3.2). Nevertheless, these two types of faults do not affect the (N)LR format of the gadgets between the correction blocks.

Following the additional constraints of combined security resistance and the presence of faults, our objective is twofold:

(i) Integrating the verification of more intricate gadgets designed to satisfy a property of combined composability. These gadgets consist of $n = 2k + 1$ replications of an original (N)LR-gadget with intermediate correction blocks, where all the gates may be impacted by set or reset faults. We will refer to these new gadgets as C(N)LR gadgets in the following (with the additional 'C' letter to signify the presence of intermediate correction blocks).

[7] `https:github.com/CryptoExperts/IronMask`

(ii) Not only verifying the random probing composability of input gadgets but also considering faults to verify the random fault security, the known-fault random combined security and their compositional variants.

Extended Verification of Random Probing Security. Below, we introduce the new C(N)LR gadgets, which serve as inputs for our extension tool, IronMask$^+$. Subsequently, we elaborate on their verification process.

New C(N)LR-Gadgets. As previously outlined, IronMask takes the implementation of (N)LR-gadgets as input. Our extension continues to support these gadgets while also accommodating more intricate ones that we refer to as C(N)LR-gadgets. These C(N)LR-gadgets are constructed by parallel replication of (N)LR-gadgets, incorporating intermediate non-linear correction blocks (e.g., implementing the majority function) and whose gates may be modified by set and reset faults.

An example is provided in Figure 3 with the $(1,1)$-CINI gadget from [25], implementing a masked multiplication with two masked inputs $a = (a_0, a_1) \in \mathbb{F}_2^2$ and $b = (b_0, b_1) \in \mathbb{F}_2^2$, $k = 1$ (i.e., 3 copies) and two correction blocks referred to as C_0 and C_1. Copies of the same variable x, that are defined for fault resistance, are denoted x^0, x^1, and x^2. For the sake of clarity, we do not represent copy operations that are mandatory to replicate any variable that is to be used in more than one gate.

Verification of C(N)LR-Gadgets. To date, no technique has been able to efficiently and accurately verify the (random) probing security of highly non-linear circuits. The computation of exact probe distributions quickly becomes prohibitively inefficient, while techniques relying on the analysis of symbolic expressions of variables prove incomplete in the presence of non-linear randomness (with the exception of specialized scenarios like the one developed in IronMask). To approach completeness for a C(N)LR-gadget G, we propose a novel and efficient verification methodology in three steps. This methodology operates under the assumption that each output of a correction block remains correct provided that no fault occurs along its corresponding path within the correction block. Our method comes in three steps described below.

Step 1. Each wire in G is labeled with its symbolic expression made of the outputs of the correction blocks (starting from the last one), the random values, and the gadget inputs. Namely, we introduce symbolic variables $s_{i,j}^{\ell}$ to represent copies of the outputs of each correction block C_i, where i is the correction block index, j the share index and ℓ the replication index. Then, all the subsequent variables are directly expressed according to these $s_{i,j}^{\ell}$ that are not anymore further developed with their symbolic expression. For instance, the first output of the gadget represented in Figure 3 would be labeled $a_0^0 \cdot s_{0,0}^0 \oplus s_{1,0}^0$, with $s_{0,0}^0$ the first copy of the first output of the first correction block C_0 and $s_{1,0}^0$ the first copy of the first output of the second correction block C_1.

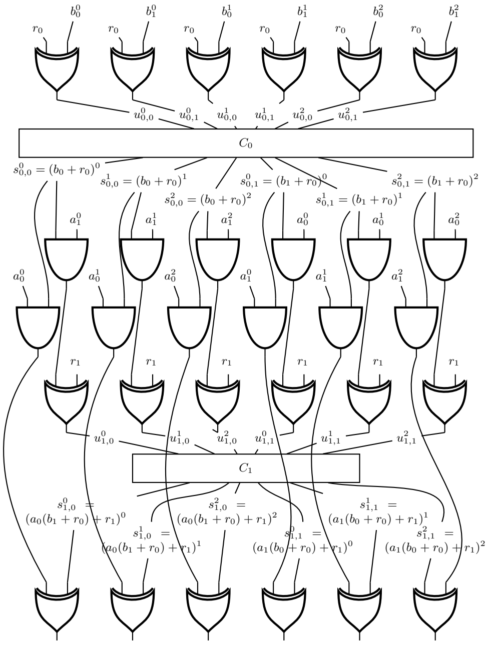

r_0
b_0^0
r_0
b_1^0
r_0
b_0^1
r_0
b_1^1
r_0
b_0^2
r_0
b_1^2
$u_{0,0}^0$
$u_{0,1}^0$
$u_{0,0}^1$
$u_{0,1}^1$
$u_{0,0}^2$
$u_{0,1}^2$
C_0
$s_{0,0}^0 = (b_0 + r_0)^0$
$s_{0,0}^1 = (b_0 + r_0)^1$
$s_{0,0}^2 = (b_0 + r_0)^2$
$s_{0,1}^0 = (b_1 + r_0)^0$
$s_{0,1}^1 = (b_1 + r_0)^1$
$s_{0,1}^2 = (b_1 + r_0)^2$
a_1^0
a_1^1
a_1^2
a_0^0
a_0^1
a_0^2
a_0^0
a_0^1
a_0^2
a_1^0
a_1^1
a_1^2
r_1
r_1
r_1
r_1
r_1
r_1
$u_{1,0}^0$
$u_{1,0}^1$
$u_{1,0}^2$
$u_{1,1}^0$
$u_{1,1}^1$
$u_{1,1}^2$
C_1
$s_{1,0}^0 = (a_0(b_1 + r_0) + r_1)^0$
$s_{1,0}^1 = (a_0(b_1 + r_0) + r_1)^1$
$s_{1,0}^2 = (a_0(b_1 + r_0) + r_1)^2$
$s_{1,1}^0 = (a_1(b_0 + r_0) + r_1)^0$
$s_{1,1}^1 = (a_1(b_0 + r_0) + r_1)^1$
$s_{1,1}^2 = (a_1(b_0 + r_0) + r_1)^2$

Fig. 3. CNLR gadget which implements a masked multiplication with 2-shared inputs, 3 replications and 2 intermediate correction blocks.

Step 2. The gadget G is modified with the following step that is repeated until nothing can be further replaced (at most as many times as the number of correction blocks): the symbolic expression of each wire which depends on at least one output copy $s_{i,j}^{\ell}$ of a correction block, which is not impacted by a fault injected inside the correction block, is modified to replace $s_{i,j}^{\ell}$ by its correct symbolic expression (i.e., an hypothetical input of the correction block for which no fault had occurred in the path). In our example, if there is no fault on the path producing $s_{1,0}^0$ in C_1, the symbolic expression of the first output of G would become $a_0^0 \cdot s_{0,0}^0 \oplus (a_0 \cdot s_{0,1} \oplus r_1)$.

Step 3. All the wires in the updated gadget G' are still labeled with symbolic expression made of the outputs of the correction blocks, the random values, and the gadget inputs. At this step, we can apply IronMask's original verification on each tuple of variables with slight modifications.

Recall that IronMask's original verification operates within two steps. The first step basically performs a Gaussian elimination on the probes that are linear combinations of sub-products and random values that do not appear in the sub-products. Given the structure of (N)LR gadgets, all the random values that appear in these linear combinations are eliminated at this point. The second step then manipulates probes that are either the remaining sub-products or linear combinations of input shares and other random values. After a factorization step, described in [11], to linearize the probe expressions, a second Gaussian elimination is finally performed.

In IronMask$^+$, to consider both the output correction blocks or the intermediate variables of correction blocks (that are impacted by faults, from the first steps), they need to be replaced by the corresponding inputs of their correction block during the two verification steps. Specifically, the Gaussian elimination is performed progressively for each probe in the considered tuple. Once the probe is treated, each output of correction block or intermediate variable in its expression is replaced by the corresponding $2k+1$ inputs of its correction blocks. Similarly, each intermediate variables of a correction block is replaced by the corresponding $2k+1$ inputs of its correction blocks. They are then treated like regular probe in the Gaussian elimination. The exact same scenario happens in the second step of IronMask$^+$'s verification during the second Gaussian elimination procedure. A pseudo-code of this verification with blue highlights on the added replacements is available in Algorithm 5 in Appendix C.

While IronMask provides complete verification without false positives, our new methodology may yield artificial attack paths due to the third item. The first two points of our methodology simply substitute symbolic expressions with simplified expressions of equivalent value to aid in computation of correction steps. However, the third item addresses non-linear terms not conforming to IronMask's format due to specific faults. In such cases, intermediate variables are replaced by all their dependencies, which is a conservative approach. By granting attackers access to sets of variables rather than functions of these variables, our methodology enhances their potential reach. Put differently, if our extended tool IronMask$^+$ determines a tuple as not being a failure, then indeed it is not. Conversely, if it indicates a failure path, there is a possibility of false positives in certain scenarios.

Remark 2. One could easily extend our verification procedure to gadgets with a different format (i.e., not covered by our (C)(N)LR gadgets), or whose authorized faults might impact the non-linearity, by proceeding as in the third point. Namely, as soon as the symbolic expressions of variables within a targeted tuple are not eligible to a complete verification, they can trivially be replaced by all their dependencies for the rest of the verification. Although not tight, this method benefits from capturing all the possible attack paths.

Verification of Random Fault Security. IronMask$^+$ is completed with a Python script to generate all the faulty scenarios for a circuit that can be corrected. It takes as input a circuit description, a single type of faults (*set* or *reset*), a number of faults and a number of tolerated faults (usually linked to the number of replications $2k+1$). The script then evaluates copies of the circuit outputs in SageMath using symbolic computation (Boolean polynomial ring) based on randoms and input shares (where the copy of the input share is considered as the input share, i.e., a_0^0 evaluates to a_0). Then, for a combination of faults, the circuit is re-evaluated in SageMath, and the outputs are compared: if, for any of the output shares, there are strictly more than k faulty copies, it constitutes a faulty scenario that cannot be corrected. For the types of faults, we have either *set* or *reset*, and if the fault is on a copy of an input share, it is treated as a new symbolic variable, i.e., a_0^0 with a fault becomes a new variable in the Boolean polynomial ring, instead of evaluating to a_0 when not faulty. However, for faults on the randoms, the outputs are not compared to the original circuit (golden circuit) but rather to the circuit with the same faults on the randoms (as justified for combined analysis in [37]).

Verification of General and Combined Models. We briefly discuss the possible extension of IronMask$^+$ to general models and its application to verify combined security and compositional notions.

General Models. We should note that we do not verify the general random probing security in IronMask$^+$ for various probabilities. While this is not particularly challenging to implement, it is significantly less efficient, as IronMask$^+$ only computes the cardinals of the leaking tuples based on their size (formerly referred to as $|\mathcal{W}_{\#i}^{\infty}|$ in Equation 4). Similarly, our Python script to verify the random fault security currently applies for faults with the same probability for efficiency reasons. Adapting to the general random fault model would simply require to associate a specific probability to each faulty circuit.

Combined Security. The new verification of the random fault security with the extension to C(N)LR gadgets makes it possible to verify the known-fault random combined security directly from both advantages, as detailed in Section 5.1. The attacker is computed following Equation 7 from the result of Equation 6 with identical fault probabilities.

Compositional Notions. As in previous versions of IronMask, the compositional notions are all based on the verification of the previously detailed properties (random probing security, random fault security and known-fault random combined security). As depicted in Appendix A, the main difference lies in exploring all the input and output behaviors.

6 Evaluation

We selected three different gadgets for our experiments on both tools in the new known-fault random combined model (cf. Definition 4 and Definition 9), namely the $(1,1)$-CPC$_1$ gadget from [25], the flawed $(1,1)$-HPC$_1^C$ gadget from [27], and the $(1,1)$-SININA gadget originally proposed for software in [20] and modified for hardware in [37]. Please note that the latter two gadgets were shown to be insecure under composition [25,37]. The goal of our case studies is to showcase the capabilities and limitations of VERICA$^+$ and IronMask$^+$ and give a qualitative comparison between the gadgets. We specifically do not perform an extensive security analysis of the mentioned gadgets under realistic leakage and fault distributions. The determination of such distributions is left for future work. Note, however, that such an analysis would only differ in the absolute values of the failure probabilities, not in the qualitative expressiveness of the data.

All the tables displaying our results are structured in the same way. The first column defines the evaluated gadget, while the second and third column displays the number of gates (FIA locations) and the number of wires (SCA locations), respectively. The fault transformation that can be performed is indicated by τ, while p represents the leaking random probing probability and q denotes the random fault probability for each gate. Note that faults and leakage on different wires are independent in this setting and the probabilities are the same for each wire (except where we select only a subset of SCA and FIA locations, where some probabilities are set to zero), for efficiency reasons. The remaining columns present our results: μ_{min} and μ_{max} denote the bounds on the advantage of the attacker in the random fault model or for RFC, while ϵ_{min} and ϵ_{max} represent the bounds on the advantage of the attacker in the random probing model or for RPCUF. For the computation, we selected a threshold of $\alpha = 2$ and $\beta = 2$ for the number of probes and faults, respectively. Finally, γ_{min} and γ_{max} represent the global attacker advantage for RCS_{KF} or RCC_{KF}. Whether the bounds are computed for the general security or composition is indicated for each table individually. Please be reminded, that the bounds on ϵ correlate to the bounds on μ_{max}. We first present our results in terms of security parameters and verification timings for both tools individually. Finally, we conduct the same experiments on both tools to validate the correctness and compare timings on specific scenarios.

6.1 Results on VERICA$^+$

All our experiments for VERICA$^+$ have been conducted on a machine equipped with two AMD EPYC 7742 64-Core processors and 512 GB memory. This allowed us to perform all experiments on 256 threads.

The analysis results and performance of VERICA$^+$ for RCC_{KF} of the three gadgets under different scenarios can be found in Table 1. We first observe that, under the same fault and leakage conditions, the $(1, 1)$-CPC_1 gadget is more secure than the other two gadgets. The reason is that this gadget is shown composable in the threshold combined model (CINI [27]), while the other two have shown to be flawed under composition in the threshold combined model [25,37]. The discrepancy between the $(1, 1)$-CPC_1 and the $(1, 1)$-HPC_1^C gadget is mainly caused by the probing factor, while the discrepancy to the $(1, 1)$-SININA gadget is caused nearly entirely by the faulting factor. The reason is, that for the SININA gadget, there exist some input-fault combinations that cannot be corrected properly (even without internal faults), specifically, when there are k faults in all the inputs. Those input-fault combinations are always considered insecure and, hence, dominate the failure probability for RFC because we determine the maximum over the input faults. This also explains why some of the ϵ bounds for the SININA are zero. Since those bounds are related to μ_{max}, which is one, there is no side-channel evaluation performed. In this case, any internal faults

Table 1. $\mathrm{RCC}_{\mathrm{KF}}$ analysis of three gadgets with different fault and leakage probabilities on VERICA^{+}. Here, all faults and wire leakages happen with the same indicated probability.

Design	#Loc.		Model			Probabilities			Time
	FIA	SCA	τ	q	p	$\mu_{\min}/\mu_{\max}$	$\epsilon_{\min}/\epsilon_{\max}$	$\gamma_{\min}/\gamma_{\max}$	t
$(1,1)$-CPC_1 [25]	174	98	*set*	2^{-10}	2^{-10}	5.13e-2/5.14e-2	8.19e-2/8.21e-2	1.29e-1/1.29e-1	53.3 min
			set	2^{-10}	2^{-15}	5.31e-2/5.33e-2	2.57e-3/2.57e-3	5.56e-2/5.57e-2	53.3 min
			set	2^{-15}	2^{-10}	1.52e-3/1.52e-3	8.43e-2/8.44e-2	8.57e-2/8.58e-2	52.6 min
			set	2^{-15}	2^{-15}	1.65e-3/1.65e-3	2.65e-3/2.65e-3	4.29e-3/4.29e-3	52.6 min
			reset	2^{-10}	2^{-10}	6.78e-2/6.79e-2	8.17e-2/8.19e-2	1.44e-1/1.44e-1	40.9 min
			reset	2^{-10}	2^{-15}	6.78e-2/6.79e-2	2.66e-3/2.66e-3	7.03e-2/7.04e-2	41.3 min
			reset	2^{-15}	2^{-10}	2.04e-3/2.04e-3	8.42e-2/8.44e-2	8.61e-2/8.63e-2	40.9 min
			reset	2^{-15}	2^{-15}	2.19e-3/2.19e-3	2.65e-3/2.65e-3	4.84e-3/4.84e-3	40.8 min
			flip	2^{-10}	2^{-10}	6.96e-2/6.97e-2	8.18e-2/8.20e-2	1.46e-1/1.46e-1	38.8 min
			flip	2^{-10}	2^{-15}	6.96e-2/6.97e-2	2.66e-3/2.66e-3	7.20e-2/7.22e-2	38.5 min
			flip	2^{-15}	2^{-10}	2.10e-3/2.10e-3	8.43e-2/8.44e-2	8.62e-2/8.64e-2	38.6 min
			flip	2^{-15}	2^{-15}	2.26e-3/2.26e-3	2.65e-3/2.65e-3	4.90e-3/4.90e-3	38.5 min
$(1,1)$-HPC_1^C [27]	180	104	*set*	2^{-10}	2^{-10}	5.68e-2/5.69e-2	1.61e-1/1.61e-1	2.09e-1/2.09e-1	46.6 min
			set	2^{-10}	2^{-15}	5.68e-2/5.69e-2	5.48e-3/5.48e-3	6.20e-2/6.21e-2	47.0 min
			set	2^{-15}	2^{-10}	1.83e-3/1.83e-3	1.61e-1/1.61e-1	1.63e-1/1.63e-1	46.5 min
			set	2^{-15}	2^{-15}	1.83e-3/1.83e-3	5.48e-3/5.48e-3	7.30e-3/7.30e-3	46.4 min
			reset	2^{-10}	2^{-10}	6.41e-2/6.43e-2	1.61e-1/1.61e-1	2.15e-1/2.15e-1	40.5 min
			reset	2^{-10}	2^{-15}	6.41e-2/6.43e-2	5.48e-3/5.48e-3	6.93e-2/6.94e-2	40.5 min
			reset	2^{-15}	2^{-10}	2.07e-3/2.07e-3	1.61e-1/1.61e-1	1.63e-1/1.63e-1	41.0 min
			reset	2^{-15}	2^{-15}	2.07e-3/2.07e-3	5.48e-3/5.48e-3	7.54e-3/7.54e-3	40.6 min
			flip	2^{-10}	2^{-10}	6.41e-2/6.42e-2	1.61e-1/1.61e-1	2.15e-1/2.15e-1	39.1 min
			flip	2^{-10}	2^{-15}	6.41e-2/6.42e-2	5.48e-3/5.48e-3	6.92e-2/6.93e-2	39.1 min
			flip	2^{-15}	2^{-10}	2.07e-3/2.07e-3	1.61e-1/1.61e-1	1.63e-1/1.63e-1	39.0 min
			flip	2^{-15}	2^{-15}	2.07e-3/2.07e-3	5.48e-3/5.48e-3	7.54e-3/7.54e-3	39.0 min
$(1,1)$-SININA [37,20]	198	122	*set*	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	46.0 min
			set	2^{-10}	2^{-15}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	46.2 min
			set	2^{-15}	2^{-10}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	46.1 min
			set	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	46.1 min
			reset	2^{-10}	2^{-10}	9.99e-1/9.99e-1	2.99e-4/4.52e-2	9.99e-1/9.99e-1	48.2 min
			reset	2^{-10}	2^{-15}	9.99e-1/9.99e-1	8.99e-6/1.34e-3	9.99e-1/9.99e-1	47.7 min
			reset	2^{-15}	2^{-10}	1.00e-0/1.00e-0	8.11e-3/4.52e-2	1.00e-0/1.00e-0	47.6 min
			reset	2^{-15}	2^{-15}	1.00e-0/1.00e-0	2.44e-4/1.34e-3	1.00e-0/1.00e-0	48.1 min
			flip	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	45.0 min
			flip	2^{-10}	2^{-15}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	45.0 min
			flip	2^{-15}	2^{-10}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	44.9 min
			flip	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	44.9 min

can at most make the gadget more secure (by compensating the impact of input faults) and hence, the respective μ gets larger with smaller fault probability q. Another interesting observation is that for the $(1,1)$-CPC_1 gadget the bounds for μ are not consistent when keeping q constant. This is because we determine the maximum $\gamma_{\min}$ and report the respective bounds for ϵ and μ. Since the ϵ changes with p the resulting maximum in $\gamma_{\min}$ can change as well, meaning that we report different bounds for μ. We would like to reiterate that the selected fault and leakage distributions are not realistic and, hence, the results in Table 1 do not reflect the absolute vulnerability of the gadgets (except where input faults cause the fault failing probability to be one).

Table 2. $\mathrm{RCS}_{\mathrm{KF}}$ analysis of the $(1,1)$-CPC_1 gadget [25], with computation threshold for probes $\alpha = 2$ and for faults $\beta = 2$, where only a subset of gates and wires are faulted and potentially leaked to the adversary.

	#Loc.		**Model**			**Probabilities**			**Time**
Location	FIA	SCA	τ	q	p	$\mu_{\min}/\mu_{\max}$	$\epsilon_{\min}/\epsilon_{\max}$	$\gamma_{\min}/\gamma_{\max}$	t
all; all	98	174	*reset*	2^{-10}	2^{-10}	7.40e-4/8.72e-4	2.44e-3/3.00e-3	3.18e-3/3.87e-3	13.6 s
reg; rep. index 0	18	106	*reset*	2^{-10}	2^{-10}	5.07e-5/5.15e-5	8.89e-4/1.03e-3	9.39e-4/1.08e-3	0.8 s
reg; cone of c_0^0	18	103	*reset*	2^{-10}	2^{-10}	5.07e-5/5.15e-5	7.50e-4/8.76e-4	8.00e-4/9.27e-4	0.8 s

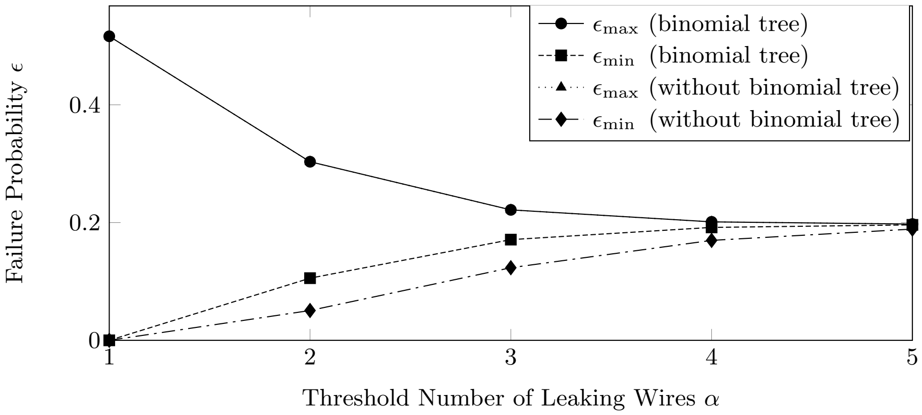


Fig. 4. Comparison of the computation of the lower and upper bounds of the simulation failure probability ϵ in the random probing model ($p = 0.010$), computed with and without the optimization based on the binomial trees, for a $(1,1)$-CPC_1 gadget.

With respect to performance, we can see that VERICA$^+$ can estimate the failure probabilities, by enumerating all fault and probe combinations with up to two faults and probes, for all three gadgets in less than one hour. This is only possible because of the massive parallelization exploited by VERICA$^+$. In comparison, the evaluation of $\mathrm{RCS}_{\mathrm{KF}}$ is much faster. As can be seen in Table 2, the $(1,1)$-CPC_1 gadget can be analyzed within seconds. This speedup is possible because $\mathrm{RCC}_{\mathrm{KF}}$ needs to perform essentially the same analysis as for $\mathrm{RCS}_{\mathrm{KF}}$, however, for all input-fault and output-probe combinations.

In addition, we showcase the ability of VERICA$^+$ to analyze a design under different fault and leakage probabilities (where each location is faulted or leaks with independent probabilities). Specifically, we analyze the $(1,1)$-CPC_1 gadget [25] where we restrict the fault locations to registers only and the location of leaking wires to either all wires of the first replication (*rep. index 0*) or all wires that influence the output c_0^0 (*cone of* c_0^0). This could resemble an adversary that uses clock glitching for fault injection and an EM probe for side-channel analysis. Then, the two cases can be seen as representations of two possible layouts of the chip design, such that either all values of one replication index or all the values that influence the same output value are placed in close proximity. To give some context, we also provide in Table 2 the results of an analysis where all gates are faulted and all wires can leak. As can be seen, this allows a tighter analysis for specific attack scenarios, potentially enabling a reduction of the implementation cost if the attacker can be restricted in the possible fault and leakage distributions.

Finally, we present a case study where we compare the computation of the lower and upper bounds of the simulation failure probability for the random probing security with and without the optimization based on binomial trees (cf. Section 5), for the $(1,1)$-CPC$_1$ gadget, in Figure 4. Each wire is assumed to leak with the same leaking wire probability $p = 0.010$, and no fault is injected. Note, that we choose a higher leakage probability to get more pronounced differences between the upper and lower bounds. The bounds of the simulation failure probability are evaluated for combinations of $\alpha = 1, \ldots, 5$ wires for a more in-depth analysis.

We observe that the two methods provide similar probabilities for the upper bound $\epsilon_{\max}$ (appearing as a unique solid line). However, the binomial tree approach gives a higher value for the lower bound $\epsilon_{\min}$ than without. Hence, we see that the optimization provides a tighter lower bound and converges faster to the actual value of the simulation failure probability ϵ.

6.2 Results on IronMask$^+$

Contrary to VERICA$^+$, all our experiments for IronMask$^+$ have been conducted on a single thread, on a machine equipped with an AMD Ryzen Threadripper PRO 7995WX processor with 96 cores (192 threads) and 512GB of RAM. The analysis outcomes and performance results of IronMask$^+$ for RCC$_{\mathrm{KF}}$ across the three gadgets under various scenarios are detailed in Table 3.

In this extension, we exclusively focused on *set* and *reset* faults, deferring the inclusion of other fault types for future work. As mentioned earlier, the chosen fault and leakage distributions are not realistic, thus the results depicted in Table 3 do not accurately represent the absolute vulnerability of the selected gadgets, although they follow the same trends as for VERICA$^+$. Moreover, it is worth noting that expanding the evaluation to encompass larger tuples would refine the reported ranges. Nonetheless, it is notable that, as anticipated, IronMask$^+$ performs quite efficiently (especially given its execution on a single thread), thanks to its evaluation of symbolic variables with efficient rules, which, on the other hand, allows for some tolerance of false positives, as detailed in Section 5.

Regarding performance, the four computations involving different probabilities under identical settings (gadget, fault type) can be partially generated concurrently. For example, when considering the $(1,1)$-SININA gadget for *set* faults as introduced in [37,20], the selection of faulty circuits that can be corrected takes 18 minutes and 38 seconds. Subsequently, identifying failure tuples within these selected circuits requires 1 hour, 54 minutes, and 8 seconds. Both of these operations are common to any combination of faults and probe probabilities. Finally, computing security advantages based on the probabilities of faults and probes requires between 63 and 66 minutes.

Table 3. $\mathrm{RCC}_{\mathrm{KF}}$ analysis of three gadgets with different fault and leakage probabilities on IronMask$^+$. Here, all faults and wire leakages happen with the same indicated probability.

	#Loc.		**Model**			**Probabilities**			**Time**
Design	FIA	SCA	τ	q	p	$\mu_{\min}/\mu_{\max}$	$\epsilon_{\min}/\epsilon_{\max}$	$\gamma_{\min}/\gamma_{\max}$	t
$(1,1)$-CPC_1 [25]	182	316	*set*	2^{-10}	2^{-10}	8.89e-2/8.95e-2	1.62e-1/1.66e-1	2.37e-1/2.41e-1	21h 42min 38s
			set	2^{-10}	2^{-15}	8.89e-2/8.95e-2	8.47e-3/8.48e-3	9.66e-2/9.73e-2	21h 42min 19s
			set	2^{-15}	2^{-10}	2.56e-3/2.56e-3	1.61e-1/1.65e-1	1.63e-1/1.67e-1	21h 41min 46s
			set	2^{-15}	2^{-15}	2.93e-3/2.93e-3	5.66e-3/5.66e-3	8.57e-3/8.57e-3	21h 41min 48s
			reset	2^{-10}	2^{-10}	8.89e-2/8.96e-2	1.62e-1/1.65e-1	2.36e-1/2.40e-1	20h 33min 24s
			reset	2^{-10}	2^{-15}	8.89e-2/8.96e-2	7.50e-3/7.51e-3	9.58e-2/9.64e-2	20h 33min 21s
			reset	2^{-15}	2^{-10}	2.59e-3/2.59e-3	1.61e-1/1.65e-1	1.63e-1/1.67e-1	20h 33min 35s
			reset	2^{-15}	2^{-15}	2.93e-3/2.93e-3	5.63e-3/5.63e-3	8.54e-3/8.54e-3	20h 33min 15s
$(1,1)$-HPC_1^C [27]	188	322	*set*	2^{-10}	2^{-10}	3.68e-2/3.76e-2	9.93e-1/9.97e-1	9.93e-1/9.97e-1	23h 54min 38s
			set	2^{-10}	2^{-15}	3.68e-2/3.76e-2	9.96e-1/9.97e-1	9.96e-1/9.97e-1	23h 55min 04s
			set	2^{-15}	2^{-10}	1.16e-3/1.16e-3	9.96e-1/1.00e-0	9.96e-1/1.00e-0	23h 54min 58s
			set	2^{-15}	2^{-15}	1.16e-3/1.16e-3	1.00e-0/1.00e-0	1.00e-0/1.00e-0	23h 55min 18s
			reset	2^{-10}	2^{-10}	8.53e-2/8.60e-2	9.95e-1/1.00e-0	9.96e-1/1.00e-0	23h 32min 20s
			reset	2^{-10}	2^{-15}	8.17e-2/8.24e-2	9.99e-1/1.00e-0	9.99e-1/1.00e-0	23h 32min 26s
			reset	2^{-15}	2^{-10}	2.80e-3/2.80e-3	9.96e-1/1.00e-0	9.96e-1/1.00e-0	23h 33min 27s
			reset	2^{-15}	2^{-15}	2.68e-3/2.68e-3	1.00e-0/1.00e-0	1.00e-1/1.00e-0	23h 33min 10s
$(1,1)$-SININA [37,20]	230	364	*set*	2^{-10}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	3h 12min 53s
			set	2^{-10}	2^{-15}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	3h 12min 17s
			set	2^{-15}	2^{-10}	9.99e-1/1.00e-0	0.00e-0/0.00e-0	9.99e-1/1.00e-0	3h 25min 34s
			set	2^{-15}	2^{-15}	1.00e-0/1.00e-0	0.00e-0/0.00e-0	1.00e-0/1.00e-0	3h 33min 52s
			reset	2^{-10}	2^{-10}	9.99e-1/1.00e-0	1.18e-4/1.09e-1	9.99e-1/1.00e-0	3h 21min 59s
			reset	2^{-10}	2^{-15}	9.99e-1/1.00e-0	3.75e-6/3.30e-3	9.99e-1/1.00e-0	3h 18min 32s
			reset	2^{-15}	2^{-10}	1.00e-0/1.00e-0	3.84e-3/1.09e-1	1.00e-0/1.00e-0	3h 13min 28s
			reset	2^{-15}	2^{-15}	1.00e-0/1.00e-0	1.22e-4/3.30e-3	1.00e-0/1.00e-0	3h 13min 49s

6.3 Correctness and Comparison Between Tools

In this section, we aim to compare the results of both tools for two main purposes: verifying their correctness and comparing their verification time on concrete examples. We arbitrarily choose one of the gadgets previously tested, namely $(1,1)$-CPC_1 [25], with *set* and *reset* faults and probabilities $q = 2^{-10}$ and $p = 2^{-10}$. For this specific test, unlike the previous sections, we consider the same design and structure in both tools to have a fair comparison. In particular, VERICA$^+$ usually considers one copy of a wire per usage, while IronMask$^+$ simulates the repeated usage via special copy operations. Consequently, repeated usage of intermediate variables leads to a higher number of wires in IronMask$^+$. To ensure a fair comparison, we artificially eliminate the copies in both tools, assuming that each wire has precisely one copy in the circuit. The results of this comparison are depicted in Table 4.

We can see that the bounds $\mu_{\min}$ and $\mu_{\max}$ are identical for both tools. This consistency arises from the fact that copies are not taken into account for faults, as previously explained. Therefore, the presence or absence of copies has no impact on the values of $\mu_{\min}$ and $\mu_{\max}$. The alignment of these values across both tools instills confidence in the correctness of the algorithms, considering that each tool employs completely different strategies.

Table 4. Comparison of verification results on $(1,1)$-CPC$_1$ [25] with probabilities $q = 2^{-10}$ and $p = 2^{-10}$ for 170 FIA and 176 SCA locations.

Tools	Fault Model	Probabilities			CPU Time
		$\mu_{\text{min}}/\mu_{\text{max}}$	$\epsilon_{\text{min}}/\epsilon_{\text{max}}$	$\gamma_{\text{inf}}/\gamma_{\text{sup}}$	t
VERICA$^+$	*set*	8.89e-2/8.96e-2	8.77e-2/8.85e-2	1.69e-1/1.70e-1	57d 21h 45min 21s
IronMask$^+$		8.89e-2/8.95e-2	9.93e-2/1.00e-1	1.79e-1/1.81e-1	21h 17min 42s
VERICA$^+$	*reset*	8.89e-2/8.96e-2	8.76e-2/8.84e-2	1.69e-1/1.70e-1	63d 7h 0min 16s
IronMask$^+$		8.89e-2/8.96e-2	9.84e-2/9.92e-2	1.79e-1/1.80e-1	20h 44min 20s

However, there are slight differences in ϵ_{min} and ϵ_{max} between the two tools. VERICA$^+$ employs an optimization based on the binomial tree traversal that enables it to achieve tighter lower bounds on the failure probability while maintaining the same upper bound for both tools. Consequently, ϵ_{min} tends to be higher with VERICA$^+$ in both scenarios, resulting in tighter values. Regarding the upper bound on the failure probability, ϵ_{max}, they exhibit very close values, with a small discrepancy where the values computed with IronMask$^+$ are slightly higher. This divergence stems from the computation method for the upper bound, which leverages the previously computed lower bound and assumes that all non-tested tuples of probes result in failures. Since the lower bound computed in IronMask$^+$ is lower, with fewer coefficients actually computed and the remaining ones replaced by zero, the upper bound also shows a slight increase, with the uncomputed coefficients replaced by their upper bounds, which are binomial coefficients. In essence, while the upper bound of VERICA$^+$ is tighter, the discrepancy is minimal given the exponentially large number of non-tested tuples.

Finally, these results illustrate the increased efficiency of IronMask$^+$, which evaluates symbolic variables using efficient rules, compared to VERICA$^+$, which relies on an exhaustive approach to check the probing distribution. Thus, the tools offer two different valuable trade-offs in terms of accuracy versus efficiency.

7 Conclusion

In this work, we extended and refined probabilistic models for physical attacks with general leaking and fault probabilities. In particular, considering the ϵ-random probing, the random fault and their combinations. This allows for precise modeling of low-level physical effects that influence SCA and FIA leakage, by dependent and different probabilities. For the resulting models, we investigated important properties, like the impact of adversarial knowledge and composition, and provided two ways of tool-bases analysis methods that enable the security assessment of design components. In the end, we hope that our model fuels research into the low-level physical effects of leakage and their expressions in probability distributions, which we left for future work.

Acknowledgments. This work was co-funded by the European Union (ERC, AMAskZONE, 101077506). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. In addition, 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 through the project CAVE (510964147), and by the German Federal Ministry of Education and Research BMBF through the project VE-HEP (16KIS1345) and 6GEM (16KISK038).

References

1. Aghaie, A., Moradi, A., Rasoolzadeh, S., Shahmirzadi, A.R., Schellenberg, F., Schneider, T.: Impeccable Circuits. IEEE Trans. Computers **69**(3), 361–376 (2020)
2. Ajtai, M.: Secure computation with information leaking to an adversary. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 715–724. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993731
3. Amiel, F., Villegas, K., Feix, B., Marcel, L.: Passive and Active Combined Attacks: Combining Fault Attacks and Side Channel Analysis. In: Fourth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2007, FDTC 2007: Vienna, Austria, 10 September 2007. pp. 92–102 (2007). https://doi.org/10.1109/FDTC.2007.4318989, https://doi.org/10.1109/FDTC.2007.4318989
4. Ananth, P., Ishai, Y., Sahai, A.: Private circuits: A modular approach. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 427–455. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_15
5. Arribas, V., Wegener, F., Moradi, A., Nikova, S.: Cryptographic Fault Diagnosis using VerFI. In: HOST 2020. pp. 229–240. IEEE (2020)
6. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y.: Verified proofs of higher-order masking. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 457–485. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46800-5_18
7. Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 116–129. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978427
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.) CHES 2016. LNCS, vol. 9813, pp. 23–39. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53140-2_2
9. Belaïd, S., Cassiers, G., Mutschler, C., Rivain, M., Roche, T., Standaert, F., Taleb, A.R.: Towards achieving provable side-channel security in practice. IACR Cryptol. ePrint Arch. p. 1198 (2023), https://eprint.iacr.org/2023/1198
10. Belaïd, S., Coron, J.S., Prouff, E., Rivain, M., Taleb, A.R.: Random probing security: Verification, composition, expansion and new constructions. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 339–368. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_12

11. Belaïd, S., Mercadier, D., Rivain, M., Taleb, A.R.: IronMask: Versatile verification of masking security. In: 2022 IEEE Symposium on Security and Privacy. pp. 142–160. IEEE Computer Society Press (May 2022). https://doi.org/10.1109/SP46214.2022.9833600
12. Berndt, S., Eisenbarth, T., Faust, S., Gourjon, M., Orlt, M., Seker, O.: Combined fault and leakage resilience: Composability, constructions and compiler. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 377–409. Springer (2023). https://doi.org/10.1007/978-3-031-38548-3_13, https://doi.org/10.1007/978-3-031-38548-3_13
13. Cassiers, G., Standaert, F.: Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. IEEE Trans. Inf. Forensics Secur. **15**, 2542–2555 (2020)
14. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M.J. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer (1999)
15. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M.: Passive and Active Combined Attacks on AES: Combining Fault Attacks and Side Channel Analysis. In: 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2010, Santa Barbara, California, USA, 21 August 2010. pp. 10–19 (2010). https://doi.org/10.1109/FDTC.2010.17, https://doi.org/10.1109/FDTC.2010.17
16. Cnudde, T.D., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., Rijmen, V.: Does coupling affect the security of masked implementations? In: Guilley, S. (ed.) Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10348, pp. 1–18. Springer (2017). https://doi.org/10.1007/978-3-319-64647-3_1, https://doi.org/10.1007/978-3-319-64647-3_1
17. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Second Edition. The MIT Press and McGraw-Hill Book Company (2001)
18. Dehbaoui, A., Dutertre, J., Robisson, B., Tria, A.: Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In: FDTC 2012. pp. 7–15. IEEE Computer Society (2012)
19. Dhooghe, S.: The random fault model. IACR Cryptol. ePrint Arch. p. 1627 (2022), https://eprint.iacr.org/2022/1627
20. Dhooghe, S., Nikova, S.: My gadget just cares for me - how NINA can prove security against combined attacks. In: Jarecki, S. (ed.) CT-RSA 2020. LNCS, vol. 12006, pp. 35–55. Springer, Heidelberg (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3_3
21. Dobraunig, C., Eichlseder, M., Groß, H., Mangard, S., Mendel, F., Primas, R.: Statistical ineffective fault attacks on masked AES with fault countermeasures. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 315–342. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_11
22. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: From probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_24
23. Dumont, M., Lisart, M., Maurine, P.: Electromagnetic Fault Injection : How Faults Occur. In: FDTC 2019. pp. 9–16. IEEE (2019)

24. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 89–120 (2018)
25. Feldtkeller, J., Güneysu, T., Moos, T., Richter-Brockmann, J., Saha, S., Sasdrich, P., Standaert, F.: Combined private circuits - combined security refurbished. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023. pp. 990–1004. ACM (2023). https://doi.org/10.1145/3576915.3623129, https://doi.org/10.1145/3576915.3623129
26. Feldtkeller, J., Güneysu, T., Schaumont, P.: Quantitative fault injection analysis. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14441, pp. 302–336. Springer (2023). https://doi.org/10.1007/978-981-99-8730-6_10, https://doi.org/10.1007/978-981-99-8730-6_10
27. 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. pp. 1023–1036. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560614
28. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Çetin Kaya., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (May 2001). https://doi.org/10.1007/3-540-44709-1_21
29. Goubin, L., Patarin, J.: DES and differential power analysis (the "duplication" method). In: Koç, Çetin Kaya., Paar, C. (eds.) CHES'99. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48059-5_15
30. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_19
31. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_27
32. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_9
33. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_25
34. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: Menezes, A. (ed.) Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3376, pp. 351–365. Springer (2005). https://doi.org/10.1007/978-3-540-30574-3_24, https://doi.org/10.1007/978-3-540-30574-3_24

35. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_9
36. Renauld, M., Standaert, F., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Paterson, K.G. (ed.) Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6632, pp. 109–128. Springer (2011). https://doi.org/10.1007/978-3-642-20465-4_8, https://doi.org/10.1007/978-3-642-20465-4_8
37. 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 TCHES **2022**(4), 255–284 (2022). https://doi.org/10.46586/tches.v2022.i4.255-284
38. Richter-Brockmann, J., Rezaei Shahmirzadi, A., Sasdrich, P., Moradi, A., Güneysu, T.: FIVER – Robust Verification of Countermeasures against Fault Injections. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(4), 447–473 (Aug 2021)
39. Richter-Brockmann, J., Sasdrich, P., Güneysu, T.: Revisiting Fault Adversary Models - Hardware Faults in Theory and Practice. IEEE Trans. Computers pp. 1 – 14 (2022)
40. Roche, T., Lomné, V., Khalfallah, K.: Combined Fault and Side-Channel Attack on Protected Implementations of AES. In: Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers. pp. 65–83 (2011). https://doi.org/10.1007/978-3-642-27257-8_5, https://doi.org/10.1007/978-3-642-27257-8_5
41. Saha, S., Bag, A., Jap, D., Mukhopadhyay, D., Bhasin, S.: Divided we stand, united we fall: Security analysis of some SCA+SIFA countermeasures against SCA-enhanced fault template attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 62–94. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92075-3_3
42. Saha, S., Jap, D., Breier, J., Bhasin, S., Mukhopadhyay, D., Dasgupta, P.: Breaking Redundancy-Based Countermeasures with Random Faults and Power Side Channel. In: 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018, Amsterdam, The Netherlands, September 13, 2018. pp. 15–22. IEEE Computer Society (2018). https://doi.org/10.1109/FDTC.2018.00011, https://doi.org/10.1109/FDTC.2018.00011
43. Saha, S., Ravi, P., Jap, D., Bhasin, S.: Non-Profiled Side-Channel Assisted Fault Attack: A Case Study on DOMREP. In: Proceedings of 29th Design, Automation and Test in Europe (DATE) 2023. pp. 1–6. IEEE, Antwerp, Belgium (April 2023)
44. Schellenberg, F., Gnad, D.R.E., Moradi, A., Tahoori, M.B.: Remote inter-chip power analysis side-channel attacks at board-level. In: Bahar, I. (ed.) Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018. p. 114. ACM (2018). https://doi.org/10.1145/3240765.3240841, https://doi.org/10.1145/3240765.3240841
45. Shahmirzadi, A.R., Rasoolzadeh, S., Moradi, A.: Impeccable Circuits II. In: DAC 2020. pp. 1–6. IEEE (2020)
46. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Çetin Kaya., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/3-540-36400-5_2

47. Zussa, L., Dutertre, J., Clédière, J., Tria, A.: Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism. In: IOLTS 2013. pp. 110–115. IEEE (2013)

Appendix A Algorithms for Compositional Notions

The computation of the simulation failure probability for the composition of gadgets in the general random probing model is described in Algorithm 1 (cf. Definition 5). Similarly, Algorithm 2 specifies how to generate the correction failure probability for the composition in the general random fault model based on Definition 7. Finally, Algorithm 3 details the computation of the correction and the simulation failure probabilities for the composition in the combined random model as well as the advantage of the combined adversary (cf. Definition 9).

Algorithm 1: General Random Probing Composability Verification

```
**Inputs:** gadget $G$ with the associated set of all wire combinations $\mathcal{W}^\infty$ and wire
    probabilities $p_{w,w\in\mathcal{W}}$, masking order $d$
**Output:** simulation failure probability $\epsilon$
$E \leftarrow []$ {** List of simulation failure probabilities}
{** Loop over all possible $\mathcal{O}$ **}
**for** each $\mathcal{O} = (\mathcal{O}_1, \ldots, \mathcal{O}_m)$, with $\forall i : \mathcal{O}_i \subseteq [s]$ and $|\mathcal{O}_i| \leq d$ **do**
    $\epsilon \leftarrow 0$
    **for** each $\tilde{\mathcal{W}}$ in $\mathcal{W}^\infty$ **do**
        $\mathcal{I} = (\mathcal{I}_1, \ldots, \mathcal{I}_h) \leftarrow$ ShareSelect$(\tilde{\mathcal{W}}, \mathcal{O})$ {** Get input shares}
        **if** there exists $j \in [1, h]$ such that $|\mathcal{I}_j| > d$ **then**
            $\epsilon \leftarrow \epsilon + \prod_{w\in\tilde{\mathcal{W}}} p_w \cdot \prod_{w'\in\mathcal{W}\setminus\tilde{\mathcal{W}}} (1 - p_{w'})$ {** Equation 4}
        **end if**
    **end for**
    Append $\epsilon$ to $E$
**end for**
**return** maximum$(E)$
```

Algorithm 2: General Random Fault Composability Verification

```
Inputs: gadget $G$ with the associated set of fault combinations $\mathcal{F}^\infty$ and fault
  probabilities $q_{f,f\in\mathcal{F}}$, replication order $k$
Output: correction failure probability $\mu$
$M \leftarrow []$ {** List of correction failure probabilities}
{** Loop over all possible $\mathcal{I}'$ **}
for each $\mathcal{I}' = (\mathcal{I}'_1, \dots, \mathcal{I}'_h)$ with $\forall i : \mathcal{I}'_i \subseteq [n]$ and $|\mathcal{I}'_i| \leq k$ do
    $\mu \leftarrow 0$
    for each $\tilde{\mathcal{F}}$ in $\mathcal{F}^\infty$ do
        $\mathcal{O}' = (\mathcal{O}'_1, \dots, \mathcal{O}'_m) \leftarrow$ ReplicationSelect($G^{\tilde{\mathcal{F}}}$,$\mathcal{I}'$) {** Get output faulty
          replications for the faulted gadget}
        if there exists $j \in [1, m]$ such that $|\mathcal{O}'_j| > k$ then
            $\mu \leftarrow \mu + \prod_{f\in\tilde{\mathcal{F}}} q_f \cdot \prod_{f'\in\mathcal{F}\setminus\tilde{\mathcal{F}}}(1 - q_{f'})$ {** Equation 5}
        end if
    end for
    Append $\mu$ to $M$
end for
return maximum($M$)
```

Algorithm 3: Combined Random Composability Verification

```
Inputs: gadget $G$ with $\mathcal{F}^\infty$ and fault probabilities $q_{f,f\in\mathcal{F}}$, and with $\mathcal{W}^\infty$ and
  wire probabilities $p_{w,w\in\mathcal{W}}$, masking order $d$, replication order $k$
Outputs: probabilities $\mu$, $\epsilon_{kf}$ and $\gamma_{kc}$
$G \leftarrow []$ {** List to store advantage combined adversary}
{** Loop over all possible $\mathcal{I}'$ **}
for each $\mathcal{I}' = (\mathcal{I}'_1, \dots, \mathcal{I}'_h)$ with $\forall i : \mathcal{I}'_i \subseteq [n]$ and $|\mathcal{I}'_i| \leq k$ do
    $\mu \leftarrow 0$, $\epsilon_{kf} \leftarrow 0$
    for each $\tilde{\mathcal{F}}$ in $\mathcal{F}^\infty$ do
        $\mathcal{O}' = (\mathcal{O}'_1, \dots, \mathcal{O}'_m) \leftarrow$ ReplicationSelect($G^{\tilde{\mathcal{F}}}$,$\mathcal{I}'$) {** Get output faulty
          replications for the faulted gadget}
        if there exists $j \in [1, m]$ such that $|\mathcal{O}'_j| > k$ then
            $\mu \leftarrow \mu + \prod_{f\in\tilde{\mathcal{F}}} q_f \cdot \prod_{f'\in\mathcal{F}\setminus\tilde{\mathcal{F}}}(1 - q_{f'})$ {** Equation 5}
        else
            Return $\epsilon_{kc}^{\tilde{\mathcal{F}}}$ from Algorithm 1 with inputs $\mathcal{W}^\infty$, $p_{w,w\in\mathcal{W}}$ and $d$
            $\epsilon_{kf} \leftarrow \epsilon_{kf} + \epsilon_{kc}^{\tilde{\mathcal{F}}} \cdot \prod_{f\in\tilde{\mathcal{F}}} q_f \cdot \prod_{f'\in\mathcal{F}\setminus\tilde{\mathcal{F}}}(1 - q_{f'})$ {** Equation 6}
        end if
    end for
    $\gamma_{kc} \leftarrow \mu + \epsilon_{kf} \cdot (1 - \mu)$ {** Equation 7}
    Append $(\gamma_{kc}, \mu, \epsilon_{kf})$ to $G$
end for
return maximum value $\gamma_{kc}$ in $G$ and its corresponding $\mu$ and $\epsilon_{kf}$
```

Appendix B Extension of VERICA in the (General) Random Probing Model

B.1 Binomial Tree Definition

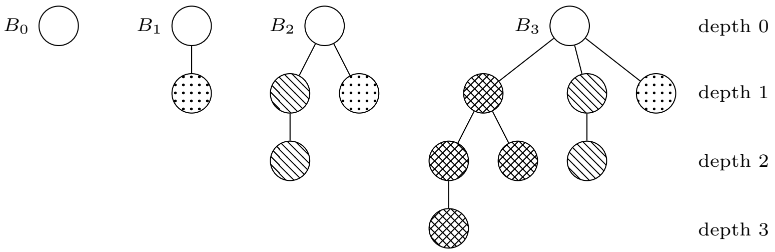


Fig. 5. Recursive definition of binomial trees, from B_0 to B_3.

B.2 Lower Bound in the Random Probing Model

If a failing wire combination is found at depth $l_{\text{fail}} \in [1, \alpha]$ (i.e., the failing wire combination contains l_{fail} wires), VERICA$^+$ determines the depth of the subtree of the current node $l_{\text{sub}} \in [0, |\mathcal{W}| - l_{\text{fail}}]$. It then updates the computation of the simulation failure probability by adding the following:

$$\begin{aligned} & p^{l_{\text{fail}}} \cdot (1-p)^{|\mathcal{W}|-l_{\text{fail}}-l_{\text{sub}}} \cdot \overbrace{\sum_{k=0}^{l_{\text{sub}}} \binom{l_{\text{sub}}}{k} p^k (1-p)^{l_{\text{sub}}-k}}^{\text{subtree cut (binomial CDF)} = 1} \\ \Leftrightarrow \quad & p^{l_{\text{fail}}} \cdot (1-p)^{|\mathcal{W}|-l_{\text{fail}}-l_{\text{sub}}} . \end{aligned} \tag{12}$$

Thus, the subtree of the current node is included entirely in the computation of the simulation failure probability. This is explicitly reflected in Equation 12 with the binomial CDF for a sample of size l_{sub} (all combinations of wires in the subtree are considered) and probability p. We evaluate it for l_{sub}, thus it equals one.

Additionally, about Figure 2, the above equation would be calculated with $l_{\text{fail}} = l_{\text{sub}} = 2$ and $|\mathcal{W}| = 4$, which is equal to p^2.

B.3 An Upper Bound in the (General) Random Probing Model

VERICA$^+$ computes an upper bound for the simulation failure probability, by considering that if combinations of α wires did not lead to a simulation failure, then combinations of $\alpha + 1$ wires will (cf. Section 5.1).

In short, VERICA$^+$ performs the binomial tree traversal and the computation of the lower bound of the simulation failure as explained in Section 5.2 and

Appendix B.2. However, in addition, if it visits a node at depth α and the set of leaking wires to verify does not lead to a simulation failure, it considers that the remaining nodes in the un-visited subtree of depth $l_{\mathrm{sub}_\alpha} \in [0, |\mathcal{W}| - \alpha]$ will lead to a simulation failure with the parent nodes. The exact computation depends on the security model.

In the Random Probing Model. If $l_{\mathrm{sub}_\alpha} > 0$, VERICA$^+$ additionally updates the computation of $\epsilon^p_{\mathrm{max}}$ with:

$$\begin{aligned} & p^\alpha \cdot \sum_{k=1}^{l_{\mathrm{sub}_\alpha}} \binom{l_{\mathrm{sub}_\alpha}}{k} p^k (1-p)^{|\mathcal{W}|-\alpha-k} \\ \Leftrightarrow \quad & p^\alpha \cdot (1-p)^{|\mathcal{W}|-\alpha-l_{\mathrm{sub}_\alpha}} \cdot \left(1-(1-p)^{l_{\mathrm{sub}_\alpha}}\right) . \end{aligned} \tag{13}$$

Note that the sum from the first equation starts at $k = 1$ since we consider that the combinations of the α wires from the root to the current node do not fail simulation, but their combinations with any wire represented in the children nodes will.

In the General Random Probing Model. Similarly, VERICA$^+$ adds to the computation of ϵ_{max}:

$$\begin{aligned} & \prod_{w\in\tilde{\mathcal{W}}_\alpha} p_w \cdot \prod_{\substack{w'\in\mathcal{W}\\ w'\notin\tilde{\mathcal{W}}_\alpha\\ w'\notin\mathcal{W}_{\mathrm{sub}_\alpha}}} (1-p_{w'}) \cdot \sum_{k=1}^{l_{\mathrm{sub}_\alpha}} \sum_{\substack{\tilde{\mathcal{W}}\in\mathcal{W}^{\infty}_{\#k}\\ \tilde{\mathcal{W}}\subseteq\mathcal{W}_{\mathrm{sub}_\alpha}}} \prod_{w\in\tilde{\mathcal{W}}} p_w \prod_{\substack{w'\in\mathcal{W}_{\mathrm{sub}_\alpha}\\ w'\notin\tilde{\mathcal{W}}}} (1-p_{w'}) \\ \Leftrightarrow \; & \prod_{w\in\tilde{\mathcal{W}}_\alpha} p_w \prod_{\substack{w'\in\mathcal{W}\\ w'\notin\tilde{\mathcal{W}}_\alpha\\ w'\notin\mathcal{W}_{\mathrm{sub}_\alpha}}} (1-p_{w'}) \cdot \left(1 - \prod_{w''\in\mathcal{W}_{\mathrm{sub}_\alpha}} (1-p_{w''})\right) , \end{aligned} \tag{14}$$

where $\tilde{\mathcal{W}}_\alpha$ are the α wires from the root to the current node that do not lead to a simulation failure, and $\mathcal{W}_{\mathrm{sub}_\alpha}$ is the set of the wires that are represented by the children nodes of the current node at depth α. Note that the first expression is similar to Equation 8, but the outer sum starts at $k = 1$ for the same reason stated in the random probing model. Therefore, with the double summation, we construct all wire combinations of size $k > 1$ from the wires in $\mathcal{W}_{\mathrm{sub}_\alpha}$ that are assumed to fail the simulation. Its probability simplifies to one minus the probability to not take any wire in $\mathcal{W}_{\mathrm{sub}_\alpha}$.

B.4 Binomial Tree Traversal

VERICA$^+$ considers only the intermediate wires $\mathcal{W}$ of a given circuit C, which are labeled in topological order. As explained in Section 5.2, to evaluate Equation 4 for up to α wires, VERICA$^+$ uses the binomial tree structure, where each node symbolizes a unique wire (except the root). The tree represents all the possible wire combinations from the set of wires $\mathcal{W}$. VERICA$^+$ then creates the leaking wire combinations $\tilde{\mathcal{W}}$ by traversing the tree in a preorder fashion up to depth α. We take advantage of the node labeling to traverse the binomial tree $B_{|\mathcal{W}|}$ using a recursive function in Algorithm 4.

Algorithm 4: Binomial Tree Preorder Traversal to Compute Simulation Failure Probability

```
Inputs: circuit $C$ with the set of wires $\mathcal{W}$ and corresponding wire
  probabilities $p_{w,w\in\mathcal{W}}$, threshold number of wires $\alpha$
Output: bounds of simulation failure probability $\epsilon_{\min}$, $\epsilon_{\max}$
$\epsilon_{\min} \leftarrow 0$, $\epsilon_{\max} \leftarrow 0$ {** Lower and upper bounds}
$\tilde{\mathcal{W}} \leftarrow \{\}$ {** Leaking wire combination to verify}
{** Define recursive function **}
procedure PREORDER($e_{\min}, e_{\max}, \tilde{\mathcal{W}}, k$)
  for $i = k$ to $|\mathcal{W}|$ do
    if $|\tilde{\mathcal{W}}| = \alpha$ then
      $\epsilon_{\max} \leftarrow \epsilon_{\max} +$ update_upp_bound($\mathcal{W}, \alpha, \tilde{\mathcal{W}}, p_{w,w\in\mathcal{W}}$)
      {** Equation 13, or Equation 14 with $\tilde{\mathcal{W}} = \tilde{\mathcal{W}}_\alpha$}
      break
    else
      Append $w_i$ to $\tilde{\mathcal{W}}$
      if VERICA$^+$ returns a simulation failure for $\tilde{\mathcal{W}}$ then
        $\epsilon_{\min} \leftarrow \epsilon_{\min} +$ update_low_bound($\mathcal{W}, \alpha, \tilde{\mathcal{W}}, p_{w,w\in\mathcal{W}}$)
        $\epsilon_{\max} \leftarrow \epsilon_{\max} +$ update_low_bound($\mathcal{W}, \alpha, \tilde{\mathcal{W}}, p_{w,w\in\mathcal{W}}$)
        {** Equation 12, or Equation 8 with $\tilde{\mathcal{W}} = \tilde{\mathcal{W}}_{\text{fail}}$}
      else
        $\epsilon_{\min}$, $\epsilon_{\max} \leftarrow$ PREORDER($\epsilon_{\min}, \epsilon_{\max}, \tilde{\mathcal{W}}, i+1$)
      end if
      Remove $w_i$ from $\tilde{\mathcal{W}}$
    end if
  end for
  return $\epsilon_{\min}$, $\epsilon_{\max}$
end procedure
{** Call recursive function **}
$\epsilon_{\min}$, $\epsilon_{\max} \leftarrow$ PREORDER($\epsilon_{\min}, \epsilon_{\max}, \tilde{\mathcal{W}}, 1$)
```

Appendix C Extension of IronMask in the Random Probing Model

A pseudo-code of IronMask$^+$ for the verification of a tuple in the random probing model is given in Algorithm 5, with blue highlights on the added replacements. The only differences with the original IronMask's verification lies in the calls to the new function `replace_correction_outputs`, available in Algorithm 6.

Functions `factorize` and `input_shares` are not detailed here since they are explained in [11] and are not useful in this extension's description.

Algorithm 5: IronMask^{+} Verification of a Tuple

```
Inputs: a tuple of probes $\boldsymbol{w} = (w_1, \ldots, w_k)$
Outputs: inputs shares required for the simulation input_shares
$L = []$ {**  List of symbolic expressions}
$R = []$ {**  List of random masks for each symbolic expression}
{**  First Step  **}
for $i = 1$ to $k$ do
  for $j = 1$ to length($L$) do
    if $R[j]$ is in the symbolic expression of $w_i$ then
      $w_i \leftarrow w_i + L[j]$
    end if
  end for
  Append $w_i$ to $L$
  if there is any remaining additive random $r$ in the expression of $w_i$ then
    Append $r$ to $R$
  else
    Append 0 to $R$
  end if
  call replace_correction_outputs($w_i$, $L$, $R$)
end for
{**  Second Step  **}
$L' = []$
$R' = []$
for $i = 1$ to length($L$) do
  if $R[i] = 0$ then
    fact $\leftarrow$ factorize($L[i]$) {**  factorized expressions of $L[i]$}
    for each $w$ in fact do
      for $j = 1$ to len($L'$) do
        if $R'[j]$ is in the symbolic expression of $w$ then
          $w \leftarrow w + L'[j]$
        end if
      end for
      Append $w$ to $L'$
      if there is any remaining additive random $r$ in the expression of $w$ then
        Append $r$ to $R'$
      else
        Append 0 to $R'$
      end if
      call replace_correction_outputs($w$, $L'$, $R'$)
    end for
  end if
end for
input_shares $\leftarrow$ get_input_shares($L', R'$) {**  input shares from $L'$}
```

Algorithm 6: Function `replace_correction_outputs`

```
Inputs and Outputs: probe $w$, lists $L$ and $R$
if $R[-1] \neq 0$ then
    return
end if
for each correction block output $s_i$ in the expression of $w$ do
    $w_i \leftarrow s_i$
    for $j = 1$ to len($L$) do
        if $R[j]$ is in the expression of $w_i$ then
            $w_i \leftarrow w_i + L[j]$
        end if
    end for
    Append $w_i$ to $L$
    if there is any remaining additive random $r$ in the expression of $w_i$ then
        Append $r$ to $R$
    else
        Append 0 to $R$
    end if
    call replace_correction_outputs($w_i$, $L$, $R$)
end for
```