Combined Stability: Protecting against Combined Attacks

Dilara Toprakhisar[1][0000−0003−4551−6775], Svetla Nikova[1][0000−0003−3133−9261], and Ventzislav Nikov[2]

[1] COSIC, KU Leuven, Leuven, Belgium
`firstname.lastname@esat.kuleuven.be`
[2] NXP Semiconductors, Leuven, Belgium
`venci.nikov@gmail.com`

Abstract. Physical attacks pose serious challenges to the secure implementation of cryptographic algorithms. While side-channel analysis (SCA) has received significant attention, leading to well-established countermeasures, fault attacks and especially their combination with SCA (*i.e.,* combined attacks) remain less researched. Addressing such combined attacks often requires a careful integration of masking and redundancy techniques to resist the reciprocal effects of faults and probes. Recent research on combined security has gained momentum, with most approaches relying on composable security notions involving error correction, typically applied after each nonlinear operation. While effective, this approach introduces an area and performance overhead, along with additional security challenges posed by the correction circuits themselves. In this work, we take a different direction, following the concept of stability introduced in StaTI (CHES 2024), which ensures fault propagation to protect against ineffective faults. We extend this concept to combined security by proposing a new composable security notion, *combined stability*, which integrates an extended stability notion, *diffused stability*, with arbitrarily composable glitch-extended probing security notions. Notably, this framework requires only a single error detection at the end of the computation, avoiding costly intermediate error checks and corrections. To demonstrate practicality, we describe a combined secure AES S-box hardware implementation. Our results show that this approach, achieving combined security with competitive implementation costs, offers a promising alternative to error-correction-based schemes.

1 Introduction

While cryptographic algorithms are carefully designed to resist mathematical cryptanalysis, their hardware implementations deployed in physical devices often remain vulnerable to attacks exploiting or manipulating physical characteristics of the implementations. These physical attacks fall into three categories: (i) passive attacks, which observe the physical behavior of the device (*e.g.,* power consumption [KJJ99], timing [Koc96], and electromagnetic emanation [GMO01]),

(ii) active attacks, which deliberately disrupt the computation through physical manipulations (*e.g.*, clock/voltage glitching [AK97], electromagnetic interference [DDRT12], and laser injections [Hab65]), and (iii) combined attacks, which leverage passive and active attack techniques simultaneously.

A widely studied form of passive attacks is side-channel analysis (SCA), where an adversary can exploit the observable leakage arising from the physical characteristics of the implementations. To counter such passive attacks, algorithm-level countermeasures based on formal security models are employed. Among the most established countermeasures against SCA is masking [CJRR99, ISW03, RBN$^+$15, GMK16], which splits the secret data into a number of statistically independent shares. This ensures that observing a leakage from all-but-one share remains independent of the secret data. The theoretical basis for many masking schemes is rooted in the probing model by Ishai *et al.* [ISW03] which allows an adversary to observe up to a limited number of internal values in the circuit. Since evaluating large circuits under the probing model is computationally expensive, the research has focused on composable security notions (*e.g.,* SNI [BBD$^+$16], PINI [CS20a]). These notions establish security properties for smaller circuit components, ensuring that when these components are composed into a larger circuit, the security properties are inherited.

While SCA passively exploits the observable leakage, fault attacks take an active approach by intentionally disrupting the computations through physical fault injection mechanisms. The reaction of the device to this disruption is then exploited to extract secret information. Since the foundational work of Boneh *et al.* [BDL97] that introduces fault attacks on RSA, the field has evolved considerably. To counteract fault attacks, redundancy-based techniques have been widely adopted that detect or correct faults through redundant computations. These detection techniques either discard or randomize the output upon fault detection to prevent exploitation. While SCA has seen the development of several composable security notions, fault attacks have proven more challenging in this regard and remain largely limited to the notion of stability [DOT24, DOT25] to protect against ineffective faults.

Combined attacks, which jointly leverage the capabilities of SCA and fault attacks, have gained increasing attention in recent years. Practical results (*e.g.,* [SBJ$^+$21]) have demonstrated that relying on the naive combination of masking and redundancy is insufficient to protect against combined attacks. Compared to SCA and fault attacks, the attack surface grows even further, making a security analysis of a cryptographic implementation infeasible in practice. As a result, also in the field of combined attacks, the research community has shifted its focus towards composable security notions. Dhooghe and Nikova [DN20b] proposed the first composable security notions for combined attacks. However, their first composable combined secure multiplication gadget relies on intermediate abort mechanisms, making it impractical for hardware implementations. Furthermore, their second proposal was later shown to be flawed by Richter-Brockmann *et al.* [RFSG22]. Feldtkeller *et al.* [FRSG22] introduced the notion of Combined-Isolating Non-Interference (CINI) along with several combined secure multiplication gadgets. However, these gadgets were subsequently demonstrated to be insecure by Feldtkeller *et al.* [FGM+23]. The authors then proposed the refurbished versions of the CINI gadgets. Lastly, Feldtkeller *et al.* [FRSG24] show how to transform TI-like constructions to protect against combined attacks. A crucial observation is that all these countermeasures, which have not been shown to be insecure, or impractical to be implemented in hardware, rely on error correction. However, the naive implementation of majority voting, commonly adopted as an error correction mechanism, has been shown to be flawed for higher-order faults [FGM+23]. As a result, designing a combined secure correction mechanism based on majority voting would entail significantly higher implementation costs. This motivates a reconsideration of error detection as a viable alternative. In this work, we challenge the prevailing belief that error detection is ineffective against combined attacks, a view largely because it was assumed to require intermediate error checks and, thus, abort mechanisms, which are considered impractical in hardware, and that the abort signal inherently leaks information.

Contributions. In this work, we propose a composable combined security notion, *combined stability.* Combined stability extends arbitrarily composable glitch-extended probing security notions by incorporating a new property, which we define as *diffused stability.* Diffused stability advances the notion of stability introduced in StaTI [DOT24] by enforcing the fault propagation at the level of individual output shares, such that any fault-induced deviation from a correct codeword is input-independent. The notion of combined stability, being composable, enables the construction of small secure circuits that can be composed into larger designs while preserving the combined security guarantees. Moreover, it eliminates the need for complex abort mechanisms in hardware implementations, requiring only a single error detection mechanism at the end of the whole computation. This makes combined stability the first notion to achieve combined security without relying on intermediate error detection or correction circuits.

We present a generic transformation for constructing combined stable circuits secure against side channel attacks (SCA), fault attacks, and their combination. We demonstrate this transformation to second-order masking of AND gate over three shares based on Consolidating Masking Schemes [RBN+15], by integrating our diffused stable gadgets implementing addition and multiplication over $\mathbb{F}_{2^n}$. The resulting gadget implementing shared multiplication achieves second-order side channel security, first-order fault security, and protection against single-probe single-fault combined attacks.

This gadget is then used to implement a combined secure AES S-box in hardware. Its implementation cost is compared to the state-of-the-art combined countermeasures. Our results show that our approach proves itself competitive with existing schemes, which typically provide single-probe single-fault combined secure implementations due to the impractical overhead of higher-order fault secure error correction mechanisms. By avoiding intermediate error checks and requiring a single error check, our construction achieves a low overhead factor compared to the implementations that protect only against SCA.

2 Preliminaries

In this section, we introduce the probing, fault and the combined adversary models with their corresponding security models. We then introduce Boolean masking and redundancy as countermeasures against probing and fault adversaries, respectively. Additionally, we introduce the StaTI countermeasure [DOT24], from which we adopt the stability notion. Finally, we provide a brief overview of related work.

2.1 Adversary and Security Models

We consider an adversary with both probing and faulting capabilities in a scenario where these capabilities can be combined. Similar to [ISW03], computations are represented as arithmetic circuits, with the attack surface modeled as a directed acyclic graph (DAG), where vertices correspond to Boolean logic and memory gates, and edges represent wires carrying elements in $\mathbb{F}_{2^n}$. These circuits also integrate randomness, with specific nodes having no input but generating independent and uniformly distributed output elements in $\mathbb{F}_{2^n}$, without affecting the correctness of the circuit. Additionally, we consider nodes without outputs that can be triggered at the end of the computation.

Beyond this, we work with gadgets correctly executing a function $F : \mathbb{F}_q^{k_1} \rightarrow \mathbb{F}_q^{k_2}$. We construct more complex circuits by composing these gadgets, while preserving their individual security properties. The resulting circuit can be viewed as a DAG, where the vertices correspond to the gadgets and the edges define their interconnections. The input and the output of each gadget correspond to the incoming and outgoing edges.

We now go over the probing, faulting, and combined adversary models and the associated security models.

Wire Probing. To capture the probing capabilities of an adversary, we adopt the d-probing model introduced by Ishai *et al.* [ISW03]. In this model, the adversary can observe up to d predetermined wires within a Boolean circuit. To account for the impact of glitches in hardware implementations, we extend the probing model to the glitch-extended robust probing model by Faust *et al.* [FGP+18]. In this model, the adversary is allowed to access not only the probed wires, but also all registered inputs leading to those probed wires.

To analyze the security in the probing model, we follow a simulation model as introduced by Ishai *et al.* [ISW03]. To demonstrate security, a simulator needs to simulate the probed wire values without any access to the secret inputs of the circuit such as the secret key or the plaintext of a block cipher. This proof of simulation security boils down to demonstrating that the distribution of probed wire values remains independent of the secret input.

Gate/Register Faulting. To capture the faulting capabilities of an adversary, we follow the same approach as in StaTI [DOT24] that allows the adversary to alter the outputs of up to k gates or registers using reset, set or bitflip faults.

We assume that the injected faults are non-persistent, meaning, only effective for the current execution. Since each wire is an output of a register or a gate, we interpret wire faults as faults injected at the corresponding registers or gates. This excludes the cases where an adversary cuts the wire connections and alter the specific sets of wires.

By default, a fault refers to a single-bit fault. However, for certain word-level operations, such as field inversion in AES, a single-bit fault can effectively be perceived as a fault on the entire word, such as a byte in the case of AES.

For the security model, we require that the circuit is *correct* and *private*. A circuit is correct against the gate/register faulting adversary if, for all possible faults, it consistently gives either a correct output or an abort signal. A circuit is private if, for every fixed injected fault, the probability of the abort signal is independent of the secret inputs of the circuit. Similar to the probing model, we define a simulation game in which the simulator needs to simulate the abort signal when provided with the injected faults.

Combined Adversary. Following the notions in [DN20b], we consider a combined adversary who has the combination of probing and gate/register faulting capabilities. That is, a combined adversary can (at the same execution) probe up to d wires and fault up to k gates or registers. Then, such an adversary is given the glitch extended probes, the abort signal, and the output of the circuit if the abort signal is not triggered when the circuit is concatenated with an error detection logic. In this work, we consider a first-order probing and first-order faulting combined adversary. That is, we specifically assume an adversary that is capable of probing a single value and faulting a single register or a gate. For the security model, we again consider the correctness and the privacy of the circuit and define it as follows:

Definition 1 (Single-Probe Single-Fault Combined Security). *A circuit C is single-probe single-fault combined secure if for any set of a single probe and a single fault on the circuit's variables, the following holds:*

(a) Correctness: The circuit either aborts or gives a correct output.
(b) Privacy: The probed variable together with the status of the abort signal does not depend on the secrets of the circuit.

2.2 Boolean Masking and Redundancy

We use Boolean masking against probing adversaries, where each variable $x \in \mathbb{F}_{2^n}$ in the circuit is split into s_x shares $\bar{x} = (x_0, x_1, ..., x_{s_x-1})$ such that $x = \sum_{i=0}^{s_x-1} x_i$ over $\mathbb{F}_{2^n}$. Splitting variables enables computations to be performed independently of the secret data. Various secure Boolean masking schemes have been proposed in the literature [ISW03, NRR06, RBN+15, GMK16], each distinguished by how they approach to handling nonlinear operations.

The fundamental approach to protect against fault attacks is to incorporate redundancy, allowing for the detection or correction of the injected faults.

To encode the circuit to protect against fault attacks (specifically first-order), we use duplication as redundancy. Similar to StaTI, we first share the state of the cipher, and then duplicate each share. For example, a bit $x \in \mathbb{F}_2$ is first shared into $x_0, x_1 \in \mathbb{F}_2$ such that $x_0 + x_1 = x$, and then each share is duplicated $\langle x_0^0, x_1^0, x_0^1, x_1^1 \rangle$. In our notation, subscripts indicate the share domain, and superscripts indicate the replication domain.

2.3 StaTI: Protecting against Fault Attacks Using Stable Threshold Implementations

StaTI [DOT24] is a fault countermeasure framework combining threshold implementations [NRR06] and linear encoding techniques. It is built upon the notions originating from threshold implementations: *correctness*, *non-completeness*, and *uniformity*. Correctness ensures that the output is the sharing of the correct output. Non-completeness ensures that each coordinate function operates on data independent of the secret input. Uniformity requires that the function outputs a uniform sharing given the input sharing is uniform. Further details on these notions can be found in [NRR06]. In addition to these notions, StaTI [DOT24] introduces two notions: *stability* and *fault non-completeness* to protect against a single gate/register-faulting adversary when complemented the threshold implementation notions. Stability states that any fault present in the input codeword of a function propagates to the output, ensuring that an incorrect input codeword cannot be matched to a correct output codeword. This notion is particularly effective in preventing the occurrence of ineffective faults. We include the definition from [DOT24]:

Definition 2 (Stability). *Consider a shared and encoded register-to-register function $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a code $\mathcal{C} \subset \mathbb{F}_2^n$. We call $\tilde{F}$* stable *if for any $\tilde{x} \in \mathcal{C}$ and $e \notin \mathcal{C}$, $\tilde{F}(\tilde{x} + e) \notin \mathcal{C}$.*

The stability notion is introduced as a serially composable property, preventing injected faults from becoming ineffective, particularly in serially composed circuits. Later, in StaMAC [DOT25], stability was proven to be arbitrarily composable, making it the first composable notion that protects against ineffective faults without requiring intermediate error checks or corrections.

Fault non-completeness ensures that no gate is shared between two coordinate functions. This prevents faults injected to the gates to affect a single output bit. Relaxed version of this notion ensures that each gate of the circuit drives a non-complete set of output shares, ensuring that any gate fault results in an additive fault at the output which is independent of the secret data.

StaTI proposes stable encodings for XOR and AND gates, as these gates are not inherently injective and therefore lack stability on their own. The stable encodings are defined as follows:

$$\begin{aligned} &\text{XOR} \quad z^0 = a^0 + b^0 + (a^0 + a^1)(b^0 + b^1) \quad z^1 = a^1 + b^1\,, \\ &\text{AND} \quad z^0 = a^0 b^0 + (a^0 + a^1)(b^1 + 1) + (b^0 + b^1)(a^1 + 1) \quad z^1 = a^1 b^1\,. \end{aligned}$$

The extension of XOR gate to the shared domain is trivial, however, a shared AND gate is more complex. StaTI proposes the following two share AND gate:

$$
\begin{array}{ll}
\text{Stage 1} & \text{Stage 2} \\
x_0^0 = a_0^1 b_0^1 + b_0^0 + (a_0^0 + a_0^1)(b_0^0 + b_0^1 + 1) & \\
x_1^0 = a_0^1 b_1^0 + (a_0^0 + a_0^1) + (b_1^0 + b_1^1)(a_0^0 + 1) & z_0^0 = x_0^0 + x_1^0 + (x_0^0 + x_0^1)(x_1^0 + x_1^1) \\
\hline
x_2^0 = a_1^1 b_1^0 + (a_1^0 + a_1^1) + (b_1^0 + b_1^1)(a_1^0 + 1) & z_1^0 = x_2^0 + x_3^0 + (x_2^0 + x_2^1)(x_3^0 + x_3^1) \\
x_3^0 = a_1^1 b_0^1 + b_0^0 + (a_1^0 + a_1^1)(b_0^0 + b_0^1 + 1) & \\
 & \\
x_0^1 = a_0^1 b_0^1 + b_0^1 & \\
x_1^1 = a_0^1 b_1^1 & z_0^1 = x_0^1 + x_1^1 \\
\hline
x_2^1 = a_1^1 b_1^1 & z_1^1 = x_2^1 + x_3^1 \\
x_3^1 = a_1^1 b_0^1 + b_0^1 &
\end{array}
$$

2.4 Related Work

The earliest countermeasures against combined attacks rely on MAC tags and error detection. The CAPA countermeasure, proposed by Reparaz *et al.* [RMB+18], is based on the principles of the Multiparty Computation protocol SPDZ [DPSZ12]. CAPA performs computations on shared data and shared associated MAC tags, combining masking with fine-grained redundancy that employs intermediate error checks. It claims higher-order security against SCA, fault attacks, and their combination. However, its high implementation costs and a recent attack [TNN24] exploiting a single probe and single fault challenge its practicality. M&M [MAN+19] relaxes CAPA's security model by extending any probing-secure masking scheme with MAC tags. However, it does not address ineffective faults, as also shown by a recent zero-value attack published at CHES 2024 [HMA+24]. To address this attack, Hirata *et al.* [HMA+24] proposed λ-detection M&M, adding intermediate cross-checks at critical points inside the AES S-box. However, as stated in [DOT25], λ-detection M&M is specifically designed against clock glitching, and does not protect against a gate/register-faulting adversary.

Dhooghe and Nikova [DN20b] proposed the first combined secure composable gadgets. However, their first proposal relies on intermediate abort mechanism which is deemed impractical to be implemented in hardware, and the second proposal was shown insecure by Richter-Brockmann *et al.* [RFSG22]. Then, Dhooghe and Nikova [DN20a] presented a gadget similar to CAPA, which again requires intermediate abort mechanisms. Later on, Feldtkeller *et al.* [FRSG22] proposed several combined secure gadgets based on the proposed notion of Combined-Isolating Non-Interference (CINI). These gadgets were later shown to be flawed by Feldtkeller *et al.* [FGM+23], who subsequently proposed the corrected versions. In a later work, Feldtkeller *et al.* [FRSG24] demonstrated how to adapt TI-like constructions to protect against combined attacks. All these countermeasures either rely on intermediate error correction, which becomes increasingly costly under multiple faults, or intermediate error detection, which requires intermediate abort mechanisms that are impractical to implement.

3 Stability Notion is Not Secure against Combined Attacks

In this section, we discuss the limitation of the original stability notion presented in [DOT24], which renders it vulnerable to combined attacks.

The stability notion ensures that any incorrect input codeword will always result in an incorrect output codeword. In the case of a register-to-register function with multiple input and output bits (*e.g.,* $\mathcal{Q}^4_{12}$), a single faulty input/output bit x (where $x \neq x'$) causes the entire input/output codeword to be incorrect. Hence, in the StaTI countermeasure, error propagation is ensured at the level of the entire codeword rather than on each share. While this approach ensures the fault propagation and the circuit to abort if at least one bit of the output codeword is faulty, it can introduce vulnerabilities in the presence of combined attacks, as demonstrated in SCA-FTA [SBJ$^+$21].

The stable encoding of the two-share $\mathcal{Q}^4_{12}$ from StaTI is presented in Section A. If $a^1_0(= a^0_0)$ is faulted to $a^1_0 + \Delta$, the fault propagates to the output, as $x^1_0 = a^1_0 + \Delta$ will always be faulty ($x^0_0 \neq x^1_0$ where $x^0_0 = a_0$) producing a faulty output codeword. However, if an error check layer is implemented after $\mathcal{Q}^4_{12}$, a probe at the output of the error check of y^0_0 and y^1_0 can reveal a secret data dependent fault propagation:

$$y^0_0 = b^0_0 + a^0_0 c^0 \,, \quad y^1_0 = b^1_0 + a^1_0 c^1 + \Delta c^1 \,,$$

where $y^0_0 \neq y^1_0$ if $c = 1$, making the fault propagation dependent of the S-box input c. Moreover, even if error propagation is ensured for all output shares, under certain composable combined security notions, an adversary may observe both duplicates of a value. While the fact that they differ from each other is independent of the secrets due to the stability, the observable difference may remain data-dependent and thus be exploitable.

Although each injected fault propagates to the output, triggering an abort signal without leaking any information about secret data (with no probing), it still reveals information about the specific share where the fault was injected. In the context of fault attacks, this does not leak any information about the secret data as it is equivalent to probing that specific share. However, in the context of combined attacks, based on the described combined security model in Section 2.1, a probe on the complimentary share (in a two-share implementation) of the faulted share leaks information about the unshared secret data. Consequently, the stable two-share AND gate proposed in StaTI does not provide combined security as both shares can be compromised, one through fault injection and the other through probing.

Furthermore, a reset fault to b^0_0 ($b^0_0 = 0$, $b^1_0 = \{0, 1\}$) renders the stable two-share AND gate not secure against combined attacks in the following scenario. Such a reset fault yields the following in Stage 1:

$$\begin{aligned} x^0_0 &= a^1_0 b^1_0 \,, & x^1_0 &= a^1_0 b^0_0 + b^1_0 \,, \\ x^0_1 &= a^1_0 b^0_1 \,, & x^1_1 &= a^1_0 b^1_1 \,, \end{aligned}$$

and in Stage 2:

$$z_0^0 = a_0^1 b_0^1 + a_0^1 b_1^0 \,, \qquad\qquad z_0^1 = a_0^1 b_0^1 + b_0^1 + a_0^1 b_1^1 \,.$$

As z_0^0 is simplified to $a_0 b$ (where b is correct) which is not masked with a random, thus, not probing secure even though the circuit aborts when the injected reset fault is effective (*i.e.,* $b_0^0 = 1$).

4 Stability Extended against Combined Attacks

In this section, we extend the stability notion introduced in StaTI to address and mitigate combined attacks. While most composable notions for combined security rely on error correction, our approach is based on error detection. As a result, not only the probes, but also the abort signal may leak information about the secret data, either by revealing the values of the variables targeted by the faults, or through fault propagation patterns that may depend on the secret data. Additionally, as in StaTI, we incorporate a single error check placed at the end of the entire circuit (*i.e.,* after encryption/decryption). This requires us to account for the faults that may propagate throughout the computation before being detected.

In this work, we focus on smaller gadgets, as their reduced complexity makes them easier to analyze and verify for security properties. However, the combined security definition given in Section 2.1 (Definition 1) is not composable and must be applied to the entire circuit. In this section, we work on deriving composable combined security properties such that, for when such gadgets are arbitrarily composed, the resulting circuit satisfies the combined security definition. We first quickly introduce the notion of simulatability, originally proposed in the context of probing security, which defines security through a simulation game. This notion is particularly useful for proving the security of composable gadgets.

Definition 3 (Simulatability [CS20b]). *Let $P = \{p_1, ..., p_\ell\}$ be a set of ℓ probes of a gadget C and C_P the tuple of values of the probes for an execution of C. Let $I = \{(i_1, j_1), ..., (i_k, j_k)\} \subset \{0, ..., d-1\} \times \{0, ..., m-1\}$ be a set of input wires of C. A simulator is a randomized function $\mathcal{S} : \mathbb{F}_q^k \to \mathbb{F}_q^\ell$. The set of probes P can be simulated with the set of input wires I if there exists a simulator $\mathcal{S}$ such that for any inputs $x_{*,*}$, the distributions $C_P(x_{*,*})$ and $\mathcal{S}(x_{i_1,j_1}, ..., x_{i_k,j_k})$ are equal, where the probability is over the random coins in C and $\mathcal{S}$.*

As described in Section 2.3, stability is arbitrarily composable and ensures data-independent fault propagation. To leverage these properties effectively in the context of combined security, we make use of arbitrarily composable glitch-extended probing security notions. However, we note that while the naive combination of stability and a glitch-extended probing secure notion does not immediately provide combined security, it still offers a stronger foundation than a simple combination of duplication and masking, due to its restrictions on probe and fault propagation. In the next section, we discuss how injected faults influence probing security and explore how to effectively integrate stability with glitch-extended probing secure notions to achieve security against combined attacks.

4.1 Composable Combined Security Properties

We now define security for gadgets that are subject to both probing and fault injection within the same execution. We first establish the correctness of the circuit under the single-probing and single-faulting combined adversary described in Section 2.1. Stability already ensures the correctness in this setting. It guarantees that any fault value present in the inputs propagates to the outputs, where it is eventually detected by the error detection mechanism placed at the end of the circuit. Any fault injected at the intermediate gates either affects a single output bit, resulting in an incorrect codeword when implemented considering fault non-completeness, or becomes ineffective producing the correct output. In either case, the output is either an incorrect codeword that triggers an abort signal, or a correct output, ensuring the correctness of the circuit.

Next, we establish the privacy of the circuit using simulation based arguments. To this end, we extend the definition of simulatability to account for both injected fault and the resulting abort signal, since our approach relies on error detection. In the simulation game, the adversary interacts either with the actual gadget or with a simulator that is given the injected fault and a subset of the inputs, depending on the specific security notion, but has no access to the secrets of the gadget (*i.e.,* unshared inputs). The simulator is required to simulate the probed variables and the abort signal. The adversary's goal is to distinguish whether they are interacting with the actual gadget or the simulator. If the adversary fails to distinguish between the two, it implies that they learn no more information than what is provided to the simulator.

While the original stability notion guarantees correctness at the codeword level, it falls short of preventing information leakage under combined attacks, as explained in Section 3. Specifically, although an incorrect input codeword results in an incorrect output codeword, an adversary can still exploit data dependent fault propagation at the share level by probing the error detection, or by observing the difference between the duplicates of a value. To address this, we refine the notion of stability into "diffused stability", which enforces fault propagation at the level of individual shares and ensures that the resulting error values between the duplicates remain independent of the inputs.

We first define the *fault propagation term*, which captures the difference between duplicated values in the presence of a fault.

Definition 4 (Fault Propagation Term). *Consider a duplicated register-to-register function computing F such that for any correct duplicated input (x^0, x^1), the output is (y^0, y^1) with $y^0 = y^1 = F(x^0) = F(x^1)$. Assume the input is faulty such that $x^0 \neq x^1$, resulting in an output (y^0, y^1) where $y^1 = y^0 + \Delta_{fp}$. We define Δ_{fp} as the* fault propagation term*, i.e., the difference between the two output duplicates caused by the faulty input.*

We now define diffused stability.

Definition 5 (Diffused Stability). *Consider a shared duplicated register-to-register function* $\tilde{F} : \mathbb{F}_2^{2n} \rightarrow \mathbb{F}_2^{2m}$*. We call* $\tilde{F}$ diffused stable*, if the fault propagation is ensured for each output share, and the fault propagation terms for each output share remain independent of the inputs of* $\tilde{F}$*, for all incorrect input codewords.*

This refined notion ensures that the fault propagation remains independent of the secret data, not only at the output level, but also at the level of individual shares. As a result, it mitigates potential leakages arising from data dependent fault propagation that could be exploited by FTA-SCA [SBJ+21]. However, despite its strengthening over previous notions, diffused stability alone is not sufficient to guarantee combined security, as we discuss below. At first, this property seems similar to the independence property that limits the fault propagation [AMR+20, SRM20, RSM21], but there are key differences. Diffused stability propagates input faults across the computation, impacting each output bit individually. In contrast, the independence property aims to constrain the effect of a fault to at most one output bit and does not guarantee that faults propagate. When enforced strictly, fault non-completeness reflects this property by limiting the spread of the faults. However, this restriction only applies to gate faults and does not protect against fault/combined attacks alone.

Before formally defining the notion of *combined stability* using diffused stability and arbitrarily composable glitch-extended probing security notions, we first discuss some key observations regarding the interaction between faults and probes. One notable observation, as noted by Clavier *et al.* [Cla07], is that fault injection can, in certain cases, serve as a probing tool. This can happen by faulting a variable to a fixed value and then observing whether the injected fault was effective based on the status of the abort signal. In the simulation game, this share is then given to the simulator, which implies that the adversary learns at most one additional share for each fault injection.

Fault injection can also compromise probing security when it targets the randomness used in the computations. It is clear that if the fault does not change the distribution of the random value, *e.g.,* a bitflip fault, it does not impact the probing security. However, faults that disturb the uniformity of the random value, such as by forcing the random to a fixed value or removing it entirely, undermine the effectiveness of the refreshing operation that relies on it [RFSG22]. In such scenarios, the fault essentially acts as a probe, leaking information about the involved random value. To account for this in the security model, the simulator is given access to the targeted random value, which may indirectly reveal additional shares if a value masked by the faulted randomness is also probed.

Additionally, fault injection can also cause a set of probes (or a single probe in an error detection or correction circuit) to observe fault propagation patterns that may depend on the secret data [SBJ+21]. Diffused stability addresses this by ensuring that the fault propagation remains independent of the secret data at the level of each output share (where an error detection circuit might be placed), thereby preventing information leakage through such observations. Moreover, it ensures that the difference between the duplicates of a value, although always non-zero due to stability, is independent of the inputs, eliminating another potential source of leakage under composable probing security notions, including scenarios where both duplicates may be observed.

Since we include a single error check mechanism placed at the end of the circuit, we must account for faults that may propagate across gadgets throughout the entire computation. This means that some gadgets may receive an incorrect input codeword, caused by a fault injected earlier in the circuit and propagated across gadgets as a result of stability. Diffused stability ensures that such faults, when appearing in the subsequent gadgets remain as data-independent additive faults. In such cases, we rely on the probing security guarantees provided by the underlying glitch-extended probing security notion. As observed in DOM-REP II [PBGS24], additive input faults do not compromise the probing security provided by the underlying glitch-extended probing security notion (which is strong non-interference in that case). This holds because, even in the presence of additive faults (which preserve the distribution, and hence do not leak the value of the faulted variable) and a probe, no additional input shares are given to the simulator beyond those already given, as long as the randomness is not faulted.[3] This behavior is also ensured in duplicated circuits by the diffused stability notion, as each fault is propagated, and the resulting fault propagation terms can simulated without any knowledge of the inputs. Consequently, the simulator is not given any additional information for an incorrect input codeword. At the output, even after the recombination of the (faulty) values, the resulting fault term remains masked by randomness (just like the recombined output value), thus, it remains independent of the secret data. However, when the circuit is duplicated, as required for error detection, care must be taken, since both duplicates of a value may become observable, even under a single-probe assumption (due to the error detection circuits, or composability). In such a case, fault-induced differences between the duplicates, although individually masked, can reveal some information regarding the inputs. This issue is addressed by the diffused stability notion. As a result, diffused stability reduces the combined security to the underlying probing security of the glitch-extended probing notion when an adversary probes given the input codeword is incorrect.

We first state the lemma regarding the probing security of the diffused stable gadgets *receiving incorrect input codeword*, which we already argued in above discussion. This lemma is particularly useful when a fault is injected earlier in the circuit, and a later gadget is probed. Due to the diffused stability, the injected fault propagates through the circuit, and manifests as an additive fault in the inputs of subsequent gadgets.

Lemma 1. *Let G be a diffused stable and arbitrarily composable glitch-extended probing secure gadget. If an adversary probes a value both with a correct and an incorrect input codeword (i.e., with an additive fault that is independent of the*

[3] It is important to note that we use fresh randomness, and therefore, we do not consider the randomness as part of the input codeword, as the propagated faults throughout the entire circuit do not affect the randomness used in the computation.

secret data), then no additional information is learned from the case with the incorrect input codeword beyond what is already learned from the correct one.

We now argue about the combined security of circuits which are arbitrarily composable second-order probing secure and diffused stable. We first define the notion of combined stability.

Definition 6 (Combined Stability (CS)). *An arbitrarily composable second-order probing secure masked gadget that is fault non-complete and diffused stable is called combined stable (CS).*

Theorem 1. *A CS gadget with an output decoding incorporating a share-wise error check is single-probe single-fault combined secure (Definition 1).*

Proof. We prove the correctness and the privacy of the combined stable gadget, where an output decoding gadget is called afterwards, against an adversary with single probing and single gate/register faulting capabilities. The output decoding gadget performs an error check on each output share, and gives back an abort signal if any of these error checks detect a fault.

First, we prove the correctness of the gadget, *i.e.,* the gadget returns an abort signal or the correct output. Probing does not affect the correctness of the gadget, hence, we focus on faults only. We distinguish between faults that change the value of the targeted variable (effective) and those that do not change the value (ineffective). If an injected fault is ineffective on the targeted value, then it is clear that the output does not change (*i.e.,* correct output). For an effective input fault making the input codeword incorrect, the correctness is ensured by the definition of diffused stability: any output share codeword is either incorrect causing an abort, or correct (when it is not driven by the faulty input) over all input values. Similarly, as shown in StaTI, an effective gate fault results in an incorrect codeword due to the fault non-completeness, which is then triggers an abort signal. Hence, the adversary cannot harm the correctness of the gadget.

Second, we prove the privacy of the gadget, *i.e.,* the abort status and the probed variable do not depend on the secrets. We know from the diffused stability that fault propagation is ensured for each output bit, and the difference between the duplicates (fault propagation term) is independent of the gadget inputs. We also know that the adversary learns at most one input share (of each input) with a probe as the gadget is arbitrarily composable second-order probing secure. From the discussed observations, a bitflip fault injected to a share of the input does not affect the probing security, meaning, the adversary still learns at most one input share (of each input), which preserves the privacy of the gadget. The fault also does not reveal any information about the share itself and always results in an abort signal. Then, simulating the abort signal is trivial in this case. A set/reset fault to an input share, however, reveals the value of the targeted share due to the abort signal. Meaning, the value of the targeted share is required to simulate the abort signal. Therefore, in addition to one share of each input required to simulate the probe, in total two shares are now required. As the gadget is glitch-extended second-order probing secure, meaning the gadget is implemented using at least three shares, the privacy of the gadget is not harmed. If the adversary injects a gate fault, then we can perceive this as an additional probe. Then, the adversary still learns at most two shares in total due to the combined stability. As the (perceived) probe reveals the value of the faulted variable, it is again trivial to simulate the abort signal as an effective fault will always result in an abort signal. □

The above theorem shows that in order to protect against single-probe single-fault combined adversary, it is sufficient to design a gadget that is arbitrarily composable second-order probing secure, fault non-complete and diffused stable, combined with an output decoding that performs a share-wise error detection.

Having established that the CS gadgets are secure against single-probe single-fault combined adversaries, the next step is to show that these gadgets are arbitrarily composable, that is, when such gadgets are composed, the resulting circuit preserves the combined stability.

Theorem 2. *A composition of two CS gadgets is again CS.*

Proof. Let G_3 be the composition of G_1 and G_2. We prove that G_3 is CS given that both G_1 and G_2 are CS. W.l.o.g. we assume G_1 and G_2 share at least one wire, and the output of G_2 is not an input of G_1. We then verify that G_3 is fault non-complete, diffused stable, and glitch-extended probing secure.

First of all, fault non-completeness is ensured by using separate gates for each output share. Since this separation in design is maintained under composition, the composed circuit is inherently fault non-complete.

In StaMAC [DOT25], stability of the arbitrary composition of stable gadgets was already proven. W.l.o.g., this extends to diffused stability. Since diffused stability extends stability by enforcing it individually for each output share, it preserves the same composability properties as the original stability notion.

Finally, G_3 preserves the arbitrarily composable second-order probing security. □

Considering Lemma 1 and Theorem 2, it follows that a circuit composed of CS gadgets is secure against single-probe single-fault combined adversary. The key insight provided by the Lemma 1 is that if a fault is injected early in the circuit, probing a different gadget later in the computation, whether it receives a correct or incorrect input codeword, does not leak additional information, since the fault propagates as a data-independent additive fault.

In the next section, we focus on how to construct such CS gadgets.

4.2 Transforming Probing Secure Circuits to Combined Stable Circuits

While linear operations are trivial to implement, securing multiplication against combined attacks is more complex. In this section, we show how to transform an arbitrarily composable glitch-extended probing secure gadget into a diffused stable counterpart, ensuring that it satisfies the CS definition (Definition 6). We demonstrate this transformation using the Strong Non-Interferent (SNI) notion defined by Barthe *et al.* [BBD+16], which is an arbitrarily composable glitch-extended probing secure notion. For completeness, we briefly recall the definition of the SNI notion.

The definition of simulatability (Definition 3) is extended in the context of SNI security by specifying which information is given to the simulator.

Definition 7 (d-Strong Non-Interferent (d-SNI) [BBD+16]). *A gadget G is d-SNI if any set of d_1 probes on its intermediate variables and every set of d_2 probes on its output shares such that $d_1 + d_2 \leq d$, the totality of the probes can be simulated with d_1 shares of each input.*

Following the definition of SNI, which ensures that a gadget remains secure against a glitch-extended probing adversary in a composable manner, StaMAC [DOT25] describes how to transform a glitch-extended probing secure circuit into a stable one. This is done by replacing each $\mathbb{F}_{2^n}^k \rightarrow \mathbb{F}_{2^n}^l$ register-to-register function in the original gadget with its stable counterpart. Since diffused stability strengthens this by requiring stability at the level of each individual output share, this transformation naturally extends to diffused stable gadgets. Moreover, using separate combinatorial gates for each output share simply ensures the fault non-completeness. It is proven in Theorem 7 of StaMAC [DOT25] that this transformation preserves the SNI properties of the original gadget. Additionally, as noted in StaMAC, this transformation can be applied to other composable notions such as Probe-Isolating Non-Interferene (PINI) by Cassiers and Standaert [CS20a].

In this context, due to the propagation of faults across multiple gadgets till the error detection mechanism, the simulator is given both faulty and correct values of an input share when the corresponding share is obtained through probing. This situation can be viewed as an implicit intermediate error checking. However, because diffused stability ensures the fault propagation remains data independent, observing both the correct and faulty values reveals no additional information to the adversary.

In the following section, we apply the transformation to the 2-SNI implementation of three-share AND gate [CRB+16] based on Consolidating Masking Schemes (CMS) [RBN+15], as depicted in Figure 1.

5 Combined Secure Multiplication Gadget

In this section, we present a shared and duplicated multiplication gadget over $\mathbb{F}_{2^n}$ that achieves security against single-probe single-fault combined adversaries, constructed using the notion of CS. Our design is based on the second-order probing secure masking of AND gate over three shares built on CMS, using 9 bits of randomness, as depicted in Figure 1. This gadget, when its outputs are registered, satisfies the SNI notion, ensuring glitch-extended second order probing security, similar to the DOM multiplier with registered output as shown by Faust *et al.* [FGP+18].

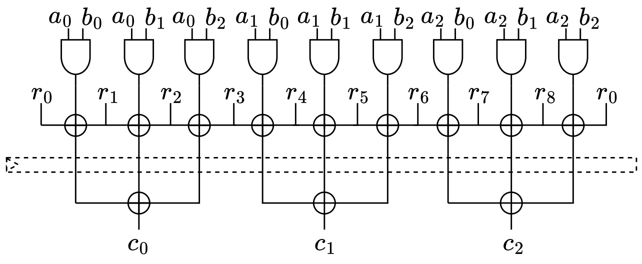
a_0 b_0 a_0 b_1 a_0 b_2 a_1 b_0 a_1 b_1 a_1 b_2 a_2 b_0 a_2 b_1 a_2 b_2
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_0
c_0 c_1 c_2

Fig. 1: Second-order three share AND gate from [CRB+16]. a_0, a_1, a_2 and b_0, b_1, b_2 are input shares, c_0, c_1, c_2 are output shares, and $r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8$ are the randoms used in the gadget.

We begin by describing the diffused stable (unshared) addition and multiplication gadgets over $\mathbb{F}_{2^n}$, which we then use as components to construct a CS multiplication gadget over $\mathbb{F}_{2^n}$. With these gadgets, we focus on ensuring the fault propagation with a data-independent Δ_{fp}.

Stable addition in $\mathbb{F}_{2^n}$. We describe the stable gadget G_+ that implements the addition in $\mathbb{F}_{2^n}$, as detailed in Algorithm 1, where $x^0 = x^1$ and $y^0 = y^1$ are the duplicated inputs. $\vee$ denotes the logical OR operation which evaluates to $1 \in \mathbb{F}_{2^n}$ if at least one of the operands is $1 \in \mathbb{F}_{2^n}$, and evaluates to $0 \in \mathbb{F}_{2^n}$ otherwise. The operation $\neq$ evaluates to $0 \in \mathbb{F}_{2^n}$ only if both operands are the same and to $1 \in \mathbb{F}_{2^n}$ otherwise.

Algorithm 1: G_+ : A stable addition gadget operating in $\mathbb{F}_{2^n}$

```
**Input:** $x^0, x^1, y^0, y^1$
**Output:** $z^0, z^1$
**if** $((x^0 + x^1) \neq 0) \vee ((y^0 + y^1) \neq 0)$ **then**
    $z^0 = 1$
    $z^1 = 0$
**else**
    $z^0 = x^0 + y^0$
    $z^1 = x^1 + y^1$
**end**
**return** $z^0, z^1$
```

G_+ is correct when the input codeword is correct (*i.e.*, $(x^0 + x^1)$ and $(y^0 + y^1)$ are both zero) since it returns $z^0 = x^0 + y^0$ and $z^1 = x^1 + y^1 = z^0$. In the following, we prove that G_+ is diffused stable.

Theorem 3. *The gadget G_+ as defined in Algorithm 1 is diffused stable.*

Proof. In order to prove the diffused stability of G_+, we show that any incorrect input codeword is mapped to an incorrect output codeword, and the fault propagation term is independent of the inputs. Let $\Delta_{x^0}, \Delta_{x^1}, \Delta_{y^0}, \Delta_{y^1}$ be the additive fault values present in x^0, x^1, y^0, y^1, respectively. In that case, the input codeword is incorrect if at least one of the following inequalities holds

$$E_x = x^0 + \Delta_{x^0} + x^1 + \Delta_{x^1} \neq 0$$
$$E_y = y^0 + \Delta_{y^0} + y^1 + \Delta_{y^1} \neq 0.$$

Then, given that the input codeword is incorrect, we show that the output codeword is also incorrect.

To check whether the output codeword is correct, we check whether $z^{0\prime} + z^{1\prime}$ equals zero, with $(z^{0\prime}, z^{1\prime})$ (faulted) outputs of G_+. We can write this out as follows.

$$z^{0\prime} + z^{1\prime} = (E_x \vee E_y) + (x^0 + \Delta_{x^0} + y^0 + \Delta_{y^0} + x^1 + \Delta_{x^1} + y^1 + \Delta_{y^1})\neg(E_x \vee E_y)$$

$z^{0\prime} + z^{1\prime}$ is zero only when both $E_x = 0$ and $E_y = 0$, which implies the input codeword is correct. When the input codeword is incorrect, $\Delta_{fp} = z^{0\prime} + z^{1\prime} = 1$ and thus independent of the inputs. That is, G_+ maps any incorrect input codeword to an incorrect output codeword with a data-independent Δ_{fp} and is thus diffused stable.

□

We also verified the diffused stability of the addition gadget over $\mathbb{F}_{2^4}$ using software by exhaustively evaluating all incorrect input codewords. The choice of $\mathbb{F}_{2^4}$ was made to keep the verification feasible, as an exhaustive evaluation over $\mathbb{F}_{2^8}$ would require significantly more computational resources.

Stable multiplication in $\mathbb{F}_{2^n}$*.* We describe the diffused stable gadget G_* that implements the multiplication in $\mathbb{F}_{2^n}$, as detailed in Algorithm 2, where $x^0 = x^1$ and $y^0 = y^1$ are the duplicated inputs.

Algorithm 2: G_* : A stable multiplication gadget operating in $\mathbb{F}_{2^n}$

```
Input: $x^0, x^1, y^0, y^1$
Output: $z^0, z^1$
if $((x^0 + x^1) \neq 0) \vee ((y^0 + y^1) \neq 0)$ then
    $z^0 = 1$
    $z^1 = 0$
else
    $z^0 = x^0 y^0$
    $z^1 = x^1 y^1$
end
return $z^0, z^1$
```

G_* is correct when the input codeword is correct (*i.e.*,$(x^0 + x^1)$ and $(y^0 + y^1)$ are both zero) since it returns $z^0 = x^0 y^0$ and $z^1 = x^1 y^1 = z^0$. In the following, we prove that G_* is diffused stable.

Theorem 4. *The gadget G_* as defined in Algorithm 2 is diffused stable.*

Proof. Similar to the diffused stable addition gadget, to prove the diffused stability of G_*, we show that any incorrect input codeword is mapped to an incorrect output codeword with a data-independent Δ_{fp}. Let $\Delta_{x^0}, \Delta_{x^1}, \Delta_{y^0}, \Delta_{y^1}$ be the additive fault values present in x^0, x^1, y^0, y^1, respectively. In that case, the input codeword is incorrect if at least one of the following inequalities holds

$$E_x = x^0 + \Delta_{x^0} + x^1 + \Delta_{x^1} \neq 0$$
$$E_y = y^0 + \Delta_{y^0} + y^1 + \Delta_{y^1} \neq 0.$$

Given that the input codeword is incorrect, we show that the output codeword is also incorrect. To check whether the output codeword is correct, we check whether ${z^0}' + {z^1}'$ equals zero, with $({z^0}', {z^1}')$ (faulted) outputs of G_*.

When we error check $({z^0}' + {z^1}' == 0)$, we obtain the following:

$${z^0}' + {z^1}' = (E_x \vee E_y) + ((x^0 + \Delta_{x^0})(y^0 + \Delta_{y^0}) + (x^1 + \Delta_{x^1})(y^1 + \Delta_{y^1}))\neg(E_x \vee E_y)$$

It is clear that ${z^0}' + {z^1}' = 0$ only when both $E_x = 0$ and $E_y = 0$. When the input codeword is incorrect, then $\Delta_{fp} = {z^0}' + {z^1}' = 1$ and input independent. That means, G_* is diffused stable.

□

Similar to the diffused stable addition gadget, we also verified the diffused stability of the multiplication gadget over $\mathbb{F}_{2^4}$ using software by exhaustively evaluating all incorrect input codewords.

In the next section, we present the construction of the CS multiplication gadget, building on the diffused stable addition and multiplication gadgets introduced in this section.

5.1 Combined Stable Multiplication Gadget

In this section, we describe the CS multiplication gadget in Algorithm 3, obtained by transforming the second-order masking of AND gate over three shares based on CMS with registered outputs into its diffused stable variant.

According to Theorem 7 in StaMAC [DOT25], each addition and multiplication gate in the probing secure multiplier can be replaced with its stable counterpart to achieve (diffused) stability. While it is not necessary to replace every addition and multiplication to achieve stability, it is important to consider that the adversary is capable of probing intermediate values, not just the final result. That is, we must ensure that any probed intermediate value (also gadget outputs) does not reveal more information than what is allowed under the SNI property. Algorithm 3 describes the CS multiplication gadget, where G_* is the unshared diffused stable multiplication gadget described in Algorithm 2, and G_+ is the three input version of the unshared stable addition gadget described in (Algorithm 1). The extension of the unshared stable addition to three inputs is trivial:

the if condition is extended to $((a^0+a^1)\neq 0)\vee((b^0+b^1)\neq 0)\vee((c^0+c^1)\neq 0)$. For clarity, we depict the algorithm for computing a single output share (z_0^0, z_0^1) in Figure 2.

Algorithm 3: CS multiplication gadget

```
Input: Independent duplicated shares of $x = (x_0^0, x_0^1, x_1^0, x_1^1, x_2^0, x_2^1)$ and
       $y = (y_0^0, y_0^1, y_1^0, y_1^1, y_2^0, y_2^1)$, and uniform randoms $r_i$ for $0 \le i \le 8$
Output: $z = xy = (z_0^0, z_0^1, z_1^0, z_1^1, z_2^0, z_2^1)$
for $i \leftarrow 0$ to 2 do
    $u_{i,i}^0, u_{i,i}^1 = G_*(x_i^0, x_i^1, y_i^0, y_i^1) + r_{3\cdot i} + r_{3\cdot i+1}$
    for $j \leftarrow i+1$ to 2 do
        $u_{i,j}^0, u_{i,j}^1 = G_*(x_i^0, x_i^1, y_j^0, y_j^1) + r_{3\cdot i+j \pmod 3} + r_{3\cdot i+j+1 \pmod 3}$
        $u_{j,i}^0, u_{j,i}^1 = G_*(x_j^0, x_j^1, y_i^0, y_i^1) + r_{3\cdot j+i \pmod 3} + r_{3\cdot j+i+1 \pmod 3}$
    end
end
for $i \leftarrow 0$ to 2 do
    $z_i^0, z_i^1 = G_+(u_{i,i}, u_{i,i+1 \pmod 3}, u_{i,i+2 \pmod 3})$
end
```

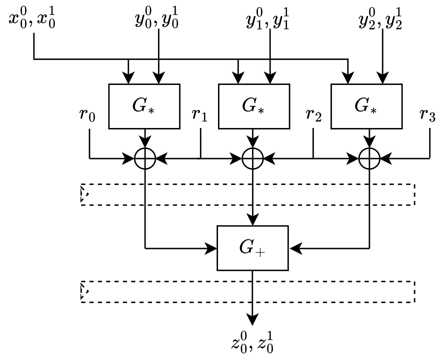

Fig. 2: The computation of the output share (z_0^0, z_0^1) in the CS multiplication gadget. G_+ and G_* are the stable addition (Algorithm 1) and multiplication (Algorithm 2) gadgets, respectively. The XOR gates refer to the duplicated XOR (the same random is added to both duplicated values).

We first prove the correctness of Algorithm 3. Given correct (duplicated) input sharing of x and y, we show that the output shares z_i are duplicated Boolean shares of xy (*i.e.*, $\sum_{i=0}^{2} z_i^0 = \sum_{i=0}^{2} z_i^1 = xy$). If the input sharing is correct, then we know that $G_*(a,b)$ returns ab, and $G_+(a,b,c)$ returns $a+b+c$, both in duplicated form. Then, the computations for both redundancy domains are being identical, we have

$$
\begin{aligned}
\sum_{i=0}^{2} z_i = \sum_{i=0}^{2} \left(\sum_{j=0}^{2} u_{i,j} \right) \\
&= \sum_{i=0}^{2} \Bigg(x_i y_i + r_{3\cdot i} + r_{3\cdot i+1} + \sum_{j<i} (u_{i,j} + r_{3\cdot j+i \pmod 3} + r_{3\cdot j+i+1 \pmod 3}) \\
&\quad + \sum_{i<j} (u_{i,j} + r_{3\cdot i+j \pmod 3} + r_{3\cdot i+j+1 \pmod 3}) \Bigg) \\
&= \sum_{i=0}^{2} \left(x_i y_i + \sum_{j<i} x_i y_j + \sum_{i<j} x_i y_j \right) \\
&= \sum_{i=0}^{2} x_i \sum_{j=0}^{2} y_j = y \sum_{i=0}^{2} x_i = xy \,.
\end{aligned}
$$

We now prove that the presented multiplication gadget is combined stable.

Theorem 5. *The presented multiplication gadget in Algorithm 3 is CS.*

Proof. Combined stability requires circuits to be fault non-complete, diffused stable, and arbitrarily composable glitch-extended second-order probing secure.

First, fault non-completeness is related to the implementation, and is achieved by using different gates for each output share.

Second, we know that the three-share CMS multiplier with registered outputs is known to be 2-SNI, which means it is arbitraily composable second-order glitch-extended probing secure. According to Theorem 7 in StaMAC, transforming an arbitraily composable glitch extended probing secure gadget into its diffused stable counterpart preserves the probing security. Additionally, 2-SNI security of the combined stable gadget has been formally verified using the verification tool VERICA [RFSG22].

Next, we show the diffused stability of the gadget. We first show that any incorrect input codeword generates an incorrect codeword at the state of the middle register layer of the multiplier. Similar to the discussion in Section 4.1, we exclude the random values, as they do not affect the correctness, but only probing security. Moreover, incorrect input codeword happens when an earlier gadget produces incorrect output codeword, and since we use fresh randomness, the random values can be faulty only if they are targeted by the fault injection. We assume at least one of the values $(x_0^0, {x_0}^1)$, $(x_1^0, {x_1}^1)$, $(x_2^0, {x_2}^1)$, $(y_0^0, {y_0}^1)$, $(y_1^0, {y_1}^1)$, $(y_2^0, {y_2}^1)$ is an incorrect codeword. As each $u_{i,j}^0, u_{i,j}^1$ couple is computed using a diffused stable multiplication gadget, an incorrect input codeword is mapped to an incorrect state codeword in the middle register layer for each $u_{i,j}$. Additionally, the differences between $u_{i,j}^0$ and $u_{i,j}^1$ are independent of the corresponding stable multiplication inputs. For the second phase of the multiplication gadget, the diffused stable additions deriving the outputs of the stable multiplications ensure that the fault present in the input is propagated to the output and it is independent of the secrets while it is ensured for each output share. Thus, the gadget is CS.

□

Lemma 2. *The CS multiplication gadget described in Algorithm 3 satisfies 1-SNI security when the input codeword is incorrect.*

Proof. We assume each input x_i^l, y_i^l has a fault value $\Delta_{x_i{}^l}, \Delta_{y_i{}^l}$. We construct a simulator for the probed intermediate value that makes use of at most one share, or output value without any use of input shares. We first classify the internal variables in the following groups.

1- $x_i^l + \Delta_{x_i{}^l}, y_i^l + \Delta_{y_i{}^l}$
2- $u_{i,j}^l$
3- z_i^l

We define two sets of indices I and J such that $|I| \leq 1$, $|J| \leq 1$, and the probed value and its duplicate can be perfectly simulated using $\{x_i^0, x_i^1\}_{i \in I}$ and $\{y_j^0, y_j^1\}_{j \in J}$. Then, we construct the sets as follows.

– If the probe is in (1) or (3), add i to I and J.
– If the probe is in (2), add i to I and j to J.

Then we have $|I| = 1$, $|J| = 1$. We now show how the simulator behaves for the observed values.

– For a probe in group (1), the simulator has access to $x_i^l + \Delta_{x_i{}^l}, y_i^l + \Delta_{y_i{}^l}$ values. Thus, these values can be perfectly simulated.
– For a probe in group (2), the simulator has access to $x_i^l + \Delta_{x_i{}^l}$ and $y_j^l + \Delta_{y_j{}^l}$ values. Additionally, any random value can be simulated as uniform random variable. Thus, $u_{i,j}^l$ values can be perfectly simulated.
– For a probe in group (3), the simulator has access to $x_i^l + \Delta_{x_i{}^l}$ and $y_i^l + \Delta_{y_i{}^l}$ values. Then we assign random values to the terms $u_{i,j}^0$, and $u_{i,j}^1 = u_{i,j}^0 + 1$ for $j = \{0, 1, 2\}$ (depending on the input fault) due to the unshared diffused stable multiplication gadget.

We now simulate the outputs z_i^l using no information from the inputs. Each output has a random value, which is not probed before. Plus, due to the unshared diffused stable addition and multiplication gates, simulator can assign z_i^0 a random, and $z_i^1 = z_i^0 + 1$, which are also perfectly simulated, completing the proof. □

We also provide a simulation-based proof that, for a given incorrect codeword (or a single data-dependent input fault), the CS multiplication gadget retains 1-SNI security in the glitch-extended probing model, meaning that the probe can be simulated using at most one share.

Lemma 2 provides a necessary step to establish that the CS multiplication gadget described in Algorithm 3 is combined secure.

Theorem 6. *The CS multiplication gadget described in Algorithm 3 is single-probe single-fault combined secure (Definition 1).*

Proof. Due to the diffused stability, simulating the abort signal is trivial and requires at most one input share. Thus, whether the input codeword incorrect due to an earlier fault or not, the probe and the abort signal can be simulated using at most two shares, meaning the three share CS multiplication gadget is combined secure. □

We verified that the CS multiplication gadget with an additional input Δ, subsequently XORed to one multiplication input (*e.g.*, x_0^0) is 2-SNI using the verification tool VERICA [RFSG22]. While this does not prove the combined stability of the gadget, the output probes are indeed simulated using no input values, and that probing two duplicates at the same time does not leak any more input shares.

We finally note that it is also possible to obtain combine stable multipliers by transforming HPC1 [CGLS21], HPC2 [CGLS21], or HPC3 [KM22] PINI-secure multiplication gadgets, as well as all SNI-secured multiplication gadgets.

6 Combined Stable AES S-box Implementation

In this section, we describe the hardware implementation of AES S-box based on the CS multiplication gadget, as detailed in Section 5.1. We first discuss its side-channel, fault, and combined security. Then, we present the hardware cost of our implementation in comparison to state-of-the-art combined attack countermeasures.

6.1 Combined Stable AES S-box

The AES S-box consists of an inversion in $\mathbb{F}_{2^8}$ (mapping zero to zero) followed by an affine transformation. As the affine transformation is a linear operation and typically straightforward to implement securely, our focus is on protecting the inversion.

Our implementation is based on the Canright's S-box [Can05], which uses a tower-field decomposition with six pipeline stages. The inverter in the Canright's S-box is composed of linear maps performing basis changes, multiplications in $\mathbb{F}_{2^2}$ and $\mathbb{F}_{2^4}$, linear scaling functions, and a linear two-bit inverter. The construction of this inverter is depicted in Figure 3.

To secure the AES S-box against a single-probe single-fault combined adversary, we first duplicate the circuit, and replace all finite field multiplications with their CS counterparts, using the CS multiplication gadget described in Algorithm 3. The resulting circuit design is shown in Figure 3. It is important to note that replacing the XOR gates with their stable variants is not necessary, as the S-box input is also the input of the multiplications in Stage 5, ensuring that any fault present in the S-box input is always propagated to the S-box output. We also note that, specifically for the first-order combined security, we use an

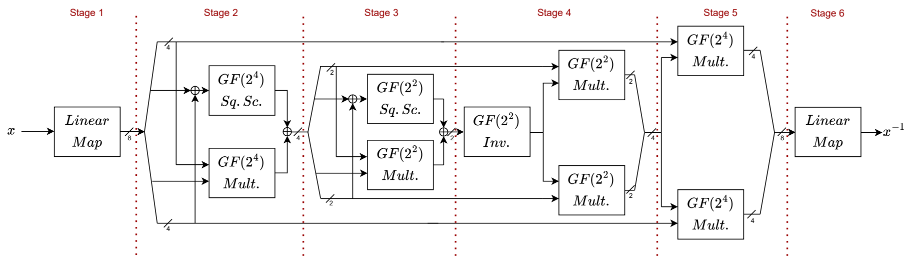

Fig. 3: Canright implementation of the $\mathbb{F}_{2^8}$ inverter in the AES S-box.

unstable addition in the CS multiplication gadget. As any fault affects either one or three shares, the design still guarantees fault propagation despite the unstable addition gadget.

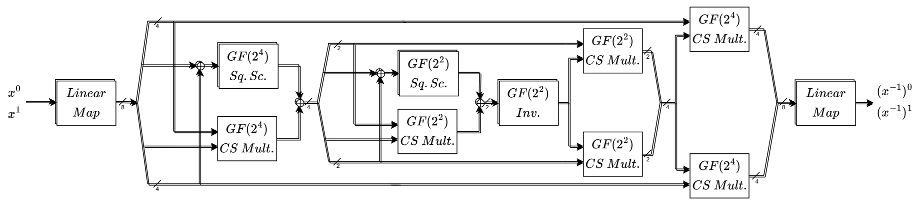

Fig. 4: AES S-box protected against single-probe single-fault combined adversary. *CS Mult.* gadgets refer to the CS multiplication gadget described in Algorithm 3.

Protecting the AES S-box does not increase the latency, *i.e.,* number of clock cycles, compared to the SCA-only S-box implementation. Similarly, we use the same amount of randomness, 162 bits, as the SCA-only design.

6.2 Side-Channel Security

In this section, we discuss the SCA security of the combined stable AES S-box implementation. To evaluate the SCA security of our design, we perform TVLA (Test Vector Leakage Assessment) [CDG+13]. We perform the assessment for the combined stable S-box by supplying it with fixed versus random inputs. The S-box is placed on a Xilinx Spartan-6 FPGA located on a SAKURA-G evaluation board [SAK], whereas mask generation is located on a separate control FPGA. The devices are supplied with a stable 6.144 MHz clock and an oscilloscope samples power consumption traces at a rate of 1 GS/sec. The results of the first, second, and third-order tests are depicted in Figure 5. The results report no leakage regarding first and second-order probing security. Since the design is second-order secure against SCA, we observe leakage starting from the third-order test.

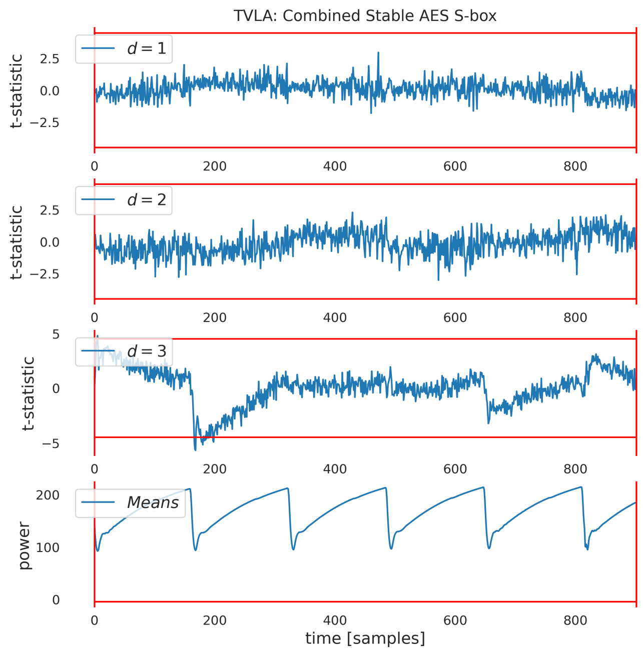

Fig. 5: Combined stable S-box TVLA result, first, second and third-order security, 100 million traces, and a sample trace.

6.3 Fault and Combined Security

To evaluate the practical fault and combined security of countermeasures, no standardized fault and combined evaluation methodologies currently exist, comparable to TVLA used to evaluate SCA security. Hence, we currently lack the means to evaluate the fault and combined security of our design practically.

Furthermore, existing fault and combined verification tools cannot be applied to our design, as they are built to verify security notions different from stability, and CS. In particular, in the proposed security notion CS, duplicated computation domains intentionally interact with each other, which results in diffusion of faults across these domains. Current verification tools and security notions are based on isolated computation domains, and limited fault diffusion, and hence, do not account the fault propagation patterns introduced by the notion of CS. As a result, these tools are unable to accurately verify the security properties provided by our design. Since the notion of CS cannot be captured by these tools, we rely on formal security proofs to establish the security of our design against single-probe single-fault combined adversaries. Consequently, our evaluation is currently limited to theoretical security proofs. These proofs demonstrate that our design satisfies the security requirements of the combined security.

To assess the diffused stability of our CS AES S-box implementation, we first verified the diffused stability of its core building gadgets (*i.e.,* unshared addition and multiplication) via a simple software tool. This tool exhaustively checks each incorrect faulty input codeword and verifies that the output codeword is also incorrect for each output share. Then, due to the composability of the diffused stability notion, the entire AES S-box is diffused stable. Following that, we used VERICA to verify that the CS multiplication gadget satisfies 2-SNI. Since this gadget is both fault non-complete and diffused stable, it fulfills the requirements of our combined security notion.

In addition to the diffused stability verification, we performed an attack simulation on the C implementation of the CS AES S-box. Faults were modeled as XOR additions applied to variables. The simulation involved extending the C implementation to inject faults by XORing a fault variable to selected critical variables (*e.g.,* inputs of the multiplications) aligning with the gate/register faulting adversary. The results verified that all injected faults were propagated to the S-box output yielding an incorrect output codeword for each output share regardless of the S-box input.

6.4 Hardware Benchmarks

In this section, we evaluate the hardware benchmarks of our combined stable AES S-box implementation compared with the AES S-box implementations protected with λ-detection M&M [HMA+24], Combined Private Circuits ($\widehat{\text{CPC}}_1$) [FGM+23], and Combined Threshold Implementation (CCMS, CTI, CNFR) [FRSG24] countermeasures. We use the Synopsys Design Compiler (version S-2021.06-SP3) together with the open source NANGATE 45nm library. The hierarchy is preserved during evaluation by enabling the *set_dont_touch* constraint and the compile option *exact_map* is used to avoid optimizations.

Table 1 compares the performance characteristics of AES S-box implementations protected with the listed countermeasures in terms of latency, randomness and area costs, and overhead compared to SCA-only implementations. For this work, the overhead is calculated based on our own implementation of the SCA-only design by De Cnudde *et al.* [CRB+16] which is based on three-share CMS. Numbers for λ-detection M&M [HMA+24] are directly taken from the respective publication. For Combined Private Circuits ($\widehat{\text{CPC}}_1$), the reported numbers are taken from [FRSG24]. Numbers for CCMS, CTI and CNFR and their corresponding SCA-only designs were obtained by resynthesizing the publicly available reference implementations provided by the authors.

Among the countermeasures listed in Table 1, only λ-detection M&M [HMA+24] and this work employ error detection based on two replications, whereas the other designs rely on error correction mechanisms using three replications. Similarly, both λ-detection M&M [HMA+24] and our design provide second-order SCA security, while the remaining designs achieve only first-order SCA security. A key

Table 1: Hardware cost comparison of combined attack countermeasures implementing an AES S-box. s refers to number of shares and n refers to the number of replications. Overhead factor is relative to SCA-only implementations.

Design	s	n	Latency (cycles)	Rand.	Area (kGE)	Overhead (factor)
CMS [CRB+16]	2	1	5	54	1.83	-
CMS [CRB+16]	3	1	5	162	5.18	-
TI [GC17]	4	1	3	0	3.55	-
NFR [SM21]§	2	1	5	1	1.91	-
λ-detection M&M [HMA+24]†	3	2*	6	903	29.3	3.49
$\widehat{\text{CPC}}_1$ [FGM+23]‡	2	3	6	144	10.8	4.01
CCMS [FRSG24]	2	3	5	62	6.51	3.55
CTI [FRSG24]	4	3	3	0	11.45	3.22
CNFR [FRSG24]§	2	3	5	2	7.01	3.67
This work	3	2	5	162	11.57	2.23

§ Only the $\mathbb{F}_{2^8}$ inversion, * Duplication through MAC tags,
† Reported numbers by the authors, ‡ Numbers are taken from [FRSG24]

distinction of this work is that it does not rely on intermediate error detection or correction, relying instead on a single error detection mechanism at the end of the circuit.

Due to the compact structure of CMS and NFR AES S-box designs [CRB+16] with two shares, the CCMS and CNFR [FGM+23] designs achieve the lowest absolute area among the combined countermeasures listed in Table 1, with areas of 6.51kGE and 7.01kGE, respectively. However, these designs rely on three replications and intermediate error correction, resulting in overhead factors of 3.55 and 3.67, respectively, compared to their SCA-only counterparts.

In contrast, our design employs two replications and achieves combined security with a lower relative overhead factor of 2.23, despite having higher absolute area (11.57kGE). This lower overhead is due to the absence of intermediate error checks and the use of diffused stable operations to enable a single error detection with two replications. Although the area is higher in absolute terms, the relative overhead remains competitive given the stronger security guarantees.

Finally, it is worth noting that our implementation is based on the second-order CMS multiplication gadget, which requires 162 bits of randomness. Replacing this gadget with an alternative SNI-secure multiplication gadget requiring less randomness could potentially reduce the area further.

7 Conclusions and Future Work

In this work, we addressed the challenge of protecting cryptographic implementations against combined attacks without relying on intermediate error checks and corrections. Building on the concept of stability introduced in StaTI, we proposed the notion of “combined stability”, a novel composable security notion that extends glitch-extended probing security notions with “diffused stability”. This notion enables the secure composition of small gadgets into larger circuits, requiring only a single error detection mechanism placed at the end of the computation. Compared to traditional approaches that often involve multiple layers of error correction, this represents a significant improvement in the design of countermeasures against combined attacks.

We proposed a generic transformation for building composable combined secure gadgets, and demonstrated its practicality on a second-order CMS-based AND gate. This gadget was then used to implement a combined secure AES S-box. The resulting design achieves second-order SCA security, first-order fault security, and single-probe single-fault combined security, all with competitive implementation cost.

To the best of our knowledge, this is the first work to demonstrate that combined security can be achieved with a single error detection mechanism placed at the end of the circuit. This pioneering use of a single detection layer opens new directions for scalable and practical countermeasure design in the context of combined attacks. Our approach focuses on robust first-order combined security as a solid foundation, leaving higher-order protection as a promising direction for future research. While higher-order SCA protection can be achieved by increasing the number of shares, supporting multiple faults is more challenging due to the significantly larger attack surface. In such cases, the stability may be broken, for instance, if a second fault targets the equality checks in the CS gadgets. This may require reintroducing intermediate error checks or replicating equality checks with increased redundancy/number of shares. Future work could also explore alternative CS gadget designs that resist multiple faults.

Although some countermeasures are theoretically scalable to higher-order combined security, we note that they are rarely implemented beyond first-order security in practice. This is largely due to the complexity and inefficiency of higher-order protections, for instance, reliable majority voting becomes quickly impractical.

Another interesting future work direction is to extend the notion of stability to support error correction. This could enable different trade-offs. For instance, replacing a design with three share, two replications, and a single error detection, with one uses two shares, three replications, and a single error correction. Such an approach could benefit from the compact area of two-share SCA secure designs, though the overhead (and practicality) of stable gadgets with three replications remains to be evaluated. Finally, integrating the notion of combined stability into formal verification tools would enable the systematic evaluation of hardware designs with respect to this security notion.

Acknowledgements. This work was supported by CyberSecurity Research Flanders with reference number VR20192203. This work has been partially supported by the National Science Fund of Bulgaria under Grant KP-06-N82/5.

A Stable encoding of $\mathcal{Q}_{12}^4$

Stage 1	Stage 2
$k_0^0 = a_0^0$	$x_0^0 = k_0^0$
$k_1^0 = a_1^0$	$x_1^0 = k_1^0$
$l_0^0 = b_0^0 + a_0^0 c_0^0$	
$l_1^0 = a_0^0 c_1^0$	$y_0^0 = l_0^0 + l_1^0 + (l_0^0 + l_0^1)(l_1^0 + l_1^1)$
$l_2^0 = a_1^0 c_1^0$	$y_1^0 = l_2^0 + l_3^0 + (l_2^0 + l_2^1)(l_3^0 + l_3^1)$
$l_3^0 = b_1^0 + a_1^0 c_0^0$	
$m_0^0 = c_0^0 + a_0^0 b_0^0 + a_0^0 c_0^0$	
$m_1^0 = a_0^0 b_1^0 + a_0^0 c_1^0$	$z_0 = m_0^0 + m_1^0 + (m_0^0 + m_0^1)(m_1^0 + m_1^1)$
$m_2^0 = a_1^0 b_0^0 + a_1^0 c_0^0$	$z_1^0 = m_2^0 + m_3^0 + (m_2^0 + m_2^1)(m_3^0 + m_3^1)$
$m_3^0 = c_1^0 + a_1^0 b_1^0 + a_1^0 c_1^0$	
$n_0^0 = d_0^0$	$w_0^0 = n_0^0$
$n_1^0 = d_1^0$	$w_1^0 = n_1^0$
$k_0^1 = a_0^1$	$x_0^1 = k_0^1$
$k_1^1 = a_1^1$	$x_1^1 = k_1^1$
$l_0^1 = b_0^1 + a_0^1 c_0^1$	
$l_1^1 = a_0^1 c_1^1$	$y_0^1 = l_0^1 + l_1^1$
$l_2^1 = a_1^1 c_1^1$	$y_1^1 = l_2^1 + l_3^1$
$l_3^1 = b_1^1 + a_1^1 c_0^1$	
$m_0^1 = c_0^1 + a_0^1 b_0^1 + a_0^1 c_0^1$	
$m_1^1 = a_0^1 b_1^1 + a_0^1 c_1^1$	$z_0^1 = m_0^1 + m_1^1$
$m_2^1 = a_1^1 b_0^1 + a_1^1 c_0^1$	$z_1^1 = m_2^1 + m_3^1$
$m_3^1 = c_1^1 + a_1^1 b_1^1 + a_1^1 c_1^1$	
$n_0^1 = d_0^1$	$w_0^1 = n_0^1$
$n_1^1 = d_1^1$	$w_0^1 = n_1^1$

References

AK97. Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1997.

AMR$^+$20. Anita Aghaie, Amir Moradi, Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, Falk Schellenberg, and Tobias Schneider. Impeccable circuits. *IEEE Trans. Computers*, 69(3):361–376, 2020.

BBD$^+$16. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 116–129. ACM, 2016.

BDL97. Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

Can05. David Canright. A very compact s-box for AES. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.

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

CGLS21. Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

CJRR99. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

Cla07. Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2007.

CRB$^+$16. Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 shares in hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016.

CS20a. Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

CS20b. Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

DDRT12. Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic transient faults injection on a hardware and a software implementations of AES. In Guido Bertoni and Benedikt Gierlichs, editors, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*, pages 7–15. IEEE Computer Society, 2012.

DN20a. Siemen Dhooghe and Svetla Nikova. Let's tessellate: Tiling for security against advanced probe and fault adversaries. In Pierre-Yvan Liardet and Nele Mentens, editors, *Smart Card Research and Advanced Applications - 19th International Conference, CARDIS 2020, Virtual Event, November*

18-19, 2020, Revised Selected Papers, volume 12609 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2020.

DN20b. Siemen Dhooghe and Svetla Nikova. My gadget just cares for me - how NINA can prove security against combined attacks. In Stanislaw Jarecki, editor, *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, volume 12006 of *Lecture Notes in Computer Science*, pages 35–55. Springer, 2020.

DOT24. Siemen Dhooghe, Artemii Ovchinnikov, and Dilara Toprakhisar. Stati: Protecting against fault attacks using stable threshold implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(1):229–263, 2024.

DOT25. Siemen Dhooghe, Artemii Ovchinnikov, and Dilara Toprakhisar. Stamac: Fault protection via stable-mac tags. *IACR Cryptol. ePrint Arch.*, page 455, 2025.

DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

FGM$^+$23. Jakob Feldtkeller, Tim Güneysu, Thorben Moos, Jan Richter-Brockmann, Sayandeep Saha, Pascal Sasdrich, and François-Xavier Standaert. Combined private circuits - combined security refurbished. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 990–1004. ACM, 2023.

FGP$^+$18. Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.

FRSG22. Jakob Feldtkeller, Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. CINI MINIS: domain isolation for fault and combined security. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1023–1036. ACM, 2022.

FRSG24. Jakob Feldtkeller, Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. Combined threshold implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(4):307–334, 2024.

GC17. Ashrujit Ghoshal and Thomas De Cnudde. Several masked implementations of the boyar-peralta AES s-box. In Arpita Patra and Nigel P. Smart, editors, *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings*, volume 10698 of *Lecture Notes in Computer Science*, pages 384–402. Springer, 2017.

GMK16. Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Pro-*

ceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016, page 3. ACM, 2016.

GMO01. Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

Hab65. D. H. Habing. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. *IEEE Transactions on Nuclear Science*, 12(5):91–100, 1965.

HMA$^+$24. Haruka Hirata, Daiki Miyahara, Victor Arribas, Yang Li, Noriyuki Miura, Svetla Nikova, and Kazuo Sakiyama. All you need is fault: Zero-value attacks on AES and a new λ-detection m&m. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(1):133–156, 2024.

ISW03. Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

KJJ99. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

KM22. David Knichel and Amir Moradi. Low-latency hardware private circuits. In *CCS*, pages 1799–1812. ACM, 2022.

Koc96. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

MAN$^+$19. Lauren De Meyer, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. M&m: Masks and macs against physical attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):25–50, 2019.

NRR06. Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

PBGS24. Matthias Probst, Manuel Brosch, Michael Gruber, and Georg Sigl. DOM-REP II. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2024, Tysons Corner, VA, USA, May 6-9, 2024*, pages 112–121. IEEE, 2024.

RBN$^+$15. Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

RFSG22. Jan Richter-Brockmann, Jakob Feldtkeller, Pascal Sasdrich, and Tim Güneysu. VERICA - verification of combined attacks automated formal verification of security against simultaneous information leakage and tampering. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):255–284, 2022.

RMB$^+$18. Oscar Reparaz, Lauren De Meyer, Begül Bilgin, Victor Arribas, Svetla Nikova, Ventzislav Nikov, and Nigel P. Smart. CAPA: the spirit of beaver against physical attacks. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 121–151. Springer, 2018.

RSM21. Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, and Amir Moradi. Impeccable circuits III. In *IEEE International Test Conference, ITC 2021, Anaheim, CA, USA, October 10-15, 2021*, pages 163–169. IEEE, 2021.

SAK. SAKURA. Side-channel Attack User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html.

SBJ$^+$21. Sayandeep Saha, Arnab Bag, Dirmanto Jap, Debdeep Mukhopadhyay, and Shivam Bhasin. Divided we stand, united we fall: Security analysis of some SCA+SIFA countermeasures against sca-enhanced fault template attacks. In Mehdi Tibouchi and Huaxiong Wang, editors, *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*, volume 13091 of *Lecture Notes in Computer Science*, pages 62–94. Springer, 2021.

SM21. Aein Rezaei Shahmirzadi and Amir Moradi. Re-consolidating first-order masking schemes nullifying fresh randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):305–342, 2021.

SRM20. Aein Rezaei Shahmirzadi, Shahram Rasoolzadeh, and Amir Moradi. Impeccable circuits II. In *57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020*, pages 1–6. IEEE, 2020.

TNN24. Dilara Toprakhisar, Svetla Nikova, and Ventzislav Nikov. CAPABARA: A combined attack on CAPA. In Romain Wacquez and Naofumi Homma, editors, *Constructive Side-Channel Analysis and Secure Design - 15th International Workshop, COSADE 2024, Gardanne, France, April 9-10, 2024, Proceedings*, volume 14595 of *Lecture Notes in Computer Science*, pages 76–89. Springer, 2024.