Combined Private Circuits - Combined Security Refurbished

Jakob Feldtkeller
Ruhr University Bochum
Horst Görtz Institute for IT Security
Bochum, Germany
jakob.feldtkeller@rub.de

Tim Güneysu
Ruhr University Bochum
Horst Görtz Institute for IT Security
Bochum, Germany
tim.gueneysu@rub.de

Thorben Moos
Université catholique de Louvain
Crypto Group, ICTEAM Institute
Louvain-la-Neuve, Belgium
thorben.moos@uclouvain.be

Jan Richter-Brockmann
Ruhr University Bochum
Horst Görtz Institute for IT Security
Bochum, Germany
jan.richter-brockmann@rub.de

Sayandeep Saha
Université catholique de Louvain
Crypto Group, ICTEAM Institute
Louvain-la-Neuve, Belgium
sayandeep.saha@uclouvain.be

Pascal Sasdrich
Ruhr University Bochum
Horst Görtz Institute for IT Security
Bochum, Germany
pascal.sasdrich@rub.de

François-Xavier Standaert
Université catholique de Louvain
Crypto Group, ICTEAM Institute
Louvain-la-Neuve, Belgium
fstandae@uclouvain.be

ABSTRACT

Physical attacks are well-known threats to cryptographic implementations. While countermeasures against passive Side-Channel Analysis (SCA) and active Fault Injection Analysis (FIA) exist individually, protecting against their combination remains a significant challenge. A recent attempt at achieving joint security has been published at CCS 2022 under the name CINI-MINIS. The authors introduce relevant security notions and aim to construct arbitrary-order gadgets that remain trivially composable in the presence of a combined adversary. Yet, we show that all CINI-MINIS gadgets at any order are susceptible to a devastating attack with only a single fault and probe due to a lack of error correction modules in the compression. We explain the details of the attack, pinpoint the underlying problem in the constructions, propose an additional design principle, and provide new (fixed) provably secure and composable gadgets for arbitrary order. Luckily, the changes in the compression stage help us to save correction modules and registers elsewhere, making the resulting Combined Private Circuits (CPC) more secure *and* more efficient than the original ones. We also explain why the discovered flaws have been missed by the associated formal verification tool VERICA (TCHES 2022) and propose fixes to remove its blind spot. Finally, we explore alternative avenues to repair the compression stage without additional corrections based on non-completeness, i.e., constructing a compression that never recombines any secret. Yet, while this approach could have merit for low-order gadgets, it is, for now, hard to generalize and scales poorly to higher orders. We conclude that our refurbished arbitrary order CINI gadgets provide a solid foundation for further research.

CCS '23, November 26–30, 2023, Copenhagen, Denmark

CCS CONCEPTS

• **Security and privacy → Side-channel analysis and countermeasures**.

KEYWORDS

Side-Channel Analysis; Fault-Injection Analysis; Combined Attacks; Gadgets; CINI MINIS

1 INTRODUCTION

Standard black-box assumptions about cryptographic primitives often fail to hold in practice as an adversary can extract information about the internal computations of a cryptographic device by observing or manipulating its physical characteristics. On the one hand, a *passive* adversary exploits the fact that physical effects, such as timing [26], power consumption [27], or electromagnetic (EM) radiation [20] of a device are correlated with the data being processed. On the other hand, an *active* adversary deliberately perturbs the operating conditions (e.g., by clock/voltage glitches [4, 42], or EM/laser pulses [31, 41]) of the device to create computational faults and extracts information through faulty system behavior [8, 9].

Over the years, several countermeasures have been proposed to counter such passive Side-Channel Analysis (SCA) and active Fault Injection Analysis (FIA). For SCA *masking* [13] was introduced, which provides quantifiable security guarantees. The main idea behind masking is to split each variable into multiple independent random *shares* (cf. Section 3.1.4) so that the adversary is bound to perform a higher-order/multivariate statistical analysis with an exponential data complexity in the number of shares to recover the actual intermediate values (under some assumption of noise and independence [16, 17]). A well-researched strategy to circumvent FIA is to incorporate computational *redundancy* in space, time, or information [24] to detect/correct occurring faults (cf. Section 3.2.4). In addition to attacks and countermeasures, the research community developed theoretical models to argue about security formally. The formal models for SCA consider access to internal values without

or with noise [16, 23, 32]. Likewise, FIA is modeled by allowing restricted capabilities of circuit manipulation [22, 35].

Countermeasures Against Combined Attacks. While both SCA and FIA are well-explored in isolation, both theoretically and practically, the security against an attacker able to mount both SCA and FIA simultaneously is a new field of research. Indeed, only recently it has been shown that such Combined Analysis (CA) is feasible in practice, even with low-end injection and measurement setups [2, 14, 36, 37, 39, 40]. In addition, the pure combination of countermeasures against SCA and FIA is not sufficient to protect against CA, neither in theoretic models [15, 34] nor in practical implementations [36, 37, 40]. Therefore, there is a need for countermeasures in the context of CA with provable and formally verifiable security claims.

Related Work: Only a few works have addressed the problem of constructing combined countermeasures. Initial efforts in this regard were based on the principles of Multi-Party Computation (MPC) [29, 33]. However, proving security in such MPC-based schemes is highly complex. Recently the SCA research community has focused on designing small, secure *gadgets*, i.e., small subcircuits dedicated to some specific functionality, for which security can be proven even when combined to larger circuits (composition) [6]. In a nutshell, such composability notions restrict the propagation of leakage between different gadgets with isolation and re-randomization. The earliest notions in this regard are Non-Interference (NI) [5] and Strong Non-Interference (SNI) [6]. Later, Probe-Isolating Non-Interference (PINI) [12] has been proposed as an elegant way to achieve composition by isolating so-called *share-domains* (cf. Section 3.1.5). Overall, gadget-based, composable constructions are a convenient way to build secure implementations against physical attacks.

Similar to NI and SNI, composability notions have also been proposed for FIA and CA [15, 19, 34]. For CA, the notions for composition come in two flavors: (i) the SCA-based security order (i.e., the number of observations an adversary can make) is dependent on the number of injected faults, (ii) the security guarantees in SCA and FIA are independent of each other. The most recent work in this regard is Combined-Isolating Non-Interference (CINI) [19], where the concept of share-domain isolation in PINI was extended to the context of CA. More precisely, the concept of Shared Redundancy Domains (SRDs), i.e., the intersection of shares and redundancies, is introduced. Then the leakage of probes is isolated within share domains, while the leakage from faults is isolated within SRDs (cf. Section 3.3.4). The authors also adapted a verification tool for CA, called VERICA [34], to support this composability notion for the automatic verification of small gadgets. Finally, the authors proposed several gadget constructions (CINI-MINIS), which, to the best of our knowledge, are the only existing gadgets based on Boolean masking and redundancy for hardware in the context of CA which until now have not been shown insecure. In this work, we demonstrate that these constructions need refinement, too, since they are missing a crucial design principle.

Contributions: In this work, we first extend the well-established concepts of probe and fault propagation to a propagation framework in a combined setting (Section 3.3.3). Building on that, we show that the gadgets proposed by Feldtkeller et *al.* [19] are vulnerable to a simple yet effective combined attack when composed (Section 4). More precisely, an adversary can observe secret-dependent fault propagation, breaking the security with only a single probe and fault. Even worse, this attack was not detected by VERICA when verifying for CINI. This fact highlights the complexity of designing for CA. To tackle this complexity, design guidelines and principles are an essential building block. Given our attack, we pinpoint the exact vulnerability and extend the design principles for constructing CINI gadgets, eventually contributing to the general understanding of CA-secure constructions. The introduced principle guides the construction of two types of new provably secure CINI gadgets (called Combined Private Circuit (CPC)) and a fix to VERICA (Section 5). The first gadget adds additional correction modules between the computation of partial products and the compression, however, it enables us to save some corrections and registers at other locations (Section 6). In total, this results in a net benefit in terms of area while fixing the security. In addition, we explore gadgets based on the *non-completeness* property from Threshold Implementations (TIs) [30] (Section 7). While this approach could have merit for low-order gadgets, it is, for now, hard to generalize and scales poorly to higher orders. For all our gadgets, we evaluate the performance and provide a tool-based verification (Section 8).

2 BACKGROUND

In this section, we provide important notations, describe our circuit model, and discuss proof techniques based on simulation.

2.1 Notation

In general, we use upper-case calligraphic font for sets (e.g., $\mathcal{S}$) and sans serif font for functions (e.g., f). Further, we use superscript to denote the replication index and subscript for the share index of a value. Throughout the paper, the term *faulty value* is used as shorthand for a fault that was injected in the gate that produces the corresponding value.

2.2 Circuit Model

Without loss of generality, we restrict the set of combinatorial gates to $\mathcal{G}_\mathsf{c} = \{\mathsf{inv}, \mathsf{and}, \mathsf{xor}\}$ and the set of memory gates to clocked registers $\mathcal{G}_\mathsf{m} = \{\mathsf{reg}\}$. For randomized behavior, we use a randomness-generating gate $\mathcal{G}_\mathsf{r} = \{\mathsf{rand}\}$, that outputs an independent and uniformly chosen value from $\mathbb{F}_2$ per clock cycle.

Then, we model a digital-logic circuit C as a *directed acyclic graph* $\mathcal{D} = \{\mathcal{V}, \mathcal{E}\}$, where vertices $v \in \mathcal{V}$ represent logical gates $g \in \mathcal{G}_\mathsf{c} \cup \mathcal{G}_\mathsf{m} \cup \mathcal{G}_\mathsf{r}$ and edges $e \in \mathcal{E}$ represent wires carrying an element of the finite field $\mathbb{F}_2$.

2.3 Security via Simulation

Simulation is a technique often used for security proofs [10, 28]. It defines two games, a *real* and an *ideal* game. The ideal game is trivially secure (under some adversary model), while the real game is secure iff there exists no adversary who can distinguish between the ideal and the real game with a probability higher than 1/2. The ideal game is constructed as a probabilistic polynomial-time simulator reproducing the view of the adversary without knowledge

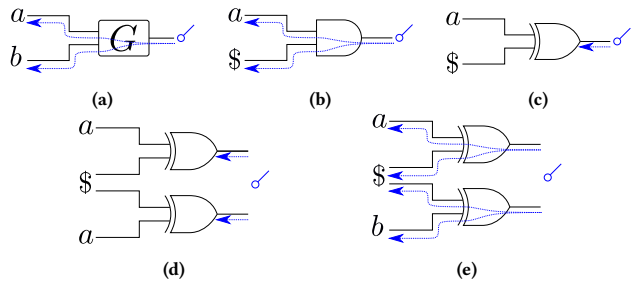

Figure 1. Rules for probe propagation.

of any secret. This simulator treats each input-dependent value as deterministic.

3 PROBING & FAULTING

3.1 Side-Channel Analysis

3.1.1 Adversary Model. In the stateless *d-probing model* [23], the adversary $\mathcal{A}_p$ is given access to a circuit C that can be invoked multiple times. Before each invocation, $\mathcal{A}_p$ can select up to d wires of C to be probed. Then, while executing, the circuit leaks the *glitch-extended probes* [18] to $\mathcal{A}_p$, i.e., all values stored in registers the probed values directly depend on[1]. In this work, we directly consider the glitch-extended instead of the standard probing model to focus on hardware implementations primarily.

3.1.2 Probing Security. In this context, a circuit C is *probing-secure* iff the view of $\mathcal{A}_p$ can be simulated without access to any secret [23].

3.1.3 Probe Propagation. The information a probing adversary $\mathcal{A}_p$ can learn from a set of probes is dependent on the structure of the circuit leading up to the probes. The leaked information captured by a set of probes can be determined via the concept of *probe propagation* [7]. Here, a probe propagates from wire w_1 to wire w_0 iff the value of wire w_0 is required for the simulation of w_1. We provide the most important propagation rules in Figure 1, where probes always propagate from right to left. In particular, probes propagate backwards through the circuit (cf. Figure 1a and 1b) until they are stopped by the addition of unique randomness (cf. Figure 1c). Here, uniqueness means that the random value is not observed by any other propagating probe or at least only while masking the same intermediate value (cf. Figure 1d and 1e). We emphasize that a placed probe is first glitch-extended to the associated registers before a probe propagates backward through the circuit, starting from the respective registers. While probe propagation and glitch extension of a probe have some similarities, they still have fundamentally different properties. Probe propagation is stopped only by refreshing an intermediate value and is not stopped by registers. In contrast, glitch extension is stopped by registers but not by refreshing.

3.1.4 Masking. A well-understood countermeasure against SCA is *Boolean masking* [13]. Here, each value $x \in \mathbb{F}_2$ is replaced by a vector $\langle x_0, \dots x_{s-1} \rangle \in \mathbb{F}_2^s$, where each $x_i \in \mathbb{F}_2$ is uniformly random, $x = \bigoplus_{i=0}^{s-1} x_i$, and each subset $\hat{X} = \{x_i \mid i \in [s-1]\}$ with $|\hat{X}| < s$ is independent of x. A component x_i is called a share with share index i. To securely process masked values, the computation circuit is transformed to a *shared circuit*, where each logical operation consists of a set of gates manipulating the shared values. In such a shared circuit, the initial sharing and final unsharing operation are not part of the shared circuit and cannot be probed by $\mathcal{A}_p$ [3, 23].

Definition 3.1 (Share Domain). The *share domain i* of a shared circuit is defined by all wires with share index i.

3.1.5 Probe-Isolating Non-Interference. While being the fundamental goal, probing-secure circuits are not always composable, i.e., the combination of two probing-secure circuits is not necessarily probing secure again. To ease the construction of probing-secure circuits, different notions of composition were introduced, which define how to construct atomic building blocks, so-called *gadgets*, that can be securely composed into larger structures. PINI [12] requires the isolation of probe propagation within *share domains* (cf. Definition 3.1), i.e., a probe is only allowed to propagate within a single share domain. This ensures that the combination of multiple PINI gadgets is always PINI again.

Definition 3.2 (Probe-Isolating Non-Interference [12]). A gadget G is d-PINI iff for any set of d_1 internal probes and any set $\mathcal{S}_2$ of d_2 share domains, such that $d_1 + d_2 \leq d$, there exists a set $\mathcal{S}_1$ of at most d_1 share domains such that the outputs of the share domains in $\mathcal{S}_2$ and the probes can be simulated with the inputs of the share domains in $\mathcal{S}_1 \cup \mathcal{S}_2$.

3.2 Fault-Injection Analysis

3.2.1 Adversary Model. A faulting adversary $\mathcal{A}_f$ is given access to a circuit C that can be invoked multiple times [22, 34]. Before each invocation, $\mathcal{A}_f$ can select up to k gates of C to be faulted and for each such gate a fault type from a set of allowed fault types $\mathcal{T}$. Then, before invocation, the faulted gates are replaced by a different gate type specified by the fault type $t \in \mathcal{T}$ [35]. Commonly used fault types are *set*, *reset* (replacing the targeted gate with a constant one or zero, respectively), or *bit flips* (inversion of the gate). After execution of the circuit, the correctness is leaked to $\mathcal{A}_f$, i.e., whether the output is equal to the output of the *golden circuit*, which is the fault-free version of C. Please note, we do not consider fault-detection mechanisms in this work, due to the difficulty to implement them in hardware [34].

3.2.2 Fault Security. In this context, a circuit C is *fault-secure* iff all faults can be corrected at the output [34]. That is, there exists a correction circuit G^C, such that the concatenation $G^C(C(\cdot))$ always yields an output equal to the golden circuit.

3.2.3 Fault Propagation. Similar to probe propagation, fault propagation is a well-known concept in the literature [1, 38]. A fault propagates from wire w_0 to w_1 iff a faulty value at w_0 causes a faulty value at w_1, i.e., if a difference between C and the golden circuit in w_0 causes a difference between C and the golden circuit in w_1. As such, faults propagate always towards the outputs of a circuit. Again, we provide the most important rules in Figure 2, where faults always propagate from left to right. A fault occurs in an output wire w_g of a gate g if the fault injected in g causes an effective fault at w_g (cf. Figure 2a), i.e., the value carried by w_g

[1]Glitches are short-term evaluation defects that occur due to timing differences in the propagation path of signals. Providing $\mathcal{A}_p$ with all stable inputs is a worst-case assumption on the leakage via glitches [18].

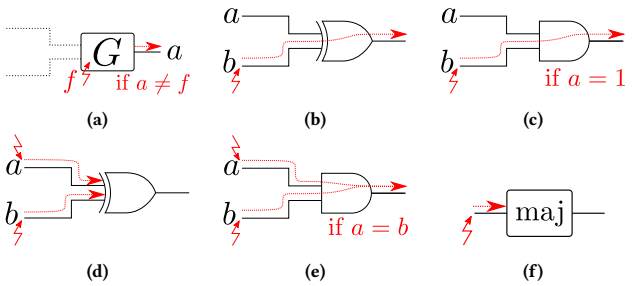

Figure 2. Rules for fault propagation.

in the faulted circuit differs from the value in the golden circuit. Hence, observing this fault provides information about the correct value of w_g. Similarly, a single fault propagates through a multiplication (we talk about addition and multiplication in $\mathbb{F}_2$ from here on) only if the other operand a is equal to 1 (cf. Figure 2c). Again, the observation of such a fault propagation leaks some information about a. In contrast, a single fault always propagates through an addition (cf. Figure 2b). When both input operands are faulty then the fault propagates through a multiplication only if both operands are equal and does not propagate through an addition (cf. Figure 2d and 2e). A majority vote always stops the propagation of a certain maximum number of incoming faults (cf. Figure 2f).

3.2.4 Redundancy. Protecting against fault attacks always requires some form of redundancy, either in time, space, or information. The most basic form of redundancy is *replication*, where all data and computation are instantiated multiple times in parallel. Then, correction of corrupted data is possible with $2k+1$ replications via a majority vote maj. The initial replication and the final error correction are not part of the replicated circuit and cannot be faulted by $\mathcal{A}_f$ [34]. For correctness, all replications must use the same random values whenever a randomness gate is used in C.

Definition 3.3 (Redundancy Domain). The *redundancy domain* ℓ of a redundant circuit is defined by all gates and wires with replication index ℓ.

3.2.5 Fault-Isolating Non-Interference. The combination of two fault-secure circuits is in general not fault-secure. Hence, additional notions of composition were introduced for construction and analysis. Similar to PINI, Fault-Isolating Non-Interference (FINI) [19] requires the isolation of fault propagation within redundancy domains (cf. Definition 3.3). This is a natural expression of the security guarantees given by replication codes and ensures that any combination of FINI gadgets is FINI again.

Definition 3.4 (Fault-Isolating Non-Interference [19]). A gadget G is k-FINI iff the following holds:

(i) For any set $\mathcal{F}_1$ of k_1 faulty redundancy domains and every set of k_2 faults injected in gates of G, with $k_1 + k_2 \leq k$, there exists a set of at most k_2 redundancy domains $\mathcal{F}_2$, such that the gadget gives an output where all values, except those belonging to the redundancy domains $\mathcal{F}_1 \cup \mathcal{F}_2$, are equal to the values of the golden circuit.

(ii) There exists a decoding gadget G^{D}, such that given an input with at most k faulty redundancy domains, G^{D} outputs a correct result.

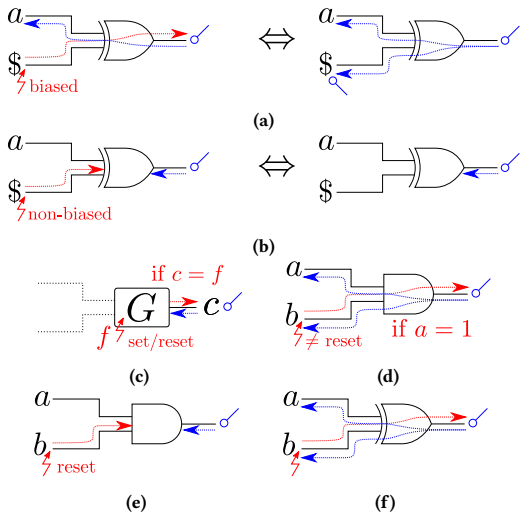

Figure 3. Rules for combined propagation.

3.3 Combined Analysis

3.3.1 Adversary Model. An adversary $\mathcal{A}_c$ with both faulting and probing capabilities is a trivial combination of $\mathcal{A}_p$ and $\mathcal{A}_f$ [15, 34], i.e., $\mathcal{A}_c$ features both capabilities (selecting d wires for probing and selecting k gates with fault types for faulting) and both leakages (glitch-extended probes and correctness of the output). However, for CA the definition of the golden circuit is slightly modified. In particular, all faults in randomness gates $g_r \in \mathcal{G}_\mathrm{r}$ are also injected into the golden circuit [34]. Please note, faults change the distribution of affected intermediate values and in turn affect the observation made by the placed probes (cf. Section 3.3.3).

3.3.2 Combined Security. In this context, a circuit C is *combined-secure* iff the view of $\mathcal{A}_c$ can be simulated without access to any secret (*privacy*) and there exists a correction circuit G^C, such that the concatenation $G^C(C(\cdot))$ always yields an output equal to the golden circuit (*correctness*) [34]. Following probing and fault security, $\mathcal{A}_c$ is not allowed to probe or fault the initial sharing and replication, nor the final unsharing and correction.

3.3.3 Combined Propagation. The concepts of probe and fault propagation can be combined to analyze the leakage of information in a combined attack setting. We provide the most important rules in Figure 3. In general, a fault can impact a set of probes in four different ways. Two are beneficial to $\mathcal{A}_c$, one is to the disadvantage of $\mathcal{A}_c$, and one has no impact.

Entropy Removal. Since probing security is primarily achieved through randomization the removal of entropy via faults can enhance the propagation of probes. For example, manipulating a random value with a biased fault gives $\mathcal{A}_c$ the control over the used randomness and, hence, is equivalent to a known random value (cf. Figure 3a) [34]. However, this is only true for biased faults, while non-biased faults provide the adversary with no advantage at all (cf. Figure 3b) [34].

Conditioned Fault Propagation. Whenever a fault propagation is conditioned on some internal value a and $\mathcal{A}_c$ can distinguish

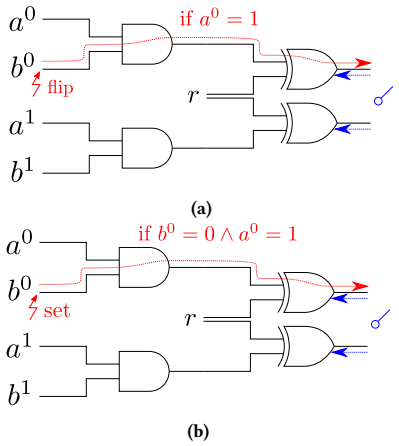

Figure 4. Combined propagation examples where $a^0 = a^1$ and $b^0 = b^1$.

between a faulty and a non-faulty wire (i.e., whether the fault propagates or does not propagate) the condition on a is leaked to $\mathcal{A}_c$. This can happen due to a biased fault injected into a gate since the fault sets the output to a known value that is either correct or not (cf. Figure 3c). Or it happens when a non-reset fault propagates through a multiplication since the fault only propagates if the other input is set to one (cf. Figure 3d). While propagating through the circuit a fault can accumulate different propagation conditions and thereby indirectly recombine multiple shares of a secret. Intuitively, such a fault serves as an additional probe on the conditioned value. However, without probe propagation and, therefore, no protection via refreshing. It is important to note that such a fault alone is not sufficient for leakage. Instead, a probe is required that captures the existence of fault propagation, e.g., by observing a correct and faulty redundant value.

Information Removal. A biased fault completely determines the output distribution of an affected gate. Hence, the output is determined only by the known fault and not by the input to the gate. This essentially disconnects the input and output of the faulted gate and thereby stops the propagation of an incoming probe (cf. Figure 3c). A similar effect is achieved when a reset fault propagates through a multiplication since again the output of the gate is fully determined by the faulty input and always set to zero (cf. Figure 3e).

No Impact. A propagating fault has no impact on probe propagation for gates with deterministic inputs (cf. Figure 3f and 3d), as long as the fault does not remove information (see above). In such a case, the fault and probe propagation passes one another.

The four types of fault impacts are by no means exclusive and a single fault can cause both effects at different locations of the circuit. As probes are passive, probe propagation has no impact on faults.

Example. For clarification of the concept of combined propagation, let us consider the example in Figure 4a. From a pure probing security perspective, this circuit does not reveal any information about the inputs, since $a^0 = a^1$ and $b^0 = b^1$ which means r is observed two times refreshing the same value (cf. Figure 1e). A bit-flip fault in b^0 will cause an effective fault at the output of $a^0b^0 + r$ iff $a^0 = 1$. Observing only this value also reveals no information about the inputs, as $\mathcal{A}_c$ cannot distinguish between an effective and an ineffective fault. However, observing both outputs in Figure 4a allows $\mathcal{A}_c$ to identify effective faults (if both values are different). Hence, the condition of the probe propagation, i.e., $a^0 = 1$, is leaked to the adversary, which means the input a^0 is required for the simulation of the probes (if $a^0 = 0$ both probes are random but equal; if $a^0 = 1$ both probes are random but different).

When injecting a set fault in b^0 instead, both a^0 and b^0 are required for simulation (cf. Figure 4b). The reason is that now the effectiveness of the fault in b^0 is dependent on the value of b^0 in the first place. The fault is only effective if $b^0 = 0$. Again, the fault propagates to the probe only if $a^0 = 0$ and $\mathcal{A}_c$ can observe the effectiveness of the fault by comparing both outputs.

Definition 3.5 (Shared Redundancy Domain). The *SRD* (i, ℓ) of replicated and shared circuits is defined by all gates and wires with share index i and replication index ℓ.

3.3.4 Combined-Isolating Non-Interference. Again, the combination of two combined-secure circuits is not always combined-secure and additional notions for composition were introduced. CINI [19] is a combination of PINI and FINI in that it isolates probe propagation and glitch extension within share domains and fault propagation within SRDs (cf. Definition 3.5). An SRD is the intersection of a share domain with a redundancy domain and it is required to restrict a fault to a single SRD to avoid the cross-share-domain leakage via faults [19]. In its simple form, CINI requires the order of probing security to be always at least the sum of the injected faults and probes, i.e., the number of injected faults reduces the number of remaining probes. The reason is the probe-like nature of faults when conditioned on internal values. We provide a formal definition of CINI in Definition 3.6. With Independent Combined-Isolating Non-Interference (CINI_{ind}) [19] there exists also a variant where the number of probes is always only restricted by the order of probing security and independent of the injected faults, which potentially allows more efficient implementations for a certain security order. Both variants are trivially composable, i.e., any combination of CINI (CINI_{ind}) gadgets is CINI (CINI_{ind}) again. In this work, we focus on CINI and discuss the relation to CINI_{ind} only briefly.

Definition 3.6 (Combined-Isolating Non-Interference [19]). A gadget G is (d, k)-CINI iff for any set $\mathcal{F}_1$ of k_1 faulty SRDs, every set of k_2 faults injected in gates of G, any set of d_1 probes placed on intermediate values, and any set $\mathcal{S}_2$ of d_2 share domains, such that $k_1 + k_2 \leq k$ and $d_1 + d_2 + k_1 + k_2 \leq d$, there exists a set $\mathcal{F}_2$ of at most k_2 SRDs and a set $\mathcal{S}_1$ of at most $d_1 + k_2$ share domains such that the following holds:

Correctness: The gadget gives an output where all values, except those belonging to the SRDs $\mathcal{F}_1 \cup \mathcal{F}_2$, are equal to the golden circuit, and there exists a decoding gadget G^{D}, such that given an input with at most k faulty SRDs, G^{D} outputs a correct result.

Privacy: The outputs of the share domains in $\mathcal{S}_2$, the outputs violating the independence property of Boolean sharing, and the probes can be simulated with the

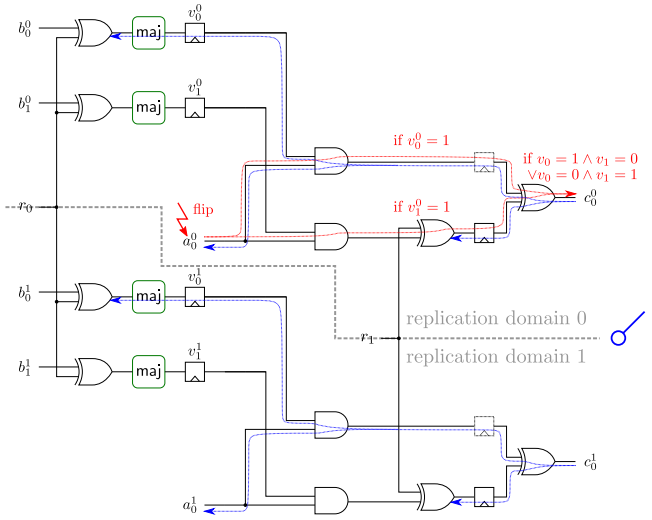

Figure 5. Conditioned fault propagation exploited by our attack.

inputs of the share domains in $\mathcal{S}_1 \cup \mathcal{S}_2$ and knowledge of the faults both injected and on inputs in $\mathcal{F}_1$.

4 ATTACKING THE COMBINED GADGET

In this section, we present a combined attack on the CINI and CINI_{ind} gadgets proposed by Feldtkeller et *al.* [19]. All three of those gadgets follow the same design principle: computing everything in multiple redundancy domains and performing error correction (using all redundancy domains), and mask refreshing whenever an intermediate value crosses SRD boundaries. This principle should ensure that probe propagation is isolated within share domains and fault propagation within SRDs.

In the following, we describe a combined attack on those gadgets, based on the properties of combined propagation (cf. Section 3.3.3), without violating the basic security conditions described by [19]. The attack requires a single probe and a single fault in a simple composition of gadgets, exploiting a secret-dependent fault propagation observed by the probe.

Attack Outline. The core principle of our attack is shown in Figure 5. The general structure of the presented gadgets follows the structure of the attacked gadgets, even if it only has two shares and two replications (the computation of c_1 is omitted for brevity). The attack places a fault on a_0^0 which, due to the structure of the gadget, propagates to the output c_0^0 conditioned on v_0^0 and v_1^0. In particular, the fault only propagates to the output if $v_0^0 \neq v_1^0$ which is equivalent to $b = 1$. By observing both c_0^0 and c_0^1 the adversary can distinguish between effective and ineffective faults in c_0^0 and hence, the secret b is leaked to $\mathcal{A}_c$.

In the following, we discuss the details when attacking real-sized gadgets. For this, our explanation focuses on HPC_1^{C}. However, the attack is also applicable to the remaining gadgets. In particular, attacking HPC_1^{I} is possible without any changes while for the attack against HPC_2^{C} the probe placement has to be adjusted.

Algorithm 1: HPC_1^{C}: CINI multiplication.

function $\text{HPC}_1^{\text{C}}(a_0^0, \ldots, a_d^n, b_0^0, \ldots, b_d^n)$:
  Require: $n = 2k + 1$
  Require: $a_i^\ell = a_i^{\ell'}$ and $b_i^\ell = b_i^{\ell'}$ for $0 \leq \ell, \ell' \leq n, 0 \leq i < d$
  Require: $\sum_{j=0}^{d} a_j^\ell = a$ and $\sum_{j=0}^{d} b_j^\ell = b$ for $0 \leq \ell < n$
  `// Initialize randomness`
  for $i = 0$ *to* d **do**
    for $j = i + 1$ *to* d **do**
      $\tilde{r}_{i,j} \xleftarrow{\$} \mathbb{F}_2; \tilde{r}_{j,i} \leftarrow \tilde{r}_{i,j}$
      $r_{i,j} \xleftarrow{\$} \mathbb{F}_2; r_{j,i} \leftarrow r_{i,j}$
  `// Refreshing`
  for $\ell = 0$ *to* $n - 1$ **do**
    for $j = 0$ *to* d **do**
      $\tilde{v}_j^\ell \leftarrow b_j^\ell + \sum_{i=0, i\neq j}^{d} \tilde{r}_{i,j}$
  `// Correction`
  for $\ell = 0$ *to* $n - 1$ **do**
    for $i = 0$ *to* d **do**
      for $j = 0$ *to* d **do**
        $v_{i,j}^\ell \leftarrow \text{maj}(\tilde{v}_i^0 \ldots \tilde{v}_i^{n-1})$
  `// Multiplication`
  for $\ell = 0$ *to* $n - 1$ **do**
    for $i = 0$ *to* d **do**
      $w_i^\ell \leftarrow a_i^\ell \cdot \text{reg}[v_{i,i}^\ell]$
      for $j = 0$ *to* $d, j \neq i$ **do**
        $z_{i,j}^\ell \leftarrow a_i^\ell \cdot \text{reg}[v_{j,i}^\ell] + r_{i,j}$
      $c_i^\ell \leftarrow \text{reg}[w_i^\ell] + \sum_{j=0; j\neq i}^{d} \text{reg}[z_{i,j}^\ell]$
  Ensures: $c_i^\ell = c_i^{\ell'}$ for $0 \leq \ell, \ell' \leq n, 0 \leq i \leq d$
  Ensures: $\sum_{i=0}^{d} c_i^\ell = a \cdot b$ for $0 \leq \ell \leq n$
  return $c_0^0, \ldots, c_d^n$

4.1 Attack on HPC_1^{C}

Let us consider the HPC_1^{C} gadget described in Algorithm 1. The gadget is supposed to provide security against k-bit faults and d-probing adversary even under composition. For our attack, we inject a 1-bit fault at one of the input shares of a. Assume that a_i^ℓ has been corrupted which belongs to the SRD (i, ℓ). The fault propagates through the gadget and, following CINI, corrupts only the output belonging to SRD (i, ℓ). However, the propagation is conditioned on the secret b.

The reason is that the fault-propagation path consists of addition (xor) and multiplication (and) gates and, hence, the fault propagation follows the principles illustrated in 2b and Figure 2c. Therefore, the fault propagation is unaffected by additions and conditioned when passing through a multiplication, leaking the propagation condition when $\mathcal{A}_c$ can distinguish between an effective/ineffective fault. We have highlighted the fault-propagation path in Algorithm 1 (initial fault in red, conditioned variables in blue, and fault propagation in green).

Due to the multiplication, the fault propagation to w_i^ℓ, $z_{i,j}^\ell$, and c_i^ℓ is conditional on the corresponding $v_{i,j}^\ell$. The algebraic expression at

the output with a fault in a_i^ℓ (noted as $\hat{a}_i^\ell$) can be reduced as follows:

$$\begin{aligned} c_i^\ell &\leftarrow \hat{a}_i^\ell \cdot (b_i^\ell + \sum_{h=0,h\neq i}^{d} \tilde{r}_{h,i}) + \sum_{j=0;j\neq i}^{d} (\hat{a}_i^\ell \cdot (b_j^\ell + \sum_{h=0,h\neq j}^{d} \tilde{r}_{h,j}) + r_{i,j}) \\ &\leftarrow \hat{a}_i^\ell \cdot (b_i^\ell + \sum_{h=0,h\neq i}^{d} \tilde{r}_{h,i} + \sum_{j=0;j\neq i}^{d} (b_j^\ell + \sum_{h=0,h\neq j}^{d} \tilde{r}_{h,j})) + \sum_{j=0;j\neq i}^{d} r_{i,j} \\ &\leftarrow \hat{a}_i^\ell \cdot (b_i^\ell + \sum_{j=0;j\neq i}^{d} (b_j^\ell + \tilde{r}_{j,i} + \sum_{h=0,h\neq j}^{d} \tilde{r}_{h,j})) + \sum_{j=0;j\neq i}^{d} r_{i,j} \\ &\leftarrow \hat{a}_i^\ell \cdot b + \sum_{j=0;j\neq i}^{d} r_{i,j} \end{aligned} \tag{1}$$

Note, that registers have no impact on fault propagation and are, therefore, omitted in the expression above. Due to the fact that all (refreshed) shares of b are combined in a single c_i^ℓ, the refreshing of b is completely removed at this point. Under normal execution, this is not problematic, since the refreshing of the partial products remains untouched. However, the rearrangement of the terms shows that the fault propagation is conditioned on the secret b. Still, observing c_i^ℓ only does not directly leak the condition, since $\mathcal{A}_c$ cannot distinguish between effective and ineffective faults, i.e., whether the fault indeed propagates to the output or not.

To get the additional leakage, $\mathcal{A}_c$ needs to observe the (potentially) faulty c_i^ℓ and a (definitely) correct $c_i^{\ell'}$ from another replication domain. This can be achieved when gadgets are composed. Imagine a composition of two $\mathsf{HPC}_1^{\mathsf{C}}$ gadgets, such that the output of the first gadget is the input b of the second gadget (cf. Figure 7). Now, we place a probe at $\tilde{v}_i^\ell$ of the second gadget, which, due to glitch extension, expands to all replication domains of the input. These are all the replication domains of the output share domain i of the first gadget and c_i^ℓ and $c_i^{\ell'}$ are in the extended probes. Hence, with this probe, $\mathcal{A}_c$ can distinguish effective and ineffective faults in the above attack scenario. This leaks the secret b to $\mathcal{A}_c$, violating the privacy of CINI. This attack always requires one probe and one fault regardless of the number of shares and duplications used in the gadget.

Attack in the Standard Model. The above attack is also possible in the standard probing model, i.e., a probe leaks only the value carried by the probed wire. The reason is a dependency in the correction logic as depicted in Figure 6. We note that [37] also mentions a similar kind of leakage for DOM and TI gates, and DOM-based S-Boxes. In particular, without any input fault to the correction, the internal values of the correction are randomly distributed (since the inputs are randomly distributed). However, if there is a faulty input, some internal wires are always zero. Hence, again the secret-dependent fault propagation is needed to simulate the respective probe.

Impact of the Fault Model. The presented attack depends on the structural features of the circuit and works for all common fault models, i.e., set/reset and biased/unbiased bit-flips. Combined security is a notion of perfect security, in the sense that already a small bias towards the adversary is considered insecure. While the fault model impacts the condition of fault propagation (e.g., a set fault adds a condition dependent on the faulted value), it does not change the fact that the attack is possible for some value combinations. This already is a violation of combined security even if this may only marginally impact the success probability of a practical attack.

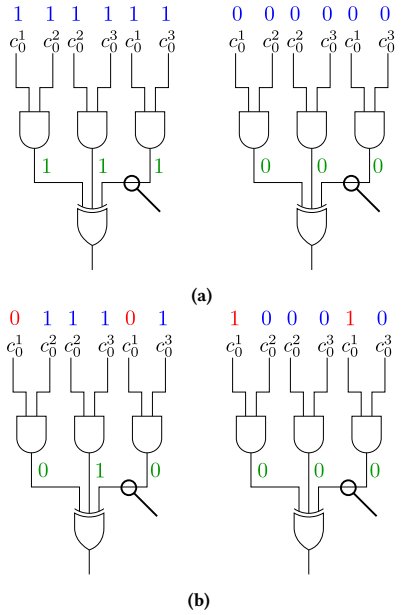

Figure 6. Illustration of leakage from single-bit error correction logic. If the codewords are correct, the outputs of the AND gates toggle with probability 0.5. If there is an error in the codeword, the probed wire gets stuck to zero.

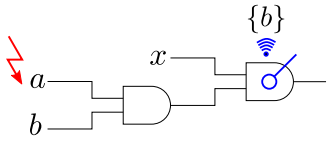

Figure 7. Composition of two $\mathsf{HPC}_1^{\mathsf{C}}$ gadgets. The fault is injected at the input of the left gadget and the leakage is measured from the error correction logic of the right gadget.

Impact of the Attack. Multiple recent works have addressed fault-propagation based leakage [37, 40]. While the fundamental cause of those attacks is the same as in the presented attack, none of them considers CA-secure gadgets, but target specific S-Box constructions. From that perspective, attacking gadgets is more generic, impacting all designs derived by compositions. Several techniques can be utilized to make the leakage exploitable. One option is to use a template attack as shown in [37, 38]. In a recent work [40], it has been shown that such leakages can be exploited even in a non-profiled setting with multi-bit imprecise and random faults. The attacks in [40] exploit the bit-slice structure of GIMLI permutation to enable multi-bit fault attacks. Given that bit slicing is also very common in gadget-based constructions (for software) [21], it is expected that such attacks would also apply in such a context.

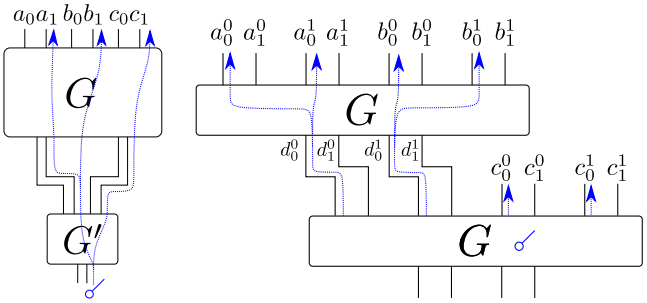

Figure 8. Probe propagation in the case of PINI and CINI.

5 DISCUSSION OF THE ATTACK

In the following, we discuss the core flaw of the CINI proof provided by Feldtkeller et *al.* [19] and give a high-level intuition about the possible fixes.

5.1 The Concept of Probed Share Domains

The concepts of PINI and CINI distinguish between probes placed internally to a gadget and probes placed outside of it. Specifically, probes within a gadget are considered as *normal* probes placed on individual wires. Those probes reveal the (glitch-extended) values of the probed wires to the adversary. In contrast, *external* probes are placed at share domains, leaking all values belonging to the probed share domain. This separation allows the trivial composition of PINI/CINI gadgets, as long as each gadget ensures the isolation of share domains to probe propagation. This is proven by the respective theorems for composition [12, 19]. Hence, the underlying concept is free probe propagation within share domains, but a strict isolation between them.

When combining individual gadgets, the separation of internal and external probes is essential to prove the isolation of share domains. In particular, a single external probe, probing an output share domain, can result in the requirement to simulate multiple output wires. Failing to do so, may lead to a gadget that is probing secure in itself, however, insecure under composition, i.e., the gadget is not PINI/CINI.

For PINI, the scenario that more wires need to be simulated than there are placed probes only occurs for multiple-output gadgets (otherwise there is only one output wire per share domain). In Figure 8a we show an exemplary gadget composition, where a single probe propagates to multiple outputs of a multiple-output gadget. This clearly shows that to be first-order secure, the two outputs belonging to share domain 1 need to be simulated at once. For CINI already a single-output gadget requires the simulation of more wires than placed probes since the replication of a wire explicitly belongs to the same share domain (cf. Definition 3.1). Intuitively this makes sense since a (non-faulted) replication of a wire contains the same information. Again, we give an exemplary gadget composition in Figure 8b. Indeed, a probe at the output of any (not otherwise protected) correction circuit contains all replications of the corrected value, by glitch extension and probe propagation. Our attack in Section 4 exploits exactly this.

5.2 Fixing VERICA

With the introduction of the CINI notion [19], the authors also provided an extension to the formal verification framework VERICA [34] and analyzed several instantiations of corresponding multiplication gadgets. However, the above-introduced flaws have not been detected by VERICA since external probes have not been extended to all wires with the same share domain, i.e., to all replications. In the following, we briefly describe the structure of VERICA and present our fixes leading to correct verifications of the CINI and CINI_{ind} security notions.

VERICA is partitioned into different passes starting with a parsing phase that is responsible to read the used cell library and the netlist under test implementing the target design. Afterwards, a preprocessing phase is executed preparing the circuit model for the verification process. This also includes a strategy that prepares the tool for SCA verifications determining among other things all valid *probe positions* and creating all valid *probe combinations* of up to d probes. However, all probe positions are treated in the same fashion to compute the probe combinations and no particular rules for external probes are applied. Hence, if an external probe has been added to the set of a valid probe combination, all outputs with the same share domain have not been considered in this particular set. To this end, we extended the corresponding function in VERICA by checking if a probe combination contains an external probe and adding all outputs with the same share domain as *virtual probes* to this set (cf. Section 5.1). This distinction is necessary due to the underlying structure and functional principle of VERICA. In the verification process, the number of *original probes* is used to compute a threshold determining the maximum number of input shares that can be used for simulation. However, the original probes plus the virtual probes are considered to compute all possible combinations to check statistical independence to the corresponding input shares as introduced by Knichel et *al.* [25] together with the verification framework SILVER. With these changes VERICA can detect the flaw in the gadgets from Feldtkeller et *al.* [19] (cf. Table 2). We integrated our fix into VERICA[2].

5.3 The Missing Design Principle

The construction of the gadgets proposed by Feldtkeller et *al.* [19] are based on a simple design principle: Every intermediate value that crosses an SRD border needs to be refreshed (to stop probe propagation) and corrected (to stop fault propagation). While necessary, our attack shows that this design principle is not sufficient. In particular, the principle does not prevent the recombination of secrets due to conditional fault propagation (which is not stopped by refreshing). In HPC_1^{C} all shares of the secret b are recombined (after refreshing and correction) within the SRD (i, ℓ), as shown by Equation 1. Then, a fault propagation within this share domain is conditioned on the recombined secret b. Indeed, this is allowed under the *correction* property of CINI. However, the *privacy* property requires the simulation of all output wires belonging to the share domain i (cf. Section 5.1), which will fail due to the fault propagation dependent on b.

Hence, the design principle needs to be extended as follows: *A fault shall never be conditioned on a recombined secret value (even*

[2] https://github.com/Chair-for-Security-Engineering/VERICA

after refreshing). In non-linear operations, like multiplication, the fault propagation is always conditioned (cf. Figure 2c). Therefore, it is required that a fault propagating through a non-linear gate does not propagate to a point in the circuit, where a secret is recombined.

This additional design principle was not discovered by Feldtkeller et *al.* [19] due to a mistake in the CINI proof of their gadgets. When looking closely at the proofs, it is apparent that an external probe does *not* capture all replications of the output. Instead, only a single wire is selected and simulated for this probe, leading always to the same amount of wires probed and simulated. As discussed above, this is not sufficient for CINI and the resulting gadgets are therefore insecure in composition, precisely because of the lack of considering the above design principle.

5.4 Fixing Gadgets

In Section 3.3.3 we discuss how a fault can impact probing security. Intuitively, each type of impact requires a different handling to counteract them. The first type, i.e., the removal of randomness, can be handled via additional randomness, registers, or an increase in the number of shares. Handling the leakage via the observation of effective/ineffective faults is more difficult. In particular, fault propagation remains unaffected by registers and the addition of randomness. Indeed, the (conditioned) propagation only depends on the logical expression of the propagation path. The only effective way to stop fault propagation is via error correction.

Therefore, we see two fundamental ways to adhere to the above design principle. (i) Inserting a correction module after each cross–domain partial-product computation ensures that the fault propagation is stopped before the recombination of the secret b happens. We provide more details on this approach in Section 6. (ii) Via non-completeness in the compression of the gadget, we can ensure that the secret b is never recombined within the gadget. Hence, fault propagation cannot be dependent on b. More details follow in Section 7.

6 GADGETS BASED ON CORRECTION

The core vulnerability exploited in Section 4 is the reduction in Equation 1, i.e., all shares of b are implicitly recombined. Adding additional error-correction modules after the refreshing of the partial products belonging to cross-domains breaks this dependency in case of fault injection. In particular, a fault in a_i^ℓ (noted as $\hat{a}_i^\ell$) does not propagate to the refreshed cross-domain $a_i^\ell b_j^\ell$ but only to the product $a_i^\ell b_i^\ell$. Equation 2 shows that in this case, the different shares do not recombine to the secret b.

$$\begin{aligned} c_i^\ell &\leftarrow \hat{a}_i^\ell \cdot (b_i^\ell + \sum_{h=0,h\neq i}^{d} \tilde{r}_{h,i}) + \sum_{j=0;j\neq i}^{d} \mathsf{maj}(\hat{a}_i^\ell \cdot (b_j^\ell + \sum_{h=0,h\neq j}^{d} \tilde{r}_{h,j}) + r_{i,j}) \\ &\leftarrow \hat{a}_i^\ell \cdot (b_i^\ell + \sum_{h=0,h\neq i}^{d} \tilde{r}_{h,i}) + \sum_{j=0;j\neq i}^{d} a_i^\ell \cdot (b_j^\ell + \sum_{h=0,h\neq j}^{d} \tilde{r}_{h,j}) + r_{i,j} \qquad (2) \\ &\leftarrow \hat{a}_i^\ell \cdot (b_i^\ell + \sum_{h=0,h\neq i}^{d} \tilde{r}_{h,i}) + a_i^\ell \cdot \sum_{j=0;j\neq i}^{d} (b_j^\ell + \sum_{h=0,h\neq j}^{d} \tilde{r}_{h,j}) + \sum_{j=0;j\neq i}^{d} r_{i,j} \end{aligned}$$

6.1 Combined-Isolating Non-Interference

Structure. We provide our CINI multiplication gadget for arbitrary order in Algorithm 2. The structure is similar to the vulnerable gadgets proposed by Feldtkeller et al. [19] but with additional error correction. First, there is a refresh and error correction for all shares belonging to input b. Due to the non-linearity of the multiplication, this is necessary to isolate the propagation of probes and faults within share domains and SRDs, respectively. Afterward, all partial products are computed, where cross-domain partial products are refreshed and corrected again. Finally, the partial products are compressed to reduce the number of output shares to the required number. For the error correction, it is necessary to do this computation in a replicated manner.

Interestingly, the additional correction modules after the partial-product computation allow us to optimize the gadget for the number of registers. In particular, the previous gadget required that each output of the correction (after the refresh of b, cf. Algorithm 1) is only used in one SRD, meaning that a correction and register was required for every partial product. However, in our construction, all cross-domain partial products are corrected again, meaning that a fault after the correction of b can only propagate to a single SRD even when only computed once. Hence, while the number of corrections remains the same we reduce the number of registers per redundancy domain from $2d^2 + 4d + 2$ (in $\mathsf{HPC}_1^{\mathsf{C}}$) to $d^2 + 3d + 2$ (in $\mathsf{CPC}_1^{\mathsf{C}}$). Due to the large area footprint of registers, this optimization leads to significant savings (cf. Section 8.1).

Formal Arguments. We continue by providing a formal argument for the CINI property of Algorithm 2. The core idea for *correctness* is that each share of input a only influences the same SRD of the output, while all shares of b are corrected before reaching the output. For *privacy* the core idea is that probes only propagate until they reach the intermediate values $z_{i,j}^\ell$ or v_j^ℓ. The reason is that the masking of those values is refreshed by $r_{i,j}$ and $\tilde{r}_{i,j}$, respectively. Of course, the probe propagation stops only at those values if the used randomness is not observed anywhere else in the probes and is not faulted. However, if that is the case, there are enough probes and internal faults to allow the probe propagation to the associated input shares. In addition, any secret-dependent fault propagation is carefully controlled to be either corrected before observation or independent of secrets.

Theorem 6.1. *The gadget* $\mathsf{CPC}_1^{\mathsf{C}}$ *as defined in Algorithm 2 with a register-free majority function is* (d,k)*-CINI in the glitch-robust probing model.*

Proof Structure. For the proof, we first show the correctness and then the privacy of the gadget. For privacy, we construct a simulator in the following two steps: (i) We define for each probe/fault combination the input shares the simulator is allowed to access, i.e., the inputs the probes can propagate to. This is done via Algorithm 3 and needs to adhere to the restrictions given in Definition 3.6. (ii) We define how the simulator computes the required intermediate values. Finally, we show that the constructed simulator has the same output distribution as the probes in the original gadget, i.e., both are indistinguishable.

Algorithm 2: $\mathsf{CPC}_1^{\mathsf{C}}$: CINI multiplication.

1 **function** $\mathsf{CPC}_1^{\mathsf{C}}(a_0^0, \ldots, a_d^n, b_0^0, \ldots, b_d^n)$:
  Require: $n = 2k$
  Require: $a_i^\ell = a_i^{\ell'}$ and $b_i^\ell = b_i^{\ell'}$ for $0 \le \ell, \ell' \le n, 0 \le i \le d$
  Require: $\sum_{j=0}^{d} a_j^\ell = a$ and $\sum_{j=0}^{d} b_j^\ell = b$ for $0 \le \ell \le n$
  // Initialize randomness
2 **for** $i = 0$ *to* d **do**
3 **for** $j = i + 1$ *to* d **do**
4 $\tilde{r}_{i,j} \xleftarrow{\$} \mathbb{F}_2$; $\tilde{r}_{j,i} \leftarrow \tilde{r}_{i,j}$
5 $r_{i,j} \xleftarrow{\$} \mathbb{F}_2$; $r_{j,i} \leftarrow r_{i,j}$
  // Refreshing
6 **for** $\ell = 0$ *to* n **do**
7 **for** $j = 0$ *to* d **do**
8 $\tilde{v}_j^\ell \leftarrow b_j^\ell + \sum_{i=0, i\neq j}^{d} \tilde{r}_{i,j}$
  // Correction
9 **for** $\ell = 0$ *to* n **do**
10 **for** $j = 0$ *to* d **do**
11 $v_j^\ell \leftarrow \mathsf{maj}(\tilde{v}_j^0 \ldots \tilde{v}_j^{n-1})$
  // Multiplication
12 **for** $\ell = 0$ *to* n **do**
13 **for** $i = 0$ *to* d **do**
14 $w_i^\ell \leftarrow a_i^\ell \cdot \mathsf{reg}[v_i^\ell]$
15 **for** $j = 0$ *to* $d, j \neq i$ **do**
16 $w_{i,j}^\ell \leftarrow a_i^\ell \cdot \mathsf{reg}[v_j^\ell]$
17 $\tilde{z}_{i,j}^\ell \leftarrow w_{i,j}^\ell + r_{i,j}$
  // Correction and Compression
18 **for** $\ell = 0$ *to* n **do**
19 **for** $i = 0$ *to* d **do**
20 **for** $j = 0$ *to* $d, j \neq i$ **do**
21 $z_{i,j}^\ell \leftarrow \mathsf{maj}(\tilde{z}_{i,j}^0 \ldots \tilde{z}_{i,j}^{n-1})$
22 $c_i^\ell \leftarrow \mathsf{reg}[w_i^\ell] + \sum_{j=0; j\neq i}^{d} \mathsf{reg}[z_{i,j}^\ell]$
  Ensures: $c_i^\ell = c_i^{\ell'}$ for $0 \le \ell, \ell' \le n, 0 \le i \le d$
  Ensures: $\sum_{i=0}^{d} c_i^\ell = a \cdot b$ for $0 \le \ell \le n$
23 **return** $c_0^0, \ldots, c_d^n$

Algorithm 3: Share-Domain Chooser for the Simulator of $\mathsf{CPC}_1^{\mathsf{C}}$.

1 **function** $\mathsf{DomainChooser}^{\mathsf{HPC}_1^{\mathsf{C}}}(\mathcal{P}, \mathcal{S}_2, \mathcal{F}_2)$:
2 $\mathcal{X} \leftarrow \emptyset$
3 **for** $\ell = 0$ *to* n **do**
4 **for** $i = 0$ *to* d **do**
5 **if** $w_i^\ell \in \mathcal{P}$ *or* $i \in \mathcal{S}_2$ **then**
6 $\mathcal{X} \leftarrow \mathcal{X} \cup \{i\}$
7 **for** $j = 0$ *to* $d, j \neq i$ **do**
8 **if** $v_i^\ell \in \mathcal{P}$ **then**
9 $\mathcal{X} \leftarrow \mathcal{X} \cup \{i\}$
10 **if** $z_{i,j}^\ell \in \mathcal{P} \wedge (i \in \mathcal{X}$ *or* $j \in \mathcal{X})$ **then**
11 $\mathcal{X} \leftarrow \mathcal{X} \cup \{i, j\}$
12 **else if** $z_{i,j}^\ell \in \mathcal{P}$ **then**
13 $\mathcal{X} \leftarrow \mathcal{X} \cup \{i\}$
14 **if** $\tilde{v}_i^\ell \in \mathcal{F}_2$, $v_i^\ell \in \mathcal{F}_2$ *or* $w_{j,i^{\ell \in \mathcal{F}_2}}$ **then**
15 $\mathcal{X} \leftarrow \mathcal{X} \cup \{i\}$
16 **if** $(i \in \mathcal{X}$ *or* $j \in \mathcal{X})$ **then**
17 **if** $\tilde{r}_{i,j} \in \mathcal{F}_2$ *or* $r_{i,j} \in \mathcal{F}_2$ **then**
18 $\mathcal{X} \leftarrow \mathcal{X} \cup \{i, j\}$
19 **return** $\mathcal{X}$

Proof. Let $\mathcal{F}_1$ be a set of k_1 SRDs and $\mathcal{S}_2$ a set of d_2 share domains. Further, let there be k_2 faults injected to the gates of the gadget and d_1 probes placed on internal wires. We first show the correctness and then the privacy of the gadget.

Correctness: For correctness, faults in a random value $\tilde{r}_{i,j}, r_{i,j}$ can be ignored, as otherwise, the correctness without any fault would depend on the concrete value of the randomness and the gadget would output a wrong result half of the time.

By construction of the gadget, the values a_i^ℓ, w_i^ℓ, $z_{i,j}^\ell$, and c_i^ℓ are only used for the computation of c_i^ℓ. Hence, faults injected to those values can only influence the SRD (i, ℓ).

Of the values influencing more than one SRD, b_j^ℓ and $\tilde{v}_j^\ell$ are corrected in Line 11 while v_j^ℓ and $\tilde{z}_{i,j}^\ell$ are corrected in Line 21. Those corrections always work properly, as at most k values can be faulted. Hence, a fault in v_j^ℓ can only affect the SRD (j, ℓ), since cross-domain products are corrected. Otherwise, the only way how a fault in those values can have an impact on an SRD (i, ℓ), with $i \neq j$, is by manipulating the computation of $z_{i,j}^\ell$ in which case there is an additional fault for SRD (i, ℓ).

In conclusion, each fault in an SRD at the input can cause a fault at the same SRD at the output. In addition, the set $\mathcal{F}_2$ is the union of all SRDs (i, ℓ) such that the computation of an intermediate value containing the indices i and ℓ was faulted. Then it holds that $|\mathcal{F}_2| \le k_2$ as each faulty intermediate value can influence at most one SRD. As there are $2k+1$ repetitions of each output, the decoding gadget G^{D} can be constructed as a majority function.

Privacy: Without loss of generality, we restrict the probes to only capture v_j^ℓ, w_i^ℓ, $z_{i,j}^\ell$, and c_i^ℓ as other glitch-extended probes are less powerful. In particular, all probes within a majority function are less powerful than probes on v_j^ℓ or $z_{i,j}^\ell$, respectively, as the majority function is implemented register-free.

Given a set of probes $\mathcal{P}$, a set of share domains $\mathcal{S}_2$, and a set of faults $\mathcal{F}_2$ Algorithm 3 returns a set of share domains $\mathcal{X}$, required for the simulation of the probes and the outputs belonging to $\mathcal{S}_2$.We set $\mathcal{S}_1 \leftarrow \mathcal{X} \setminus \mathcal{S}_2$.

First, we see that Algorithm 3 adds at most one share domain per probe, share domain in $\mathcal{S}_2$, and internal fault. Therefore, it always holds that $|\mathcal{S}_1| \le d_1 + k_2$. We continue, by showing that the share domains in $\mathcal{X}$ and knowledge of the faults are sufficient to simulate the probes $\mathcal{P}$ and outputs belonging to $\mathcal{S}_2$.

We define a simulator that computes all required values exactly as defined in Algorithm 2 (required randomness is generated), except for $\tilde{v}_j^\ell$ and $z_{i,j}^\ell$. For these values, we distinguish between the following cases: (i) if $j \in \mathcal{X}$ compute $\tilde{v}_j^\ell$ and $z_{i,j}^\ell$ according to Algorithm 2. (ii) if $j \notin \mathcal{X}$ then draw two fresh random values $\tilde{r}, r$ and set

$\forall i, \ell : \tilde{v}_j^\ell \leftarrow \tilde{r}$ and $z_{i,j}^\ell \leftarrow r$. Afterward, all values are manipulated according to the given faults.

This simulator results in the same output distribution as the probes for the following reason: All values are computed exactly as in the gadget (Algorithm 2) except for $\tilde{v}_j^\ell$ and $z_{i,j}^\ell$ in Case (ii). In this case, we argue that $\tilde{r}_{i,j}$ and $r_{i,j}$ are only observable through one intermediate value and, hence, the simulation is correct. Please note, if $i \notin \mathcal{X}$ then there is no need to compute $z_{i,j}^\ell$.

First, assume $\tilde{v}_j^\ell \leftarrow \tilde{r}$. We show that $\forall i : \tilde{r}_{i,j}$ is only observable through either $\tilde{v}_j^\ell$ or $\tilde{v}_i^\ell$. The randomness $\tilde{r}_{i,j}$ impacts the computation of $w_i^{\ell'}, w_j^{\ell'}, v_j^{\ell'}, v_i^{\ell'}, z_{i,j}^{\ell'}, z_{j,i}^{\ell'}, c_i^{\ell'}$, and $c_j^{\ell'}$ for all ℓ'. From $\tilde{v}_j^\ell \leftarrow \tilde{r}$ it follows that some probe is dependent on $\tilde{v}_j^\ell$ and $j \notin \mathcal{X}$. Hence, with Algorithm 3, it follows that $\forall \ell' : w_j^{\ell'}, v_j^{\ell'}, z_{j,i}^{\ell'} \notin \mathcal{P}$, and $j \notin \mathcal{S}_2$, but $\exists \ell'$ such that exactly one of $z_{i,j}^{\ell'} \in \mathcal{P}$ or $i \in \mathcal{S}_2$. Therefore, it holds that $j \notin \mathcal{X} \wedge i \in \mathcal{X}$. Assume $z_{i,j}^{\ell'} \in \mathcal{P}$, then $\forall \ell'' : v_i^{\ell''}, w_i^{\ell''}, z_{j,i}^{\ell''} \notin \mathcal{P}$ as otherwise $i \in \mathcal{X}$. Hence, $\tilde{r}_{i,j}$ is only observable through $\tilde{v}_j^\ell$ in $z_{i,j}^{\ell'}$. In the other case, if $i \in \mathcal{S}_2$ then $\forall \ell' : w_j^{\ell'}, v_j^{\ell'}, z_{j,i}^{\ell'} \notin \mathcal{P}$ (as established above). Hence, $\tilde{r}_{i,j}$ is only observable through $\tilde{v}_i^\ell$.

When $j \notin \mathcal{X} \wedge i \in \mathcal{X}$ it also holds that $\tilde{r}_{i,j}, r_{i,j}, \tilde{v}_j^\ell, v_j^\ell, w_{i,j}^\ell \notin \mathcal{F}_2$. Further, any fault in some input is either corrected or disconnected to the related probes, while faults on $w_j^{\ell'}, \tilde{z}_{i,j}^{\ell'}, z_{i,j}^{\ell'}$, and $c_i^{\ell'}$ can be injected after $\tilde{v}_j^\ell \leftarrow \tilde{r}$ and faults on $w_{j,i}^\ell$ are disconnected. Therefore, the simulation is also correct under faults.

Second, assume $z_{i,j}^\ell \leftarrow r$. We show that $r_{i,j}$ is only observable through $z_{i,j}^\ell$. The randomness $r_{i,j}$ impacts the values $z_{i,j}^{\ell'}, z_{j,i}^{\ell'}, c_i^{\ell'}$, and $c_j^{\ell'}$ for all ℓ'. From $z_{i,j}^\ell \leftarrow r$ it follows that some probe is dependent on $z_{i,j}^\ell$ and $j \notin \mathcal{X}$. Hence, with Algorithm 3, it follows that $\forall \ell' : z_{j,i}^{\ell'} \notin \mathcal{P}$ and $j \notin \mathcal{S}_2$. It also holds that $z_{i,j}^\ell \notin \mathcal{P}$ as glitch-extended probes require the simulation of all stable inputs to $z_{i,j}^\ell$ and not the value itself. Hence, the only probed values dependent on $r_{i,j}$ are $c_i^{\ell'}, \forall \ell'$ (when $i \in \mathcal{S}_2$) and $r_{i,j}$ is indeed only observable through $z_{i,j}^\ell$.

Again, when $j \notin \mathcal{X} \wedge i \in \mathcal{X}$ then it holds $\tilde{r}_{i,j}, r_{i,j}, \tilde{v}_j^\ell, v_j^\ell, w_{i,j}^\ell \notin \mathcal{F}_2$. Further, faults in inputs, or $\tilde{z}_{i,j}^{\ell'}$ are corrected, faults in $w_i^{\ell'}$ or $w_{j,i}^\ell$ are disconnected, while faults in $z_{i,j}^{\ell'}$ and $c_i^{\ell'}$ can be injected after $z_{i,j}^\ell \leftarrow r$. Therefore, the simulation is also correct under faults.

The two cases $\tilde{v}_j^\ell \leftarrow \tilde{r}$ and $z_{i,j}^\ell \leftarrow r$ are mutually exclusive, since if both $z_{i,j}^\ell \in \mathcal{P}$ and $i \in \mathcal{X}$ than $i, j \in \mathcal{X}$. Therefore, a separate analysis is sufficient. The given simulator shows the privacy of Algorithm 2. □

6.2 Independent Combined-Isolating Non-Interference

In their paper, Feldtkeller et *al.* [19] claim their gadget can be transformed from supporting CINI to supporting CINI_{ind} by adding multiple bits of randomness whenever a sharing is refreshed. In particular, they ensure that no combination of faults can remove a single refresh. This surprisingly easy transformation cannot be upheld when probing entire output share domains. In Section 8.2 we provide the results of a tool-based analysis of this gadget. The problem is that standard and efficient implementations of a majority vote (via sorting networks) can be faulted in such a way that a fault in the input propagates through the correction module.

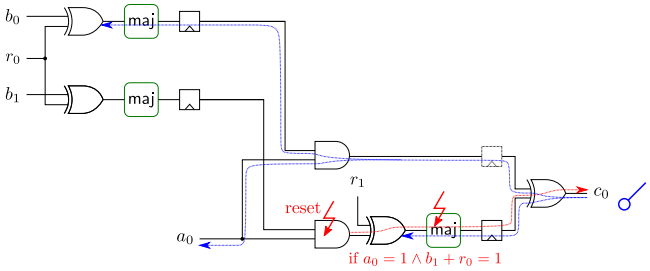

Figure 9. Attack against the trivial extention of CPC_1^{C} to CINI_{ind}.

Such a correction module allows an attack with $d \geq 1$ and $k \geq 2$, i.e., a gadget with at least 2 shares and 5 replications (cf. Figure 9). Assume a reset fault in the computation of $a_0^0 \cdot (b_1^0 + r_0)$, a fault in the corresponding correction module, and a probe on share domain 0. Then, the first fault propagates to the output c_0^0 iff $a_0^0 = 1$ and $b_1^0 + r_0 = 1$. Since the output probe captures all redundant values $c_0^0, \ldots, c_0^4$, the adversary can distinguish between effective and ineffective faults, learning the value of $b_1^0 + r_0$. In addition, the output probe propagates to the intermediate value $b_0^0 + r_0$. Hence, in case of an effective fault $\mathcal{A}_c$ can compute the secret via $b = (b_0^0 + r_0) + (b_1^0 + r_0)$.

A correction module where the output never depends on any input faults even when internal gates are faulted, would prevent this attack. The required property is different than Fault Strong Non-Interference (F-SNI) [15, 34] in that it not only requires an internal fault to allow the propagation of faults to the output but also that the distribution of the output fault is independent of any input-fault distribution. Finding such a correction module is non-trivial and we leave it for further research. Please note that an CINI_{ind} gadget can be trivially constructed via a CINI gadget with appropriate security order, e.g., a $(3,1)$-CINI gadget is always $(2,1)$-CINI_{ind}.

Essentially, the above attack translates two faults to an additional probe and is thus applicable whenever $k \geq 2$. However, in contrast to CINI_{ind}, the CINI notion requires $\mathcal{A}_c$ to trade faults for probes (since $d_1 + d_2 + k_1 + k_2 \leq d$) and, hence, this attack does not apply to the CINI gadgets presented in Section 6.1. If the required two faults are placed in CPC_1^{C} or $\text{CPC}_{\text{NC}}^{\text{C}}$ there is always one probe missing for the attack to work.

7 GADGETS BASED ON NON-COMPLETENESS

Another way to break the recombination of all shares of b in Equation 1 is a *non-complete* compression. Non-completeness is a well-known structure in SCA since it is a critical property for TI [30]. In essence, non-completeness requires each set of up to d probes to be functionally independent of at least one input share. Then, an adversary is never able to learn the input, as at least one share is always missing.

Using non-completeness in the compression of a multiplication gadget means that the partial products are combined in such a way that each set of up to $d-1$ outputs is functionally independent of at least one share of a and one share of b. Hence, in every set of up to

Algorithm 4: $\mathrm{CPC}^{\mathsf{C}}_{\mathsf{NC}}$: $(2,1)$-CINI multiplication.

function $\mathrm{CPC}^{\mathsf{C}}_{\mathsf{NC}}(a^0_0,\ldots,a^2_2,b^0_0,\ldots,b^2_2)$:
  Require: $a^\ell_i = a^{\ell'}_i$ and $b^\ell_i = b^{\ell'}_i$ for $0 \le \ell,\ell' \le 2, 0 \le i \le 2$
  Require: $\sum_{j=0}^{2} a^\ell_j = a$ and $\sum_{j=0}^{2} b^\ell_j = b$ for $0 \le \ell \le 2$
  // Initialize randomness
  for $i = 0$ *to* 6 **do**
    $r_i \xleftarrow{\$} \mathbb{F}_2$
  // Refreshing and correction of b
  for $\ell = 0$ *to* 2 **do**
    $\tilde{v}^\ell_0 = b^\ell_0 + \mathrm{reg}[r_0 + r_1]$
    $\tilde{v}^\ell_1 = b^\ell_1 + r_0;\ \tilde{v}^\ell_2 = b^\ell_2 + r_1$
  for $\ell = 0$ *to* 2 **do**
    for $i = 0$ *to* 2 **do**
      $v^\ell_i \leftarrow \mathrm{maj}(\tilde{v}^0_i \ldots \tilde{v}^2_i)$
  for $\ell = 0$ *to* 2 **do**
    // Partial products and refreshing
    $z^\ell_{0,0} \leftarrow a^\ell_0 \cdot v^\ell_0 + r_3;\ z^\ell_{0,1} \leftarrow a^\ell_0 \cdot v^\ell_1 + r_5$
    $z^\ell_{0,2} \leftarrow a^\ell_0 \cdot v^\ell_2 + r_5;\ z^\ell_{1,0} \leftarrow a^\ell_1 \cdot v^\ell_0 + \mathrm{reg}[r_4 + r_6]$
    $z^\ell_{1,1} \leftarrow a^\ell_1 \cdot v^\ell_1 + r_2;\ z^\ell_{1,2} \leftarrow a^\ell_1 \cdot v^\ell_2 + r_3$
    $z^\ell_{2,0} \leftarrow a^\ell_2 \cdot v^\ell_0 + r_6;\ z^\ell_{2,1} \leftarrow a^\ell_2 \cdot v^\ell_1 + r_4$
    $z^\ell_{2,2} \leftarrow a^\ell_2 \cdot v^\ell_2 + r_2$
    // Compression
    $\tilde{c}^\ell_0 = \mathrm{reg}[z^\ell_{1,1}] + \mathrm{reg}[z^\ell_{1,2}] + \mathrm{reg}[z^\ell_{2,1}]$
    $\tilde{c}^\ell_1 = \mathrm{reg}[z^\ell_{2,2}] + \mathrm{reg}[z^\ell_{0,2}] + \mathrm{reg}[z^\ell_{2,0}]$
    $\tilde{c}^\ell_2 = \mathrm{reg}[z^\ell_{0,0}] + \mathrm{reg}[z^\ell_{0,1}] + \mathrm{reg}[z^\ell_{1,0}]$
  // Correction
  for $\ell = 0$ *to* 2 **do**
    for $i = 0$ *to* 2 **do**
      $c^\ell_i \leftarrow \mathrm{reg}[\mathrm{maj}(\tilde{c}^0_i,\ldots,\tilde{c}^2_i)]$
  Ensures: $c^\ell_i = c^{\ell'}_i$ for $0 \le \ell,\ell' \le 2, 0 \le i \le 2$
  Ensures: $\sum_{i=0}^{2} c^\ell_i = a \cdot b$ for $0 \le \ell \le 2$
  return $c^0_0,\ldots,c^n_2$

$d-1$ outputs $\exists i, j$ such that $\forall i', j', \ell$ the terms $a^\ell_i \cdot b^\ell_{j'}$ and $a^\ell_{i'} \cdot b^\ell_j$ do not occur. We only need $(d-1)$-order non-completeness since the leakage due to the reduction in Equation 1 is crucially dependent on the injection of at least one fault (cf. Section 4), meaning $\mathcal{A}_c$ has at most $d-1$ probes left. When using a $(d-1)$-order non-complete compression, the reduction in Equation 1 is no longer possible, as the expression is missing at least one share of b. Higher-order non-completeness is required to ensure that the remaining probes cannot be used to learn the remaining shares.

Instantiation. A well-known problem with designs based on non-completeness is that the structure of the circuit is highly dependent on the required security order and cannot be expressed in a general manner. Therefore, in the following, we focus on the specific instantiation of a $(2,1)$-CINI gadget. This gadget requires three shares and three replications and is given in Algorithm 4. The general structure is very similar to the previously discussed gadget. First, there is a refresh and correction of input b, which is still required to prevent internal probes and faults from causing cross-domain leakage. Then, the partial products are computed and refreshed, before we apply the non-complete compression. Here, each output c^ℓ_i is functionally independent of the shares a^ℓ_i and b^ℓ_i, which is sufficient since $d = 2$ and we only require first-order non-completeness. Note, that we do not need a correction stage before the compression due to the non-completeness. However, a correction stage is required for each output. The reason is that each input share a^ℓ_i influences two output shares $c^\ell_{i+1\,mod\,3}$, $c^\ell_{i+2\,mod\,3}$. Without correction a fault in a^ℓ_i would propagate from SRD (i,ℓ) to $(i+1\,mod\,3,\ell)$ and $(i+2\,mod\,3,\ell)$, which is prohibited by the correctness property of CINI. Also, an output register is required to prevent the observation of effective/ineffective faults from output probes, which otherwise would propagate to the registers before the compression. We optimized the refreshes, both for b and the partial products, according to Cassiers and Standaert [11][3]. The refresh algorithms from this work can also be used in the previous gadgets. However, like non-interference, they are probing order specific, which is why we used the more general refresh method for our general gadgets. We provide a tool-based security analysis of this gadget in Section 8.2. This gadget should be easily extendable to a $(2,2)$-CINI gadget by increasing the number of replications to five. The only difference between $(2,1)$-CINI and $(2,2)$-CINI is that the latter has to also withstand an attack with two faults (and no probe).

Generalization. The above gadget structure, based on non-completeness, can be scaled to higher-order gadgets. Moving the correction from the partial products to the output reduces the number of corrections from d^2 to $d+1$ per redundancy domain (excluding the corrections required for b). However, for higher-order security, this structure has several drawbacks and challenges to overcome. First, the construction of the non-complete compression requires always at least $2d-1$ shares, which means a significant overhead in area and amount of required randomness. In addition, as already mentioned, there is no general-order description of the algorithm, making the circuit design and security analysis dependent on the implemented security order. Further, due to fault and probe propagation, a fault in one v^ℓ_i leaks all a^ℓ_i that are multiplied with v^ℓ_i and observed by a probe (where randomness does not hide a^ℓ_i due conditional fault propagation). We see three solutions to overcome this issue: (i) Adding additional constraints to the compression to ensure that the number of leaked shares still complies with the constraints from CINI. In fact, this happens in the case of Algorithm 4. However, this may not be possible for all numbers of shares and indeed seems to be impossible for five shares. (ii) Replicate v^ℓ_i not only along fault domains but for each SRD, i.e., compute $\forall i : v^\ell_{j,i}$ for the multiplication with a^ℓ_i. Then a fault in $v^\ell_{j,i}$ affects only a single SRD. However, it is important that both the corresponding correction and register are replicated as well. Hence, this suffers from high overhead in terms of logical gates (for the correction) and registers. In particular, the overhead is dependent on the number of shares, which is higher than usual due to the non-completeness. (iii) Additional registers within the compression function with careful selection of the compression order could decrease the amount of leakage, due to reduced fault and probe propagation. However, this not only increases the area due to the additional registers but also

[3] https://github.com/cassiersg/opt-refresh

Table 1. Implementation and verification results for the fixed CINI and CINI_{ind} gadgets and our new non-complete gadgets synthesized with the 45 nm Open Cell Library.

Gadget	Design						Verification		
	d	k	rand.	comb.	reg.	area [GE]	Def.	(d,k)	Time
CPC_1^{C}	1	1	2	78	18	218	CINI	$(1,1)^{\checkmark}$	0.6 s
	2	1	6	189	36	492		$(2,1)^{\checkmark}$	2.0 s
	3	1	12	348	60	876		$(3,1)^{\checkmark}$	2.6 d
	1	2	2	330	30	630		$(1,2)^{\checkmark}$	2.5 s
	2	2	6	765	60	1 420		$(2,2)^{\checkmark}$	3.2 min
	3	2	12	1380	100	2 527		*	∞
	1	3	2	686	42	1 181		$(1,3)^{\checkmark}$	1.4 h
	2	3	6	1575	84	2 660		*	∞
	3	3	12	2828	140	4 732		*	∞
$\widetilde{\text{CPC}_1^{\text{C}}}$	1	1	4	90	18	242	CINI_{ind}	$(1,1)^{\checkmark}$	0.6 s
	2	1	12	225	36	564		$(2,1)^{\checkmark}$	20.7 min
	1	2	6	370	30	710		$(1,2)^{\times}$	4.5 min
$\text{CPC}_{\text{NC}}^{\text{C}}$	2	1	7	177	48	563	CINI	$(2,1)^{\checkmark}$	6.3 min
	2	2	7	907	80	1 894		*	∞

* Due to the extensive amount of combinations, these gadgets could not be verified with VERICA.

Table 2. Implementation and verification results for the original CINI and CINI_{ind} gadgets synthesized with the 45 nm Open Cell Library. The verification has been conducted with the adapted implementation of VERICA.

Gadget	Design						Verification		
	d	k	rand.	comb.	reg.	area [GE]	Def.	(d,k)	Time
HPC_1^{C}	1	1	2	78	24	252	CINI	$(1,1)^{\checkmark}$	0.6 s
	2	1	6	189	54	600		$(2,1)^{\times}$	1.3 s
	3	1	12	348	96	1 088		*	∞
	1	2	2	330	40	697		$(1,2)^{\checkmark}$	1.8 s
	2	2	6	780	90	1 600		$(2,2)^{\times}$	61.3 s
	3	2	12	1640	160	2 887		*	∞
	1	3	2	700	56	1 270		$(1,3)^{\checkmark}$	43.2 min
	2	3	6	1890	126	2 918		*	∞
	3	3	12	3640	224	5 236		*	∞
HPC_1^{I}	1	1	4	90	24	276	CINI_{ind}	$(1,1)^{\times}$	0.6 s
	2	1	12	225	54	672		$(2,1)^{\times}$	10 min
	3	1	24	420	96	1 232		*	∞
	1	2	6	370	40	777		$(1,2)^{\times}$	3.2 s
	2	2	18	900	90	1 840		*	∞
	3	2	36	1640	160	3 356		*	∞
	1	3	8	784	56	1 438		$(1,3)^{\times}$	4 h
	2	3	24	1848	126	3 423		*	∞
	3	3	48	3360	224	6 244		*	∞

* Due to the extensive amount of combinations, these gadgets could not be verified with VERICA.

the latency. In conclusion, the structure based on non-completeness is prohibitively expensive for higher-order gadgets. While further optimization may change this assessment, we leave this for future work.

8 EVALUATION

In this section, we provide evaluation results for the introduced gadgets. First, we present performance numbers with respect to gadget area and latency. Second, we perform a security analysis using the adapted and corrected version of VERICA.

8.1 Performance

Table 1 provides implementation results for CPC_1^{C} and the $\text{CPC}_{\text{NC}}^{\text{C}}$ gadgets while the corresponding numbers for the original gadgets from Feldtkeller et *al.* [19] are given in Table 2. We re-implemented those gadgets following the algorithms presented in [19] since the netlists provided with the original paper were not functionally correct. We would like to emphasize that the amount of required randomness is the same for CPC_1^{C} and HPC_1^{C} gadgets and does not need to be adapted. Additionally, due to the novel correction strategy (i.e., splitting up the correction and moving part of it from the refreshing of b to the compression), the fixed gadgets require fewer hardware resources with respect to the gate equivalences.

Comparing the CPC_1^{C} to the non-complete gadgets $\text{CPC}_{\text{NC}}^{\text{C}}$, Table 1 shows that the non-complete gadgets perform worse with respect to required randomness and area.

8.2 Security Verification

We performed our analyses on a workstation equipped with an Intel i9-7900X CPU with 20 cores running at 3.3 GHz and 64 GB of RAM. The verification results of CPC_1^{C} and the non-complete $\text{CPC}_{\text{NC}}^{\text{C}}$ gadgets are presented in Table 1. As highlighted, all verified gadgets are secure under the CINI security notion. However, four configurations could not be analyzed due to an extensive amount of combinations that need to be checked by VERICA such that the corresponding verification would not terminate in a reasonable time. For completeness, we also include the verification results for three $\widetilde{\text{CPC}_1^{\text{C}}}$ gadgets, i.e., CPC_1^{C} gadgets with additional randomness analog to the CINI_{ind} gadgets provided by Feldtkeller et *al.* [19]. As shown, the $(1,2)$-gadget does not support the CINI_{ind} notion due to the attack explained in Section 6.2.

In order to demonstrate that our adapted version of VERICA is capable of detecting the flaws in the original gadgets from Feldtkeller et *al.* [19], we show the verification results in Table 2. All HPC_1^{C} gadgets with $d \geq 2$ are verified insecure using our adapted version of VERICA. Additionally, VERICA reports that all verified instantiations of HPC_1^{I} gadgets are insecure. Since the flaws already occur for the smallest instantiations, higher-order gadgets will be insecure as well.

9 CONCLUSION

In this work, we showed an attack against the, to the best of our knowledge, only existing composable gadgets in the context of CA for hardware. Our attack breaks the security of arbitrary-order gadgets under composition with a single probe and fault. We show that the vulnerability results from insufficient handling of output probes, both in the formal proofs and in VERICA. We then developed a new design principle for CINI, which informs two types of gadgets immune to our attack. First, we moved correction modules to a

different location, allowing us to increase the security and optimize the gadget with respect to area consumption. Second, we explored gadgets based on non-completeness. Unfortunately, those gadgets scale poorly to higher security orders. We analyzed all our gadgets with the correct treatment of output probes and, therefore, yielded secure and composable gadgets.

All our gadgets use simple replication to archive fault security. How to design gadgets based on other types of redundancy is still an open research challenge. In addition, the construction of efficient gadgets based on the CINI_{ind} notation remains an open question. In particular, this requires the construction of correction modules that prevent all kinds of secret-dependent fault propagation in all circumstances. We conclude that our refurbished arbitrary order CPC gadgets provide a solid foundation for further research.

ACKNOWLEDGMENTS

François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been co-funded by the EU through the ERC project SWORD (724725), the ERC project BRIDGE (101096871), the Horizon Europe project REWIRE (1010706275) and the Horizon Europe project CONVOLVE (101070374), by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and the project CAVE (510964147), and by the German Federal Ministry of Education and Research BMBF through the project VE-HEP (16KIS1345) and 6GEM (16KISK038). Views and opinions expressed are those of the author(s) 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.

REFERENCES

[1] Anita Aghaie, Amir Moradi, Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, Falk Schellenberg, and Tobias Schneider. 2020. Impeccable Circuits. *IEEE Trans. Computers* 69, 3 (2020), 361–376. https://doi.org/10.1109/TC.2019.2948617

[2] Frédéric Amiel, Karine Villegas, Benoit Feix, and Louis Marcel. 2007. 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*. 92–102. https://doi.org/10.1109/FDTC.2007.4318989

[3] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. 2018. Private Circuits: A Modular Approach. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III (Lecture Notes in Computer Science)*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10993. Springer, 427–455. https://doi.org/10.1007/978-3-319-96878-0_15

[4] Alessandro Barenghi, Guido Bertoni, Luca Breveglieri, Mauro Pellicioli, and Gerardo Pelosi. 2010. Low Voltage Fault Attacks to AES. In *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA*, Jim Plusquellic and Ken Mai (Eds.). IEEE Computer Society, 7–12. https://doi.org/10.1109/HST.2010.5513121

[5] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. 2015. Verified Proofs of Higher-Order Masking. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I (Lecture Notes in Computer Science)*, Elisabeth Oswald and Marc Fischlin (Eds.), Vol. 9056. Springer, 457–485. https://doi.org/10.1007/978-3-662-46800-5_18

[6] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. 2016. Strong Non-Interference and Type-Directed Higher-Order Masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 116–129. https://doi.org/10.1145/2976749.2978427

[7] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. 2016. Randomness Complexity of Private Circuits for Multiplication. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*. 616–648. https://doi.org/10.1007/978-3-662-49896-5_22

[8] Eli Biham and Adi Shamir. 1997. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings (Lecture Notes in Computer Science)*, Burton S. Kaliski Jr. (Ed.), Vol. 1294. Springer, 513–525. https://doi.org/10.1007/BFb0052259

[9] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. 1997. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding (Lecture Notes in Computer Science)*, Walter Fumy (Ed.), Vol. 1233. Springer, 37–51. https://doi.org/10.1007/3-540-69053-0_4

[10] Ran Canetti. 2001. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 136–145. https://doi.org/10.1109/SFCS.2001.959888

[11] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. 2021. Hardware Private Circuits: From Trivial Composition to Full Verification. *IEEE Trans. Computers* 70, 10 (2021), 1677–1690. https://doi.org/10.1109/TC.2020.3022979

[12] Gaëtan Cassiers and François-Xavier Standaert. 2020. Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. *IEEE Trans. Inf. Forensics Secur.* 15 (2020), 2542–2555. https://doi.org/10.1109/TIFS.2020.2971153

[13] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. 1999. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings (Lecture Notes in Computer Science)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, 398–412. https://doi.org/10.1007/3-540-48405-1_26

[14] Christophe Clavier, Benoit Feix, Georges Gagnerot, and Mylène Roussellet. 2010. 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*. 10–19. https://doi.org/10.1109/FDTC.2010.17

[15] Siemen Dhooghe and Svetla Nikova. 2020. My Gadget Just Cares for Me - How NINA Can Prove Security Against Combined Attacks. In *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings (Lecture Notes in Computer Science)*, Stanislaw Jarecki (Ed.), Vol. 12006. Springer, 35–55. https://doi.org/10.1007/978-3-030-40186-3_3

[16] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. 2014. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings (Lecture Notes in Computer Science)*, Phong Q. Nguyen and Elisabeth Oswald (Eds.), Vol. 8441. Springer, 423–440. https://doi.org/10.1007/978-3-642-55220-5_24

[17] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. 2015. Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I (Lecture Notes in Computer Science)*, Elisabeth Oswald and Marc Fischlin (Eds.), Vol. 9056. Springer, 401–429. https://doi.org/10.1007/978-3-662-46800-5_16

[18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. 2018. Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 3 (2018), 89–120.

[19] Jakob Feldtkeller, Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. 2022. CINI MINIS: Domain Isolation for Fault and Combined Security. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 1023–1036.

[20] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. 2001. Electromagnetic Analysis: Concrete Results. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings (Lecture Notes in Computer Science)*, Çetin Kaya Koç, David Naccache, and Christof Paar (Eds.), Vol. 2162. Springer, 251–261. https://doi.org/10.1007/3-540-44709-1_21

[21] Dahmun Goudarzi and Matthieu Rivain. 2017. How Fast Can Higher-Order Masking Be in Software?. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic*

Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I (Lecture Notes in Computer Science), Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.), Vol. 10210. 567–597. https://doi.org/10.1007/978-3-319-56620-7_20

[22] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David A. Wagner. 2006. Private Circuits II: Keeping Secrets in Tamperable Circuits. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings (Lecture Notes in Computer Science)*, Serge Vaudenay (Ed.), Vol. 4004. Springer, 308–327. https://doi.org/10.1007/11761679_19

[23] Yuval Ishai, Amit Sahai, and David A. Wagner. 2003. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings (Lecture Notes in Computer Science)*, Dan Boneh (Ed.), Vol. 2729. Springer, 463–481. https://doi.org/10.1007/978-3-540-45146-4_27

[24] Marc Joye and Michael Tunstall (Eds.). 2012. *Fault Analysis in Cryptography*. Springer. https://doi.org/10.1007/978-3-642-29656-7

[25] David Knichel, Pascal Sasdrich, and Amir Moradi. 2020. SILVER - Statistical Independence and Leakage Verification. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I (Lecture Notes in Computer Science)*, Shiho Moriai and Huaxiong Wang (Eds.), Vol. 12491. Springer, 787–816. https://doi.org/10.1007/978-3-030-64837-4_26

[26] Paul C. Kocher. 1996. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings (Lecture Notes in Computer Science)*, Neal Koblitz (Ed.), Vol. 1109. Springer, 104–113. https://doi.org/10.1007/3-540-68697-5_9

[27] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings (Lecture Notes in Computer Science)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, 388–397. https://doi.org/10.1007/3-540-48405-1_25

[28] Ueli Maurer. 2011. Constructive Cryptography - A New Paradigm for Security Definitions and Proofs. In *Theory of Security and Applications - Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers (Lecture Notes in Computer Science)*, Sebastian Mödersheim and Catuscia Palamidessi (Eds.), Vol. 6993. Springer, 33–56. https://doi.org/10.1007/978-3-642-27375-9_3

[29] Lauren De Meyer, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. 2019. M&M: Masks and Macs against Physical Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 1 (2019), 25–50. https://doi.org/10.13154/tches.v2019.i1.25-50

[30] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. 2006. Threshold Implementations Against Side-Channel Attacks and Glitches. In *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings (Lecture Notes in Computer Science)*, Peng Ning, Sihan Qing, and Ninghui Li (Eds.), Vol. 4307. Springer, 529–545. https://doi.org/10.1007/11935308_38

[31] Dmytro Petryk, Zoya Dyka, and Peter Langendoerfer. 2018. Optical Fault Injections: a Setup Comparison. *RESCUE-Interdependent Challenges of Reliability, Security and Quality in Nanoelectronic Systems Design* (2018).

[32] Emmanuel Prouff and Matthieu Rivain. 2013. Masking against Side-Channel Attacks: A Formal Security Proof. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings (Lecture Notes in Computer Science)*, Thomas Johansson and Phong Q. Nguyen (Eds.), Vol. 7881. Springer, 142–159. https://doi.org/10.1007/978-3-642-38348-9_9

[33] Oscar Reparaz, Lauren De Meyer, Begül Bilgin, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Nigel P. Smart. 2018. CAPA: The Spirit of Beaver Against Physical Attacks. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I (Lecture Notes in Computer Science)*, Hovav Shacham and Alexandra Boldyreva (Eds.), Vol. 10991. Springer, 121–151. https://doi.org/10.1007/978-3-319-96884-1_5

[34] Jan Richter-Brockmann, Jakob Feldtkeller, Pascal Sasdrich, and Tim Güneysu. 2022. VERICA - Verification of Combined Attacks Automated formal verification of security against simultaneous information leakage and tampering. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022, 4 (2022), 255–284. https://doi.org/10.46586/tches.v2022.i4.255-284

[35] Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. 2023. Revisiting Fault Adversary Models - Hardware Faults in Theory and Practice. *IEEE Trans. Computers* 72, 2 (2023), 572–585. https://doi.org/10.1109/TC.2022.3164259

[36] Thomas Roche, Victor Lomné, and Karim Khalfallah. 2011. 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*. 65–83. https://doi.org/10.1007/978-3-642-27257-8_5

[37] Sayandeep Saha, Arnab Bag, Dirmanto Jap, Debdeep Mukhopadhyay, and Shivam Bhasin. 2021. Divided We Stand, United We Fall: Security Analysis of Some SCA+SIFA Countermeasures Against SCA-Enhanced Fault Template Attacks. In *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II (Lecture Notes in Computer Science)*, Mehdi Tibouchi and Huaxiong Wang (Eds.), Vol. 13091. Springer, 62–94. https://doi.org/10.1007/978-3-030-92075-3_3

[38] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. 2020. Fault Template Attacks on Block Ciphers Exploiting Fault Propagation. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I (Lecture Notes in Computer Science)*, Anne Canteaut and Yuval Ishai (Eds.), Vol. 12105. Springer, 612–643. https://doi.org/10.1007/978-3-030-45721-1_22

[39] Sayandeep Saha, Dirmanto Jap, Jakub Breier, Shivam Bhasin, Debdeep Mukhopadhyay, and Pallab Dasgupta. 2018. 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*. IEEE Computer Society, 15–22. https://doi.org/10.1109/FDTC.2018.00011

[40] Sayandeep Saha, Prasanna Ravi, Dirmanto Jap, and Shivam Bhasin. 2023. Non-Profiled Side-Channel Assisted Fault Attack: A Case Study on DOMREP. In *Proceedings of 29th Design, Automation and Test in Europe (DATE) 2023*. IEEE, Antwerp, Belgium, 1–6.

[41] J. M. Schmidt and Michael Hutter. 2007. Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results. In *Proceedings of 15th Austrian Workhop on Microelectronics (Austrochip)*. Verlag der Technischen Universität Graz, Graz, Austria„ 61–67.

[42] Nidhal Selmane, Sylvain Guilley, and Jean-Luc Danger. 2008. Practical Setup Time Violation Attacks on AES. In *Seventh European Dependable Computing Conference, EDCC-7 2008, Kaunas, Lithuania, 7-9 May 2008*. IEEE Computer Society, 91–96. https://doi.org/10.1109/EDCC-7.2008.11