My Gadget Just Cares for Me - How NINA Can Prove Security Against Combined Attacks

Siemen Dhooghe(✉) and Svetla Nikova(✉)

imec-COSIC, KU Leuven, Leuven, Belgium
{siemen.dhooghe,svetla.nikova}@esat.kuleuven.be

Abstract. Differential Power Analysis and Differential Fault Analysis threaten the security of even the most trustworthy cryptographic primitives. It is important we protect their implementation such that no sensitive information is leaked using side channels and it withstands injected faults or combined physical attacks.

In this work, we propose security notions tailored against advanced physical attacks consisting of both faults and probes on circuit wires. We then transform the security notions to composable security notions. The motivation for this research includes the ease of verification time; the creation of secure components; and the isolation of primitives in larger protocols such as modes of operations. We dub our notion *NINA*, which forms the link between the established *Non-Interference (NI)* property and our composable active security property, *Non-Accumulation (NA)*.

To illustrate the NINA property, we use it to prove the security of two multiplication gadgets: an error checking duplication gadget and an error correcting duplication gadget. The NINA proofs for error detecting gadgets capture the effect of Statistical Ineffective Fault Analysis (SIFA), an attack vector which threatens most current masked implementations. Additionally, we study error correcting techniques. We show that error correcting gadgets can attain the *Independent NINA* property. A stronger property which captures a clear separation between the effect of faults and probes. Thus, we show that clever error correcting gadgets improve on error detecting ones by achieving significant higher levels of combined security along with guaranteed output delivery.

Keywords: Combined security · DFA · DPA · Masking · SIFA

1 Introduction

Differential Fault Analysis (DFA), proposed by Biham and Shamir in 1997 [8], is an attack on a physical device which effectively reveals the secret key of a cipher using well-placed faults in the encryption procedure. Differential Power Analysis (DPA) is an attack which uses a cryptographic device's power consumption to launch a divide-and-conquer attack on the private key as first described by

S. Jarecki (Ed.): CT-RSA 2020, LNCS 12006, pp. 35–55, 2020.
https://doi.org/10.1007/978-3-030-40186-3_3

Kocher et al. in 1999 [26]. To facilitate key-extraction, several physical attacks can be used against the implementation, we differentiate passive, active, and combined attacks. Passive attacks observe the behaviour of a device during its process, such as observing the process time or the device's power consumption. Active attacks tamper with the device's functioning, such as inducing computational errors by fault injections. Using passive and/or active attacks for either enhanced tampering or observation of the device's reaction to tampering are called combined attacks.

In order to defend against physical attacks without using expensive custom hardware such as shields and detectors it is the algorithm that needs to counteract passive, active, and combined attacks by securing it in a formal security model. Passive adversary models and their corresponding security notions have improved significantly over the last fifteen years, largely due to the introduction of the probing adversary by Ishai et al. [25]. This adversary is capable of reading the exact values on a number of circuit wires. The minimal number of wires the adversary observes to learn a sensitive variable is defined as the order of probing security. Duc et al. showed that the noisy leakage model [10,30] reduces to the probing model assuming the presence of sufficient noise and independent wire leakage, and more specifically that an implementation's signal to noise ratio is exponentially related to its probing security order [18]. While the probing model helps to verify implementations, the time complexity is exponential in the security order which is therefore not cost effective for larger implementations such as symmetric ciphers. To streamline this verification procedure, Barthe et al. proposed a composable security definition called Strong Non-Interference (SNI) [3]. This approach views circuits as the composition of several components and forms a sufficient security condition, such that when multiple components are linked together the total circuit is probing secure. Composable security definitions allow designers to verify and optimise separate circuit components which are small enough for a brute force verification technique. This technique has been adopted in several tools to quickly verify implementations based on modular designs [4,7,13]. The importance of a formal security notion, such as the probing model, includes the need of assurance in high-end secure devices. To guarantee such assurance, the Common Criteria was proposed as an international standard. These criteria specify the security and assurance users can have in their sensitive devices where the strongest criterion requires a target of evaluation to have a verified design which is only possible with formal security notions [20].

Apart from security models, the current literature provides countermeasures against passive attacks. One example is the methodology of Ishai, Sahai and Wagner (ISW) which guarantees protection of arbitrary circuits against passive attacks using the previous discussed probing model [25]. This countermeasure led to further study to increase its security and efficiency [5–7,9,12,19,22,34]. Another methodology to secure implementations is described in Threshold Implementations by Nikova et al. [28]. By extensively using the masking scheme's and the cipher's properties, they minimise the countermeasure's latency and

randomness costs and, as a result, the method has been used to defend various symmetric primitives [2,15,23,27,29].

Despite having formal security notions and countermeasures against passive attacks, there are only few works which consider active and combined attacks. The first is Private Circuits II [24] which provides a countermeasure where the active adversary is modelled as one who faults a bounded number of wires per clock cycle. By viewing faults as probes, the work naturally offers protection against a combined adversary. However, the implementation of the countermeasure and its efficiency is currently still a challenge [14]. Later on, the work of ParTI [33] proposes to encode intermediate variables with error correcting codes to detect errors. To protect against passive attacks, they apply threshold implementations on top of the encoding. The results are promising as they succeed in protecting the LED cipher on FPGA. However, they only provide argumentation for active security leaving out combined security and a formal adversary model. As efficiency is a major concern for practical applications, the work of Impeccable Circuits [1] only focuses on active attacks to find very efficient countermeasures. They consider an adversary who faults up to a given number of gates and consider compositional security, i.e., they look at the propagation of faults in their components. Previous works looked at adversaries faulting and reading separate wires, the work of CAPA [31] considers stronger adversaries. They use multiparty computation to provide provable security against combined attacks by proposing a new adversary model, the tile probe-and-fault model. This model considers an adversary who is capable of reading and faulting whole areas in the implementation thus ensuring hardware protection against combined attacks. However, due to their security model the countermeasures are heavy.

The adversaries considered in Private Circuits II and Impeccable Circuits are a good start towards formalising active and combined security but they do not yet allow for composable combined security definitions which are needed by designers. In this work, we combine the wire faulting adversary with the usual probing adversary to consider an attacker who can read and fault a given number of wires in a circuit. Similar to the proposition of Non-Interference by Barthe et al. [3], we build further on our adversary model by considering a modularised circuit and proposing sufficient security conditions (Strong Non-Accumulation and Strong NINA) such that modular compositions remain secure.

1.1 Contributions

The focus of our work is to propose compositional security notions which capture active and combined attacks. We propose the following three security models which provide either composable active or combined security.

– **Non-Accumulation.** With the Non-Accumulation (NA) model, we require that an injected fault only affects one output share of the gadget. Thus, an injected fault does not spread (accumulate) to more shares allowing the use of error detection mechanisms to identify whether faults have occurred in the design. As a result, the NA model effectively moves the verification process from large circuits to smaller subcomponents.

- **NINA.** The models of Non-Interference (NI) and Non-Accumulation (NA) are combined to form the NINA model capturing combined security. The model requires that a probed and faulted gadget returns an output where only a few output shares are faulted and where the adversary learns only a subset of the input shares. Due to NINA simulating the correctness of the unmasked output, it captures attackers using ineffective faults [11].
- **Independent NINA.** As the NINA notion requires the provision of shares to the simulator for every fault or probe injected in the gadget, its provided combined security is limited. We propose a stronger notion, dubbed Independent NINA, which separates the effect of faults and probes, and relaxes the requirement of giving shares to the simulator for each injected fault. The ININA notion can be attained by a gadget using error correction techniques and clever use of injected randomness.

To show our security notions in action, we propose two Strong NINA (SNINA) secure multiplication gadgets.

- **Error Detection:** We propose a multiplication gadget using duplicated Boolean shares with an error detecting mechanism. We show that the gadget is vulnerable to a Statistical Ineffective Fault Attack (see [17]) but the probability for the attack to succeed can be made arbitrarily small by increasing the number of shares. Thus, we prove that the gadget still attains SNINA security. Last, we provide an abort mechanism to show the gadget does not rely on an ideal abort command.
- **Error Correction:** For the second construction, we adapt the error detecting gadget and add error correction methods. The result is a gadget which is impervious to ineffective faults and, moreover, we show the gadget achieves the stronger notion of Strong ININA. This notion proves that the level of combined security is higher than the error detection variant, i.e., the adversary does not gain any advantage by using faults in addition to probes. This shows that, although error correction techniques are more expensive, they give a significant increase in protection against combined attacks as well as guaranteed output delivery.

For the proofs of the composability of the NINA notion and the security of our proposed gadgets, we refer to the full version of the work [16].

2 The Circuit Model and Secret Sharing

We introduce gadgets, private circuits, and the notion of simulatability. Similar to [25], we represent computations in arithmetic circuit form, a directed acyclic graph whose nodes are operations over a finite field $\mathbb{F}$ and whose edges are wires. Additionally, we consider probabilistic arithmetic circuits, meaning circuits with nodes having no input and uniform random elements over $\mathbb{F}$ as output; this randomness is independent and identically distributed, and the correctness of the circuit is not dependent on it. In order to resist fault attacks, we consider

nodes with no output and which can abort the computation. This abort signal works as a broadcast making all wires in the circuit read $\perp$ when the signal is sent out.[1] The adversary also receives this abort signal as it can view from the state of the output whether the circuit aborted or not.

In order to defend algorithms against side-channel attacks a sound and widely deployed approach is the masking countermeasure which was introduced at the same time by Chari et al. [10] and by Goubin and Patarin [21]. The technique splits each key-dependent variable x in the algorithm into shares x_i such that $x = \sum_i x_i$ over a finite field $\mathbb{F}$. In case this field is binary, this masking method is referred to as Boolean masking. If no d shares give information on the secret we say that the masking scheme has a passive threshold d. We also work with independent share vectors x and y as those where the shares of x are independent from the shares of y.

To defend an algorithm against fault attacks the core idea is to utilise redundancy to enable detection of the injected faults. This redundancy is found in encoding intermediate variables using error detecting codes. A popular encoding method is to duplicate intermediate variables, such that, by checking whether all duplicates are equal, an algorithm can detect injected faults. If all sets of k faulty shares in a share vector are detectable, we say that the encoding scheme has an active threshold k.

Using masking and encoding of variables as the core idea to protect secrets against passive and active attacks, we introduce terminology to protect algorithms. A probabilistic circuit with shared inputs/outputs and, if needed, the capability to abort the computation is dubbed a gadget.

Definition 1 (Gadget). *A gadget G is a probabilistic circuit with input in $\mathbb{F}^{nm}$ (m inputs where each input is divided into n shares), uniform randomness $r \in \mathbb{F}^{\alpha}$, and a shared output in $\mathbb{F}^{nm'}$ or abort $\perp$.*

Concerning symmetric primitives, the secrets are each potential intermediate variable of the primitive. In other words, to protect the primitive against passive or active attacks, it works solely over shared variables.

Additionally, we define private circuits as probabilistic circuits consisting of a gadget, where its inputs are first shared and the shared outputs are reconstructed.

Definition 2 (Private Circuit [25]). *A private circuit implementing the function $f : \mathbb{F}^m \to \mathbb{F}^{m'}$ is defined by a triple $(\mathcal{I}, \mathcal{C}, \mathcal{O})$, where*

- *$\mathcal{I} : \mathbb{F}^m \to \mathbb{F}^{nm}$ is a probabilistic circuit with uniform randomness, called input encoder;*
- *$\mathcal{C} : \mathbb{F}^{nm} \to \mathbb{F}^{nm'}$ is a gadget with uniform randomness and the ability to abort;*
- *$\mathcal{O} : \mathbb{F}^{nm'} \to \mathbb{F}^{m'}$ is a circuit with the ability to abort, called output decoder.*

Since we will be working with composable security definitions, we typically consider that private circuits are composed of several gadgets, i.e., the output of one gadget forms the input of another.

[1] On hardware this functionality is replaced a specialised mechanism such as a cascading gadget from [24].

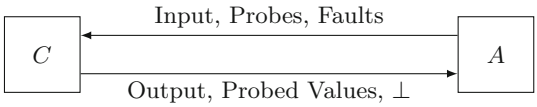

Fig. 1. Interaction between a circuit C and an adversary A.

We aim to protect against passive, active or combined adversaries as those who interact with a circuit by placing probes, faults, or both respectively. As shown in Fig. 1, the circuit responds to this adversary by setting or toggling the values on the faulted wires and returning the values on the probed wires. The state of the abort signal (true or false) is returned as well.

In order to make simulation based proofs for the secrecy of shared variables in gadgets, we define simulatability similar to the definitions proposed in [5,9]. However, we additionally consider that up to k wires in that gadget have been faulted and that the gadget can abort. Here the adversary (distinguisher) is either interacting with the actual gadget or with a simulator. This simulator is given only a part of the input and does not know the secrets of the gadget. The distinguisher's goal is to determine whether it is interacting with the simulator or with the actual gadget. A failure to do so implies that the adversary can know at most the shares given to the simulator and as a result only some inputs of the gadget.

Definition 3 (Simulatability). *Let $P = \{p_1, ..., p_d\}$ be a set of d probes of a gadget C with m inputs where each input is divided into n shares. Let the set of q shares of each input given to the simulator be denoted by $I = \{(i_1, j_1), ..., (i_m, j_q)\} \subset \{1, ..., m\} \times \{1, ..., n\}$. Let $F = \{(f_1, e_1), ..., (f_k, e_k)\}$ be a set of k injected faults e_i (either set or add) on the wire f_i in C. Denote $C_{P,F}$ as the circuit C with probed wires as per P and injected faults as per F. Finally, let $\perp \in \{0, 1\}$ denote the state of the abort signal in the circuit.*

We define the simulator S and distinguisher D as the following probabilistic functions.

$$S : \mathbb{F}^q \times \mathbb{F}^m \times \mathbb{F}^k \to \mathbb{F}^d \times \{0, 1\}$$

$$D : \mathbb{F}^d \times \{0, 1\} \times \mathbb{F}^k \times \mathbb{F}^{nm} \to \{0, 1\}$$

We say that the set of probes P and the state of the abort signal $\perp$ of the faulted circuit C_F can be simulated with the set of values on the input wires I if there exists a simulator S, such that for any distinguisher D and any inputs $a_{*,*}$, we have that

$$\left| \Pr[D(C_{P,F}(a_{*,*}), F, a_{*,*}) = 1] - \Pr[D(S(I, F), F, a_{*,*}) = 1] \right|$$

is negligible in the passive threshold of the sharing scheme, where the probability is taken over the random coins in C, S and D.

We note that for composable security, as we will see later on, we require that the probability for the distinguisher to view the difference between the circuit

and the simulator is negligible and we should take care composing gadgets when it is not.

3 Security Definitions

In this section we specify orders of passive, active, and combined security and expand them to composable security notions which is the focus of our work.

3.1 Orders of Security

Passive Security. To model passive security we consider the known probing adversary who can read the exact values of up to a threshold number of wires in a gadget. The order of passive security is the well-known order of probing security.

Definition 4 (Order of passive security [25,32]). *A private circuit is d^{th}-order passive secure (d^{th}-order probing secure) if every d-tuple of the gadget's intermediate variables is independent of any sensitive variable.*

Active Security. We ensure protection against an adversary who is capable of faulting a given number of wires in the circuit. We note that similar adversaries have been proposed in Private Circuits II [24] and Impeccable Circuits [1]. The order of active security is determined by the number of wires in the circuit the adversary needs to fault in order to create an incorrect output. Such incorrect outputs are important as they can activate DFA attacks, thus to secure implementations we require that the private circuit either gives back a correct output or the process is aborted.

Definition 5 (Order of active security). *A private circuit is k^{th}-order active secure if any set of k faults on the gadget's intermediate variables results in either abort $\perp$ or a correct output (reconstructed output of the unfaulted circuit).*

Note that active security guarantees output correctness and does not consider fault attacks which target the privacy of a scheme such as ineffective faults.

Combined Security. We protect against a combined adversary who both faults and probes wires and consider a private circuit secure if it retains both its privacy and correctness against the combined adversary. This gives us the following combined security definition.

Definition 6 (Order of combined security). *A private circuit is (d, k)-order combined secure if for any set of k faults and d probes on the gadget's intermediate variables, the following holds.*

(a) Privacy: The probed d-tuple with the state of the abort signal is independent of any sensitive variable.
(b) Correctness: The circuit either aborts $\perp$ or gives a correct output.

The combined security model with $d = 0$ still differs from the active security model as the combined security model considers that an adversary can use the knowledge on the state of the abort signal to derive the private circuit's internal variables. The difference between the two models thus lies in the combined security model looking at both the privacy and correctness of a circuit while active security only considers its correctness.

3.2 Composable Notions of Security

We note that the previously discussed security conditions are not composable, i.e., the composition of multiple secure gadgets can be insecure. Thus, the previous security conditions should be applied to the entire implementation, instead we look at composable security notions.

Passive Security. The security notion for composable passive security has been studied by Barthe et al. [3] who defined the notion of Non-Interference (NI) using simulation based security (see Definition 3).

Definition 7 (d Non-Interferent (d-NI) [3]). *A gadget G is d-NI if any set of at most $d' \leq d$ probes can be simulated with at most d' shares of each input.*

Intuitively, the above model grants composable security since a probed value in a gadget can be simulated with an input share, which on its turn is the output share of a previous gadget. In case the latter gadget is also non-interferent, this output value can again be simulated with an input share. This chains until we reach the encoding function in a private circuit (Definition 2). Since the adversary can only probe d values we only need to use a secret sharing scheme of passive threshold at least d to protect against our probing adversary. While the notion of non-interference is a good start and captures a composable security notion over the serial composition of gadgets, the notion is not sufficient to provide protection when gadgets are composed in parallel (e.g., when two gadgets share the same input). To this end Barthe et al. introduced the notion of Strong Non-Interference (SNI).

Definition 8 (d-Strong Non-Interferent (d-SNI) [3]). *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 by only d_1 shares of each input.*

We note that intermediate variables can also be the input or output variables of the gadget.

When the above notion of non-interference is combined with a sharing scheme with a high enough passive threshold, the composable notion provides for probing security.

Active Security. Recall that we defined the order of active security as the maximal number of faulty wires such that the circuit still returns a correct output. We now make this into a composable notion, thus we look at the effect of a fault in a gadget which is part of a larger whole. Ideally an injected fault in the gadget is not propagated, i.e., the fault does not affect the output of that gadget. However, the adversary can always fault its output directly, meaning that we can never guarantee that all outputs of a faulted gadget are correct. Instead, we are interested in gadgets which do not accumulate faults. In other words, we need a fault on a single input or intermediate wire to affect only a single output of the gadget. We relax this requirement by allowing countermeasures to abort the computation (e.g., by using error detecting methods). We thus find composable active security notions which are similar in nature to the definitions of NI and SNI discussed earlier. Our first notion is Non-Accumulation (NA).

Definition 9 (k-Non-Accumulative (k-NA)). *A gadget G is k-NA if for any set of $k' \leq k$ errors, the gadget either aborts or gives an output with at most k' errors.*

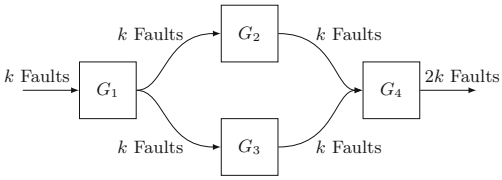

Fig. 2. An example of the propagation of faults over several k-NA gadgets for which a stronger composability notion is needed.

For a gadget which is k-NA, k faults on its intermediate variables result in the gadget giving an output with at most k faults. When composing gadgets, a stronger notion of non-accumulation is needed to guarantee the security of the composition. For example, consider the case given in Fig. 2 where each gadget G_i is k-NA. If an adversary injects k faults in the input of G_1, the gadget will give an output with at most k faulty shares. These faults propagate to the inputs of G_2 and G_3 which, because both gadgets are k-NA, results in a worst case scenario where G_4 gets an input with a total of $2k$ faulty shares. The end result is a sharing with $2k$ faulty shares even though only k faults were injected. To avoid such an accumulation of faults, one needs gadgets which are capable of erasing the errors from their input. The following definition of Strong Non-Accumulation (SNA) is sufficient to arbitrarily compose gadgets and be assured of their active security.

Definition 10 (k-Strong Non-Accumulative (k-SNA)). *A gadget G is k-SNA if for any set of k_1 errors on each input and every set of k_2 errors on the intermediate variables, with $k_1 + k_2 \leq k$, the gadget either aborts or gives an output with at most k_2 errors.*

When the non-accumulation notions are combined with a sharing scheme with a high enough active threshold, the composable notions provide active security.

Combined Security. We now look at composable security notions considering circuits which are both probed and faulted. First, we need to guarantee the correctness of the output of each gadget. To capture the effect of faults in compositions of gadgets, we use an argument similar to the one on active security. Thus, we need that an injected fault in a gadget propagates to at most one output share. However, the adversary can now place probes and thus learn part of the computation made in the gadgets. As a result, the combined security notion needs to capture the probability of an adversary breaking the correctness of a gadget given several faults and probes. In this work we only propose countermeasures where the correctness can not be broken, to give an example of a countermeasure for which this probability is non-trivial we refer the reader to the CAPA countermeasure [31]. Apart from guaranteeing the correctness of a gadget, we also guarantee its sensitive variable privacy for which we use simulation based arguments similar to non-interference. As mentioned by Clavier et al. [11], fault injections can act as a probing tool (think of an adversary faulting away the randomness in a gadget). Thus, we treat faults as probes giving extra shares to the simulator per injected fault (though we see later on that this is not always needed). Additionally, to give the designer the freedom to make countermeasures more efficient we consider security with abort. To capture the effect of the abort signal potentially revealing secrets in the gadget, we require the simulator to reproduce this signal given the injected errors and some input shares. As a result, we design a composable security notion of order (d, k) such that the gadget is (d', k')-order combined security for all sets of $d'+k' \leq d$ probes and $k' \leq k$ faults. We dub our notion NINA derived from the concatenation of the names Non-Interference (NI) and Non-Accumulation (NA).

Definition 11 ((d, k)-NINA). *A gadget G is (d, k)-NINA if for any set of $k' \leq k$ errors and any set of d' probes, such that $d' + k' \leq d$, the following holds.*

(a) Privacy: The probes and the abort signal can be simulated with $d' + k'$ shares of each input and the injected errors.
(b) Correctness: The gadget either aborts or gives an output with at most k' errors.

The NINA notion, combined with a sharing scheme having a sufficient passive and active threshold, implies the notion of combined security (see Definition 6). This follows from the simulation based security stating that the adversary can learn up to a threshold number of the gadget's inputs which, if lower than the passive threshold of the sharing scheme, gives no information on the gadget's secrets. Similarly, since the adversary can only fault up to a threshold number of outputs, a decoding gadget can detect or correct those errors given that the sharing scheme has enough redundancy in it. A formal proof of this implication is found in the full version of the paper.

Theorem 1 *A (d,k)-NINA gadget G with input encoding $\mathcal{I}$ and output decoding $\mathcal{O}$ using a secret sharing scheme with passive threshold at least d and active threshold at least k is (d',k')-order combined secure for any $d'+k' \leq d$ and $k' \leq k$.*

As a result, if we prove a gadget is NINA, we know it is combined secure. However, just as with non-interference, the NINA notion is not sufficient for composability. To this end we introduce "Strong NINA" (SNINA).

Definition 12 ((d,k)-SNINA). *A gadget G is (d,k)-SNINA if for any set of k_1 errors on each input and k_2 intermediate errors, any set of d_1 intermediate probes, any set of d_2 probes on the output, such that $d_1+d_2+k_1+k_2 \leq d$ and $k_1+k_2 \leq k$, the following holds.*

(a) Privacy: The probes and the abort signal can be simulated with $d_1+k_1+k_2$ shares of each input and the injected errors.
(b) Correctness: The gadget either aborts or gives an output with at most k_2 errors.

The notion of SNINA is sufficient for composability. In other words the composition of two SNINA gadgets is again SNINA (a proof is given in the full version).

Theorem 2. *The composition of two (d,k)-SNINA gadgets is (d,k)-SNINA.*

The above theorem together with Theorem 1 implies that the notion of SNINA is a sufficient condition to achieve composable combined security. The relations between the SNINA notion and other security models is shown in Fig. 3.

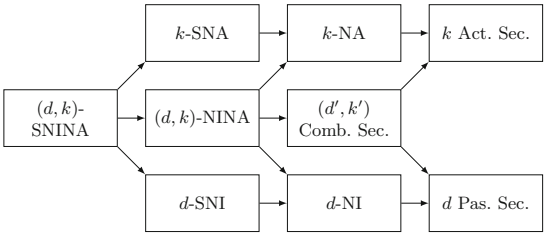

Fig. 3. A short overview of security models and relations between them.

Nevertheless, we find that there is a stronger property than NINA which gives improved protection. In case we use error correcting techniques instead of error detecting ones, specialised gadgets can attain a stronger security condition where faults are no longer modelled as probes. Thus, we propose a security notion where we claim an adversary can not learn anything by faulting a gadget which manifests itself in the security definition as the simulator not getting an extra input share for an injected fault. The result of this change is captured in the following definition which we dub "Independent NINA" or ININA.

Definition 13 ((d,k)-ININA). *A gadget G is (d,k)-ININA if for any set of $k' \leq k$ errors and any set of d' probes, such that $d' \leq d$, the following holds.*

(a) Privacy: The probes can be simulated with d' shares of each input and the injected errors.
(b) Correctness: The gadget gives an output with at most k' errors.

The above definition can again be made into a property which is sufficient for arbitrary compositions. This gives us the notion of "Strong Independent NINA" or SININA for short.

Definition 14 ((d,k)-SININA). *A gadget G is (d,k)-SININA if for any set of k_1 errors on each input and k_2 intermediate errors, any set of d_1 intermediate probes, any set of d_2 probes on the output, such that $d_1 + d_2 \leq d$ and $k_1 + k_2 \leq k$, the following holds.*

(a) Privacy: The probes can be simulated with d_1 shares of each input and the injected errors.
(b) Correctness: The gadget gives an output with at most k_2 errors.

It is evident that the ININA notions are stronger than the NINA notions, thus the above notions also provide combined security. However, the notion provides directly (d,k)-combined security instead of (d',k')-order combined secure for any $d' + k' \leq d$ and $k' \leq k$. The proof of the following theorem is given in the full version of the paper.

Theorem 3. *A (d,k)-ININA gadget G with input encoding $\mathcal{I}$ and output decoding $\mathcal{O}$ using a secret sharing scheme with passive threshold at least d and active threshold at least k is (d,k)-order combined secure.*

As a result, using the same masking scheme, a SININA secure gadget provides significant improved combined protection over an SNINA secure gadget.

Similar to SNINA, SININA is sufficient for composability. In other words the composition of two SININA gadgets is again SININA (a proof is given in the full version of the paper).

Theorem 4. *The composition of two (d,k)-SININA gadgets is (d,k)-SININA.*

4 Combined Secure Duplicated Boolean Masking

In this section we introduce a combined secure methodology for an arbitrary security order. We work over bits $\mathbb{F}_2$, share values using Boolean secret sharing and encode using duplication. We first quickly introduce the secret sharing scheme and then move on to show our methodology. The security of the gadgets is proven in the full version of the paper.

4.1 Duplicated Boolean Masking

For the proposed countermeasures, we make use of a duplicated Boolean masking approach which shares a secret x as a vector

$$(x_{1,1}, ..., x_{1,k+1}, x_{2,1}, ..., x_{d+1,k+1}),$$

such that $\sum_{i=1}^{d+1} x_{i,\ell} = x$ for all $\ell \in [k+1]$ and $x_{i,1} = ... = x_{i,k+1}$ for all $i \in [d+1]$. This method has a passive threshold d meaning that no d shares give information on the secret x and an active threshold k meaning that any faults on at most k shares could be detected in the share vector.

4.2 Duplicated Boolean Methodology

We recall that our secret sharing scheme has a passive threshold d, meaning that an adversary needs to view at least $d+1$ shares to recover the secret, and an active threshold k, thus an adversary needs to inject at least $k+1$ errors for the fault to be undetectable. We note that our methodology is similar to the one from Private Circuits II [24]. The pseudo-code to secret share a value is given in Algorithm 1.

Algorithm 1. Duplicated Boolean sharing a secret a

Input: Secret a and uniform random values r_i
Output: Duplicated Boolean shares of a

```
for ℓ ← 1 to k + 1 do
    for i ← 1 to d do
        a_{i,ℓ} ← r_i;
    end
    a_{d+1,ℓ} ← a + Σ_{i=1}^{d} a_{i,ℓ};
end
```

The addition between independent shared variables is quite simple and needs only component-wise addition between the shares. Thus, the addition between the sharing of a and b, giving a sharing of $c = a + b$, is made by $c_{i,\ell} = a_{i,\ell} + b_{i,\ell}$. To secure operations between shares and constants we ensure that the constant is not a single point of failure, as such it also needs to be duplicated, namely each constant is replicated $(k+1)$ times to form a $(k+1)$ tuple which is the encoded value of the constant. The addition of a shared value a with a constant c is done by adding the duplicated constant to the duplicated first Boolean share of the variable.

$$\forall \ell \in [k+1] : \; a_{1,\ell} \leftarrow a_{1,\ell} + c_\ell$$

A multiplication with a constant is done by multiplying the duplicated constant to each share.

$$\forall i \in [d+1], \forall \ell \in [k+1] : \; a_{i,\ell} \leftarrow a_{i,\ell} \cdot c_\ell$$

Since the above operations are all local, they are evidently (d, k)-NINA.

While linear operations are easily implemented, the multiplication between shared and encoded variables is more difficult. We give pseudo-code of our multiplication gadget in Algorithm 2. The gadget starts by multiplying two independent share vectors of a and b to create all cross products of the form $a_i b_j$. These cross products are then remasked by adding unique randomness $r_{i,j}$ created by an RNG (which is important for the SNI property). Since we add the same randomness over all duplicated cross products ($u_{i,j,\ell}$ for $\ell \in [k+1]$) all these cross products should equal each other if no fault was injected. As a result, we can error check them (which is important for the SNA property).[2] To detect errors in the cross products it is sufficient to compare a share to all its duplicated versions, in symbols:

$$\forall i, j \in [d+1], \forall \ell \in [k+1] : \; u_{i,j,1} = u_{i,j,\ell} \, .$$

Since we are working over bits, this translates to aborting the computation in case one of the $u_{i,j,1} + u_{i,j,\ell}$ is equal to 1. This abort operation is considered as a command causing all variables in the implementation to read $\perp$ as explained in Sect. 2 (in Sect. 4.3, we describe a cascading gadget in case an abort operation is not available). In case no error is detected, the gadget sums up all the cross products for different indices j and returns a duplicated Boolean sharing of ab. The proof that this multiplication procedure is SNINA is given in the full version of the paper. From this proof we see that there is a statistical ineffective fault attack (see [17]) which breaks the privacy of the algorithm. This attack works as follows, the adversary adds a non-zero fault to one of the $a_{i,\ell}$ shares (similarly $b_{i,\ell}$ shares). In case the operation does not abort, the adversary learns that all $b_{i,\ell} = 0$ (similarly all $a_{i,\ell} = 0$), which means the adversary learns an input secret and breaks the privacy of the gadget. The probability for this attack to succeed is equal to $1/|\mathbb{F}_2|^{d+1}$. Due to the attack aborting the computation when it fails, this attack does not threaten the composability of the gadget.[3] To increase the protection against the ineffective fault, the probability for the attack to succeed needs to be made sufficiently small which is done by increasing the number of shares or by increasing the field size $|\mathbb{F}|$. In Sect. 5 we look at an error correcting variant of the multiplication gadget which is not vulnerable to an ineffective fault.

In Algorithm 3 we provide a method to refresh the randomness of a shared variable and check whether there are errors present on its shares. A proof of the SNINA condition of Algorithm 3 is given in the full version of the work. We note that this gadget can be used to transform a NINA secure operation into its SNINA variant by serially composing the NINA gadget with Algorithm 3.

[2] Note that if an adversary injects a fault directly in one of the random values $r_{i,j}$, it would not be detected. Nevertheless, the gadget still outputs a valid duplicated Boolean sharing so it does not affect the correctness of the gadget. This fault should be carefully investigated for its effects on the gadget's privacy.

[3] To clarify, the passive threshold of the sharing does not need to increase to assure composability due to the only attack, causing simulator failure, aborting the computation on success.

Algorithm 2. Multiplying duplicated Boolean shared values

Input: Independent shares of a and b, and uniform random $r_{i,j}$
Output: Shares of ab or $\perp$

for $\ell \leftarrow 1$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $u_{i,i,\ell} \leftarrow a_{i,\ell} b_{i,\ell}$;
    for $j \leftarrow i+1$ **to** $d+1$ **do**
      $u_{i,j,\ell} \leftarrow a_{i,\ell} b_{j,\ell} + r_{i,j}$;
      $u_{j,i,\ell} \leftarrow a_{j,\ell} b_{i,\ell} + r_{i,j}$;
    end
  end
end
for $\ell \leftarrow 2$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    for $j \leftarrow 1$ **to** $d+1$ **do**
      $t_{i,j,\ell} \leftarrow u_{i,j,1} + u_{i,j,\ell}$;
      if $t_{i,j,\ell} = 1$ **then return** $\perp$;
    end
  end
end
for $\ell \leftarrow 1$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $c_{i,\ell} \leftarrow \sum_{j=1}^{d+1} u_{i,j,\ell}$;
  end
end

This follows from Theorem 5 which states that the serial composition between a NINA gadget and an SNINA gadget is again SNINA. The proof of this theorem is found in the full version of the paper.

Theorem 5. *The serial composition of a single input, output (d,k)-NINA gadget with a (d,k)-SNINA gadget is again (d,k)-SNINA.*

Thus, sometimes one can substitute SNINA gadgets with NINA ones without sacrificing security. This reduces costs as NINA secure gadgets are generally more efficient than their SNINA variants.

Together, all gadgets described in this section form a methodology to secure arbitrary circuits as each algorithm over a finite field can be described in terms of additions and multiplications.

4.3 A Cascading Gadget

In case an abort mechanism is not available, we provide a circuit which erases all data when a fault is detected. This method is similar to the cascading gadget described in [24] and thus we lend its name. We first make variables for the abort

Algorithm 3. Refreshing and checking a duplicated Boolean sharing

Input: Duplicated Boolean shares of a and uniform random values $r_{i,j}$
Output: Refreshed and checked shares of a or $\perp$

```
for ℓ ← 2 to k + 1 do
    for i ← 1 to d + 1 do
        t_{i,ℓ} ← a_{i,1} + a_{i,ℓ};
        if t_{i,ℓ} = 1 then return ⊥;
    end
end
for ℓ ← 1 to k + 1 do
    for i ← 1 to d + 1 do
        for j ← i + 1 to d + 1 do
            a_{i,ℓ} ← a_{i,ℓ} + r_{i,j};
            a_{j,ℓ} ← a_{j,ℓ} + r_{i,j};
        end
    end
end
```

flag, we consider $\perp_\ell$ for $\ell \in [k]$. A priori, all $\perp_\ell$ are equal to zero, however, when a fault is injected we require that each $\perp_\ell$ is set to one. In case the abort flag equals all one, no $k-1$ faults can change each $\perp_\ell$ back to zero. The above described functionality is implemented by duplicating the error checks in Algorithms 2 and 3. For example, the error checking component (the first lines) of Algorithm 3 would be changed to the following.

```
for m ← 1 to k do
    for ℓ ← 2 to k + 1 do
        for i ← 1 to d + 1 do
            ⊥_m ← (a_{i,1} + a_{i,ℓ}) ∨ ⊥_m;
        end
    end
end
```

From the above algorithm it is clear that in case one of the $a_{i,1}$ does not equal $a_{i,\ell}$, all $\perp_m$ are set to one and no $k-1$ faults can set them all back to zero.

With the above abort flag as a global variable and its functionality as described above, we can easily describe a gadget which erases its input in case a $\perp_m$ is equal to one. We give the pseudo-code of this gadget in Algorithm 4.

In case Algorithm 4 is serially composed with each Algorithm 2 or Algorithm 3, our duplicated Boolean masking methodology is secure against combined attacks without the need of an ideal abort command.

Algorithm 4. Cascading a duplicated Boolean sharing

Input: Shares of a and the abort state $\perp_m$ for $m \in [k]$
Output: The shares of a or all 0

for $\ell \leftarrow 1$ **to** $k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $a_{i,\ell} \leftarrow a_{i,\ell} \prod_{m=1}^{k}(1+\perp_m)$;
  end
end

5 A Correcting Multiplication

In the previous section we gave a combined secure methodology based on detecting errors using duplicated Boolean shares. However, Algorithm 2 is vulnerable against a statistical ineffective fault. To avoid this vulnerability one can use an error correction method instead of an error detection one. As there is no longer an abort signal, a fault does not change the state of the output and as a result ineffective faults are now actually ineffective. Note that this comes at the increased cost of using extra shares and operations to enable error correction.

Instead of just replacing the error detection mechanisms with error correction ones, we go one step further and create an error correcting variant of Algorithm 2 which attains Strong Independent NINA security (Definition 14). Whereas Algorithm 2 was secure against d probes and k faults where the combined number of probes and faults do not exceed d, our new algorithm does not require this restriction thus it is secure against up to d probes and k faults at the same time. In other words, a k-active adversary faulting the new multiplication gadget does not harm the privacy of the gadget.

We introduce the error correcting multiplication gadget. We again work over bits $\mathbb{F}_2$, share values using d Boolean secret shares, but now encode using $2k+1$ duplicated shares (instead of $k+1$ shares). As such, the secret sharing scheme has a passive threshold d, meaning that an adversary needs to view at least $d+1$ shares to recover the secret, and an active threshold $2k$, thus an adversary needs to inject at least $k+1$ errors for the fault to be uncorrectable (note the difference with the undetectability of faults). We give the pseudo-code of the multiplication gadget in Algorithm 5. The error correcting gadget works similar to the error detecting one. It starts by multiplying two independent share vectors of a and b to create all cross products. These cross products are then remasked by adding $k+1$ random elements $r_{i,j,\ell}$ to each of them. As a result, since each cross product is masked by $k+1$ random values, no set of k faults can remove all random values on a cross product. Since we add the same randomness over all duplicated cross products ($u_{i,j,\ell}$ for $\ell \in [2k+1]$) all these cross products still equal each other if no fault was injected. As a result, we can error correct them. An error correction on duplicated shares is done by majority voting the shares. If at least $k+1$ out of $2k+1$ cross products were equal to zero, the result of this majority vote is zero otherwise it is equal to one. For brevity, we denote this

Algorithm 5. Multiplying shares with error correction

Input: Independent shares of a and b, and uniform random $r_{i,j,\ell}$
Output: Shares of ab

for $\ell \leftarrow 1$ **to** $2k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $u_{i,i,\ell} \leftarrow a_{i,\ell} b_{i,\ell}$;
    for $j \leftarrow i+1$ **to** $d+1$ **do**
      $u_{i,j,\ell} \leftarrow a_{i,\ell} b_{j,\ell}$;
      $u_{j,i,\ell} \leftarrow a_{j,\ell} b_{i,\ell}$;
      for $m \leftarrow 1$ **to** $k+1$ **do**
        $u_{i,j,\ell} \leftarrow u_{i,j,\ell} + r_{i,j,m}$;
        $u_{j,i,\ell} \leftarrow u_{j,i,\ell} + r_{i,j,m}$;
      end
    end
  end
end
for $\ell \leftarrow 1$ **to** $2k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    for $j \leftarrow 1$ **to** $d+1$ **do**
      $v_{i,j,\ell} \leftarrow \mathrm{Maj}(u_{i,j,1}, ..., u_{i,j,2k+1})$;
    end
  end
end
for $\ell \leftarrow 1$ **to** $2k+1$ **do**
  for $i \leftarrow 1$ **to** $d+1$ **do**
    $c_{i,\ell} \leftarrow \sum_{j=1}^{d+1} v_{i,j,\ell}$;
  end
end

operation "Maj", where we assume for simplicity that a probing adversary can view all arguments given to the Maj function with one probe. We stress that this error correction procedure is independently applied to each cross product, such that a single fault can only affect one cross product. Our multiplication gadget again ends by summing up all the cross products for different indices j and returns a duplicated Boolean sharing of ab. The proof that this multiplication procedure is SININA is given in the full version of the work.

6 Conclusion

We provided security notions considering circuits with probed and/or faulted wires. We then extended them to active and combined composable notions similar to the extension from the probing model to Non-Interference (NI). The first notion of Non-Accumulation (NA) addresses composable active security which states that a gadget is secure if injected faults affect only one output each.

The second is the notion of composable combined security (NINA). A gadget is considered NINA if an injected fault only affects one output and a fault or probe can be simulated using only one input. We discussed both error detection and error correcting gadgets and showed that the error detection mechanism is prone to ineffective faults whereas error correction comes at an increased cost but gives significantly improved protection (Independent NINA).

The notions for composable security offer the ability to efficiently verify building blocks of larger implementations and allow for the search of efficient functions which achieve security in the corresponding model. Moreover, these composable notions enable us to use secured primitives in a larger whole such as modes of operations.

Acknowledgements. The authors would like to thank Thomas De Cnudde, Adrián Ranea, Vincent Rijmen, and Nigel Smart for their useful comments and ideas.

This work was supported in part by the Research Council KU Leuven: C16/18/004, by the NIST Research Grant 60NANB15D346, and by the EU H2020 project FENTEC. Siemen Dhooghe is supported by a Ph.D. Fellowship from the Research Foundation - Flanders (FWO). Svetla Nikova was partially supported by the Bulgarian National Science Fund, Contract No. 12/8.

References

1. Aghaie, A., Moradi, A., Rasoolzadeh, S., Schellenberg, F., Schneider, T.: Impeccable circuits. Cryptology ePrint Archive, Report 2018/203 (2018)
2. Arribas, V., Bilgin, B., Petrides, G., Nikova, S., Rijmen, V.: Rhythmic Keccak: SCA security and low latency in HW. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(1), 269–290 (2018). https://doi.org/10.13154/tches.v2018.i1.269-290
3. Barthe, G., et al.: Strong non-interference and type-directed higher-order masking. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016, pp. 116–129. ACM (2016). https://doi.org/10.1145/2976749.2978427
4. Barthe, G., Belaïd, S., Fouque, P., Grégoire, B.: maskVerif: a formal tool for analyzing software and hardware masked implementations. IACR Cryptology ePrint Archive 2018, 562 (2018). https://eprint.iacr.org/2018/562
5. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 616–648. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_22
6. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Private multiplication over finite fields. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 397–426. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_14
7. Belaïd, S., Goudarzi, D., Rivain, M.: Tight private circuits: achieving probing security with the least refreshing. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 343–372. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03329-3_12

8. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052259
9. Cassiers, G., Standaert, F.: Improved bitslice masking: from optimized non-interference to probe isolation. IACR Cryptology ePrint Archive 2018, 438 (2018). https://eprint.iacr.org/2018/438
10. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_26
11. Clavier, C.: Secret external encodings do not prevent transient fault analysis. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 181–194. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_13
12. Coron, J.-S.: High-order conversion from Boolean to arithmetic masking. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 93–114. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_5
13. Coron, J.-S.: Formal verification of side-channel countermeasures via elementary circuit transformations. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 65–82. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_4
14. De Cnudde, T., Nikova, S.: More efficient private circuits II through threshold implementations. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, 16 August 2016, pp. 114–124. IEEE Computer Society (2016). https://doi.org/10.1109/FDTC.2016.15
15. De Cnudde, T., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., Rijmen, V.: Masking AES with $d+1$ shares in hardware. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 194–212. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_10
16. Dhooghe, S., Nikova, S.: My gadget just cares for me - how NINA can prove security against combined attacks. IACR Cryptology ePrint Archive 2019, 615 (2019). https://eprint.iacr.org/2019/615
17. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: SIFA: exploiting ineffective fault inductions on symmetric cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 547–572 (2018)
18. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 423–440. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_24
19. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 89–120 (2018). https://doi.org/10.13154/tches.v2018.i3.89-120
20. Gollmann, D.: Computer Security, 3 edn. Wiley (2011). http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1118801326.html
21. Goubin, L., Patarin, J.: DES and differential power analysis the "Duplication" method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48059-5_15

22. Groß, H., Mangard, S., Korak, T.: Domain-oriented masking: compact masked hardware implementations with arbitrary protection order. In: Bilgin, B., Nikova, S., Rijmen, V. (eds.) Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October 2016, p. 3. ACM (2016). https://doi.org/10.1145/2996366.2996426
23. Groß, H., Schaffenrath, D., Mangard, S.: Higher-order side-channel protected implementations of KECCAK. In: Kubátová, H., Novotný, M., Skavhaug, A. (eds.) Euromicro Conference on Digital System Design, DSD 2017, Vienna, Austria, 30 August–1 September 2017, pp. 205–212. IEEE Computer Society (2017). https://doi.org/10.1109/DSD.2017.21
24. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_19
25. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_27
26. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
27. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_6
28. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006). https://doi.org/10.1007/11935308_38
29. Poschmann, A., Moradi, A., Khoo, K., Lim, C., Wang, H., Ling, S.: Side-channel resistant crypto for less than 2, 300 GE. J. Cryptol. **24**(2), 322–345 (2011). https://doi.org/10.1007/s00145-010-9086-6
30. Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_9
31. Reparaz, O., et al.: CAPA: the spirit of beaver against physical attacks. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 121–151. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_5
32. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_28
33. Schneider, T., Moradi, A., Güneysu, T.: ParTI – towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 302–332. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_11
34. Ueno, R., Homma, N., Sugawara, Y., Nogami, Y., Aoki, T.: Highly efficient $GF(2^8)$ inversion circuit based on redundant GF arithmetic and its application to AES design. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 63–80. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_4