Sharing Independence & Relabeling: Efficient Formal Verification of Higher-Order Masking

Roderick Bloem, Hannes Gross, Rinat Iusupov, Martin Krenn, and Stefan Mangard

Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
{firstname.lastname}@iaik.tugraz.at

Abstract. The efficient verification of the security of masked hardware implementations is an important issue that hinders the development and deployment of randomness-efficient masking techniques. At EUROCRYPT 2018, Bloem et al. [6] introduced the first practical formal tool to prove the side-channel resilience of masked circuits in the probing model with glitches. Most recently Barthe et al. [2] introduced a more efficient formal tool that builds upon the findings of Bloem et al. for modeling the effects of glitches. While Barthe et al.'s approach greatly improves the first-order verification performance, it shows that higher-order verification in the probing model with glitches is still enormously time-consuming for larger circuits like a second-order AES S-box, for instance. Furthermore, the results of Barthe et al. underline the discrepancy between state-of-the-art formal security notions that allow for faster verification of circuits. Namely the strong non-interference (SNI) notion, and existing masked hardware implementations that are secure in the probing model with glitches. In this work, we extend and improve the formal approaches of Bloem et al. and Barthe et al. on manifold levels. We first introduce a so-called sharing independence notion which helps to reason about the independence of shared variables. We then show how to use this notion to test for the independence of input and output sharings of a module which allows speeding up the formal verification of circuits that do not fulfill the SNI notion. With this extension, we are for the time able to verify the security of a second-order masked DOM AES S-box which takes about 3 seconds, and up to a fifth-order AES S-box which requires about 47 days for verification. Furthermore, we discuss in which case the independence of input and output sharings lead to composability.

Keywords: masking · formal verification · domain-oriented masking · hardware security · side-channel analysis · private circuits

1 Introduction

Side-channel analysis attacks (SCA), like differential power analysis [16] or electromagnetic emanation analysis [20], are one of the main security threats for

security-critical embedded devices. Masking has proven to be a very effective and scalable countermeasure against SCA for hardware implementations. Extensive research over the last 20 years has led to a broad variety of masking schemes and protected hardware implementations [3, 11–13, 18, 21]. The efficient and secure implementation of masked implementation, however, has also proven to be a difficult task in which flaws are easily overseen. Such flaws can be either introduced by designers of masked hardware implementations themselves, but also by the automated processes of a digital design flow (like synthesis or place-and-route) which can diminish the SCA resistance. These issues make verification of masked implementation before production and deployment inevitable.

Bloem et al. [6] introduced a formal verification approach suitable for hardware implementations that proves the security in the so-called probing model of Ishai et al. [14] under consideration of glitches. With their approach, Boolean masked hardware implementations can be formally proven directly on the netlist of a circuit which was demonstrated (among other examples) for a first-order AES S-box. The formal verification of larger circuits and for higher protection orders, however, is quite time-consuming. The verification of the first-order masked AES S-box required up to 10 hours, for instance.

One very promising approach to speed up the formal verification of masked implementations was originally introduced by Barthe et al. [1] for masked software implementations. Barthe et al. introduced the notion of strong non-interference (SNI) which allows splitting the verification of larger masked software implementations into smaller parts (composability).While the verification performance is increased enormously, the drawback of this approach is the strong requirements on the masked implementations regarding randomness usage to reach this SNI property.

Over the last years, SNI has been extensively researched and used to prove masked software implementations and schemes. The SNI approach has been extended by Faust et al. [10] to also cover the specifics of hardware implementations, and thus models signal timing effects like glitches in a similar way as the approach of Bloem et al. [6]. However, the SNI extension to hardware inherits its strong requirements about randomness usage and does not work for the most randomness efficient masking schemes despite security in the probing model with glitches.

Very recently, Barthe et al. [2] introduced an efficient tool to prove the security of masked hardware implementations using different leakage models including SNI. While Barthe et al.'s tool is already very fast for first-order (e.g., an AES S-box is verified within seconds instead of hours), still higher-order verification of larger circuits like the AES S-box does not finish within reasonable time. Furthermore, the verification results of Barthe et al. show that there is a practically noticeable formal gap between probing security with glitches and the non-interference (NI) based security. This was also shown by Faust et al. [10] on the example of the threshold implementations masking scheme [18].

Our contribution. We first demonstrate that the (strong) non-interference notion is indeed too strong, even for less specific masking schemes than threshold

implementations, but also for $d+1$ masking schemes like the domain-oriented masking scheme (DOM) [13] or the consolidating masking scheme [21]. For this purpose, we first introduce an alternative notion that allows proving the independence of sharings in the probing model with glitches. When this sharing independence notion is proven for the input and output sharings of a module, it follows that the outputs can be treated as independent circuit inputs (relabeling). Relabeling of sharings allows for tremendous verification speedups, as we demonstrate by the tool of Bloem et al. [6] and Barthe et al. [2], while still allowing to prove more general circuits than by SNI. With relabeling, we reduce the verification time of the first-order DOM AES S-box from 80 hours to 10 minutes in the tool of Bloem et al. [6]. Relabeling allows us for the first time to successfully prove the security of the generic DOM AES S-box for second-order probing security with the tool of Barthe et al. [2] which before did not finish within reasonable verification time (experiment stopped after two weeks). Furthermore, we show verification results for the DOM AES S-box design up to order five. We also note that this relabeling technique is independent of the used tool since it allows preprocessing of the circuit such that its logic-depth becomes lower.

Finally, we also argue that composability of circuits can be proven with our independence notion. While the independence of a module's inputs and outputs does not directly lead to composability of the module in the higher-order case as in SNI, it still allows arguing composability for circuits which do not feed-forward the module's inputs. Meaning that any module is composable if its outputs are independent of its inputs and the module's inputs are not connected to any other module (with nonindependent outputs), as it is e.g., the case for the S-box layer of the AES. This is a circumstance that is intuitively already leveraged in many masked hardware implementations works to reason about the probing security over multiple rounds by demonstrating the security of separate parts. With the notion of sharing independence, we now have a formal argument for the correctness of this practice without relying on the strong formal requirements of SNI.

2 Preliminaries

Masking was introduced as a countermeasure against side-channel analysis (SCA) and has been extensively researched over the last almost 20 years. The idea of masking is to counteract SCA attacks by randomizing the representation of security-sensitive data to make physical side-channel information, like power consumption or electromagnetic emanation, statistically independent (Definition 1) from the processed data.

Definition 1 (Statistical independence) *Let A, B, and Q be sets of Boolean variables and let $f : 2^A \times 2^B \to 2^Q$. We say that f is statistically independent of A if for all $q \in Q$ there is a t such that for all $a \in A$ we have $|\{b \mid f(a,b) = q\}| = t$.*

To achieve statistical independence the security-sensitive data is split into a vector of uniformly random shares, which only when combined again result in the original data. A variable a, for example, is represented as a vector of shares (called sharing) $\boldsymbol{a} = \{a_1, a_2, \dots a_n\}$ containing n random shares. We consider Boolean masking in which the sharing function is the exclusive-or (XOR) of all shares such that (although the shares are produced randomly) the relation $a_1 \oplus a_2 \oplus \cdots \oplus a_n = \boldsymbol{a}$ is always fulfilled.

Definition 2 (Sharing) *A sharing $\boldsymbol{a}$ of variable a is a vector with the elements denoted $\{a_1, a_2, ..., a_n\}$ s.t. $a = a_1 \oplus a_2 \oplus ... \oplus a_n$, and any subset of up to d elements of $\boldsymbol{a}$ is statistically independent of the sharing value a.*

The number of shares n is a function of the targeted protection order d of masked circuit. For the sake of simplicity we assume in the remainder of this work that $n = d+1$ which is the minimum number of shares to achieve d^{th}-order security in the probing model of Ishai et al. [14] has become the defacto standard model for the security evaluation of masked circuits.

We define a circuit $C = (\mathcal{G}, \mathcal{W}, R, f, \mathcal{I})$, where $(\mathcal{G}, \mathcal{W})$ is an acyclic directed graph with vertices $\mathcal{G}$ (gates) and edges $\mathcal{W} \subseteq \mathcal{G} \times \mathcal{G}$ (wires). Gates with indegree zero are called inputs I, gates with outdegree zero are called outputs O. $R \subseteq \mathcal{G}$ is a set of registers, f is a function that associates with any gate $g \in \mathcal{G} \setminus I$.

Definition 3 (Secure probes) *Let C be a circuit, $P = \{p_1, ..., p_d\}$ be a set of probes in C, and $f : \mathbb{B}^d \to \mathbb{B}$ be a function of d probes. P is secure if every $f(P)$ is statistically independent of any sharing value in C.*

Definition 4 (d-th order probing security) *Let C be a circuit. C is d-probing secure if any tuple P of up to d probes is secure.*

According to this definition, the shared variable a is securely stored in the share vector $\boldsymbol{a}$ containing $d+1$ shares because a probing attacker cannot collect all shares of a with just d probes. A securely masked circuit thus keeps the shares of a separated at all times.

Fourier expansion. The approach of Bloem et al. [6] expresses the security of a circuit in the probing model by relying on the so-called Fourier expansion of Boolean functions.

Definition 5 (Fourier expansion [19]) *A Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$ can be* uniquely *expressed as a multilinear polynomial in the n-tuple of variables $A = (a_1, a_2, \dots, a_n)$ with $a_i \in \{\pm 1\}$, i.e., the multilinear polynomial of f is a linear combination of monomials, called* Fourier characters*, of the form $\chi_T(A) = \prod_{a_i \in T} a_i$ for every subset $T \subseteq A$. The coefficient of $\chi_T \in \mathbb{Q}$ is called the* Fourier coefficient *$\widehat{f}(T)$ of the subset T. Thus we have the* Fourier representation *of f:*

$$f(A) = \sum_{T \subseteq A} \widehat{f}(T)\chi_T(A) = \sum_{T \subseteq A} \widehat{f}(T) \prod_{a_i \in T} a_i.$$

The statistical dependence (Definition 1) of a function can be directly read from its Fourier expansion using the following lemma.

Lemma 6 (Xiao-Massey [24]) *A Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ is statistically independent of a set of variables $A' \subseteq A$ iff $\forall T \subseteq A'$ it holds that if $T \neq \emptyset$ then $\widehat{f}(T) = 0$.*

(Strong) Noninterference. There also exists other definitions for the security of masked circuits. One of these notions is called non-interference (NI) which implies probing security but allows for simulation-based proofs (for more details we refer the interested reader to [1]).

Definition 7 (d^{th}-order Non-interference [1]) *A circuit is d-noninterfering (d-NI) iff for t_i probes on the circuit's internal signals, and t_o probes on its outputs with $t_i + t_o \leq d$, all probes can be simulated with $t_i + t_o$ shares per shared input variable.*

Barthe et al. [1] also define a more demanding security notion called strong non-interference (SNI). The SNI notion requires that the order for the NI resistance of a circuit only depends on the number of probes that are placed inside the circuit but not on the probes at the output of the circuit. Thereby a separation of the shared input variables and the shared output variables is achieved by adding fresh randomness.

Definition 8 (d^{th}-order Strong Noninterference [1]) *A circuit is d-strong noninterfering (d-SNI) iff for t_i probes on the circuit's internal signals, and t_o probes on its outputs with $t_i + t_o \leq d$, all probes can be simulated with only t_i shares per each shared input.*

Since SNI implies also NI, which ensures that at most one share per variable is combined within the circuit (even for masking schemes that use more than $d + 1$ shares), any circuit that fulfills SNI can serve as input of an NI circuit without compromising (interfering with) its security. Thus this SNI notion allows verifying the security of the whole circuit by performing verification of smaller and independently verifiable parts of the circuit. The SNI notion is thus often referred to as composability notion.

In the next section, we first demonstrate that the NI and the SNI notion are more demanding than actually required by the probing model or a more intuitive form of composability, respectively. We analyze differences and show examples for masked hardware implementations. We then suggest a less restrictive notion based on probing security and on our notion of sharing independence which allows us to speed up the formal verification by proving circuit parts to be independent of its inputs. Our approach, however, does not in all cases allow for independent verification of circuit parts as with SNI, but allows disconnecting parts of the circuit, and thus to lower the maximum logic depth which makes verification much faster as we will demonstrate subsequently.

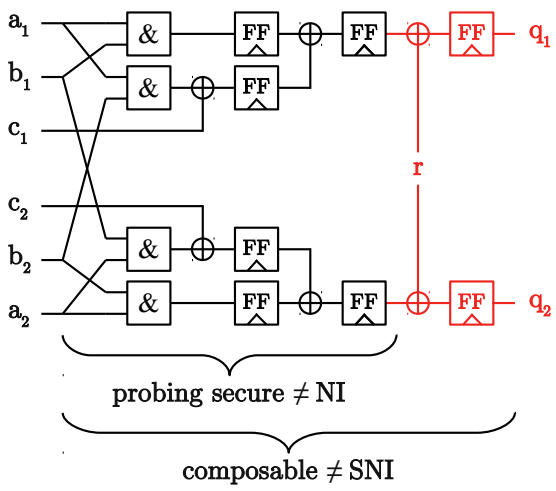

Fig. 1. Example for a first-order probing secure (black parts only) and a composable (when including the red parts) circuit that is neither NI nor SNI

3 Independence and Composability in the Probing Model

Informally speaking, composability refers to a property of a masked circuit module that makes its outputs independent of its inputs and intermediate values by using freshly generated randomness. Composability thus allows neglecting these composable modules from the verification of the overall circuit. The SNI based composability notion by Barthe et al. [1] builds on the notion of NI. While circuits that fulfill the NI notion are secure in the probing model, not all circuits that are probing secure are also NI which was, e.g., demonstrated by Faust et al. [10] for threshold implementations that use more than the minimum number of $d+1$ shares. We first show that NI is too demanding even for masking schemes that use $d+1$ shares.

As an example, we consider the first-order masked circuit in Figure 1 that calculates $ab \oplus c$ in shared form. Up to the beginning of the red circuit parts, the circuit can be expressed by following equations where the parentheses denote the order of the operations as enforced by the registers.

$$\begin{aligned} q_1 &= (a_1 b_1) \oplus (a_1 b_2 \oplus c_1) \\ q_2 &= (a_2 b_2) \oplus (a_2 b_1 \oplus c_2) \end{aligned} \tag{1}$$

This example circuit is first-order probing secure when considering all input variables to be independently shared, as can be verified by using the tool of Bloem et al. [6], for instance. However, the circuit is not NI according to Definition 7 because when considering an attacker that places a probe at the output

q_1 or q_2, the wire cannot be simulated with only one share of each variable since b_1 and b_2 are always both required to calculate the outputs (see Equation 1). Because the circuit does not fulfill the NI notion, it also does not achieve the SNI notion for composability.

Composability. The circuit is so far indeed not composable because even a (by itself harmless) linear combination with one share of c could destroy the probing security. For example, if q_1 would be fed into a circuit that adds c_1 to the signal, the resulting equation would be $(a_1b_1 \oplus a_1b_2)$ which would violate the probing security by bringing both shares of b together. However, by extending the circuit with the red circuitry in Figure 1, a fresh random share (mask) r is added to the output of the circuit. This random share leads to a remasking of the output shares, thus making the output shares statistically independent of the input shares (from a masking perspective) as long as q_0 and q_1 are never combined inside the circuit.

Informally speaking, the shared output of the circuit has the same properties as assumed for the inputs of the circuit, namely the outputs are securely and independently shared from the other inputs as we will demonstrate later in this section. If we again consider a combination of the output of q_1 with c_1 in an adjacent circuit (or any share of the input variables), the security is not violated in this case. The circuit can be fed into any other first-order probing secure circuit that even takes the shares of a, b and c (apart from r, which is assumed to be only used inside the circuit) as input without violating its security. In this sense, the circuit achieves some form of separation of input and output sharings which is, however, strictly weaker and less demanding than the SNI composability notion.

To formally express composability in the probing model, we first formalize what independence of two shared variables in the probing model means which can be easily extended to any number of variables.

Definition 9 (Order-d sharing independence) *Two sharings $\boldsymbol{a}$ and $\boldsymbol{b}$ are order-d independently shared (or d-probing independent) if any arbitrary circuit, taking at most d shares of $\boldsymbol{a}$ and d shares of $\boldsymbol{b}$ as inputs, is first-order ($d = 1$) probing secure.*

As an example how this notion would be checked, consider the two independent sharings of the input variables a $(= \{a_1, a_2\})$ and b $(= \{b_1, b_2\})$. The sharing independence is given by the fact that each possible combination of shares $(\{a_1, a_2, b_1, b_2, a_1b_1, a_1b_2, a_2b_1, a_2b_2\})$ does not contain both shares of a or b. If on the other hand, we consider the sharing independence of a with itself, this would result in $\{a_1, a_2, a_1a_2, a_2a_1\}$, which contains the insecure combinations a_1a_2 and a_2a_1.

Definition 10 (Composable module) *A subcircuit whose output sharings are d-probing independent from the inputs is a composable module.*

Based on the definition of sharing independence, we can define first-order composability as the independence of the input sharings of a module and its output sharings. To break the probing security of the output sharing of a composable module it always requires to bring all output shares together. A combination with any shares from the module's inputs (and for this reason also with any other sharings in the circuit) that by itself is probing secure cannot create a flaw. Since a first-order attacker is bound to one probe and the module itself is secure, it follows that the attacker cannot leverage from the modules internal signals and we can thus treat any first-order secure module with independent sharings as a composable module. Please note, that composability for the higher-order case does not directly follow from a separation of input and output sharings of a module.

As a very simple example for a first-order composable circuit, we consider the circuit depicted in Figure 2 which just takes a shared variable a as input, performs a resharing with r, and ensures that no glitches propagate by using registers. This circuit is securely remasked and can thus be composed with any other probing secure circuit under the assumption that the random variable r is fresh and only used for resharing inside this circuit in Figure 2. As a result, even every combination with the input shares a_1 and a_2 (that when considered on their own are first-order probing secure) at the outputs of the circuit ($a_1 \oplus r$... or $a_2 \oplus r$...) do not create an insecure circuit. For example, adding a_2 to the first output results in the equation $(a_1 \oplus r) \oplus a_2$ which is again first order probing secure because an attacker probing this signals, does not know r.

Connections to SNI. The sharing independence of inputs and outputs can be also formally expressed as a weaker form of SNI where the simulator has no access to any shares of the input variables and the attacker can place up to d probes but only on the output of the circuit.

Definition 11 (Independence of inputs and outputs as variation of SNI) *A module's outputs are independently shared from its inputs iff for* 0 *probes on the module's internal signals, and t_o probes on its outputs with $t_o \leq d$, all probes can be simulated with only* 0 *shares per each shared input.*

This definition is strictly weaker than the SNI notion since it contains only the case where each probe is spent on the outputs. It follows from this definition that each circuit that fulfills the SNI notion also fulfills the independence of input and output sharings which in the first-order case implies composability.

For the higher-order masking, it remains uncertain whether the independence of inputs and outputs is sufficient for higher-order composability in the general case. However, if the inputs of a module are exclusively used by the module that fulfills input and output independence (as it is often the case in nonlinear layers of cryptographic implementations), composability is trivially given by the fact that the input is nowhere else used in the remaining circuit. For this reason, when analyzing the probing security of a round-based cipher, for instance, it is enough to ensure security of one round and the sharing independence on the

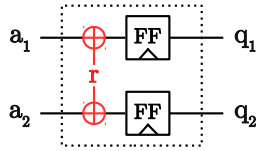

Fig. 2. Example for a very simple first-order secure and composable module

output sharings of the round function. In either case, as we discuss in the next section, the independence of modules allows for faster formal verification of the probing security in the higher-order case.

4 Improvements on the Formal Verification

The sharing independence notion as given in Definition 11 states that a module that fulfills this notion generates only outputs that have the same properties as it is also assumed of any other sharing of two input variables of a circuit. Namely, the module only produces sharings that are independent of all other sharings used in the remaining parts of the circuit by utilizing fresh randomness that is only used once and inside the module. We can thus treat the outputs of this module as if they are freshly shared inputs of the original circuit. Consider as an example the whole circuit from Figure 1. The registers ensure that any circuit connected to this circuit gets following the equation as input.

$$\begin{aligned} q_1 &= ((a_1 b_1) \oplus (a_1 b_2 \oplus c_1) \oplus r) \\ q_2 &= (\underbrace{(a_2 b_2) \oplus (a_2 b_1 \oplus c_2)}_{q = ab \oplus c} \oplus r) \end{aligned} \tag{2}$$

When we assume that r is only used inside this module, then it is quite intuitive that only by bringing the shares of q together this sharing can be demasked which would leak a, b and c. Any other composition with probing secure circuitry, even with one that contains the input sharings used to calculate the shares of q (but not the fresh mask r), is secure. Instead of tracking the whole history of the sharing of q as given by Equation 2 we can simply replace the shares of q by a d-probing secure sharing as in Equation 3 to verify the security of the circuit. We call this process of replacing complex but independent sharings by simpler sharings *relabeling*.

$$\begin{aligned} q_1 &= (q \oplus r) \\ q_2 &= r \end{aligned} \tag{3}$$

Figure 3 shows how we modify an original circuit in which one of the modules produces independently shared outputs to speed up the verification by relabeling. We thus prune the output wires of the module $M1$, create a new input port

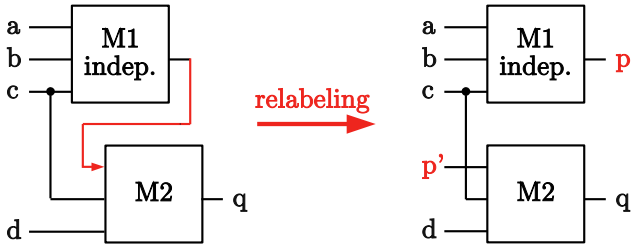

Fig. 3. Example for relabeling of independently shared outputs

(p') and a new output port (p) for the top-level circuit and reconnect the fan-out of the module to the newly created input port. The complexity of the circuit, and therefore also the complexity of its formal verification, is reduced because the logic depth of the overall circuit is lowered. It is important to denote that our composability notion does not allow to also cut the input wires of the composable module for the higher-order case ($d > 1$) as in SNI, and thus complete separation of the module from the remaining circuit. In the following, we prove the correctness of our approach.

4.1 Proofs for Relabeling

If not stated, we assume that all the functions are functions from $2^X \to \mathbb{B}$. We denote by $|f|$ the number of minterms of f.

Lemma 12 $|f| = (1 - \hat{f}(\emptyset)) \cdot 2^{|X|-1}$.

Definition 13 (Function independence) *f is independent of g if $|f \wedge g|/|g| = |f \wedge \neg g|/|\neg g|$.*

The intuitive meaning of the Definition 13 is that f gives no information about g. If g is a secret then knowing f does not produce a security flaw. It is easy to see that independence is symmetric.

Lemma 14 *f is independent of g iff $|f \wedge g| \cdot |\neg f \wedge \neg g| = |\neg f \wedge g| \cdot |f \wedge \neg g|$ iff g is independent of f.*

Definition 15 (Independent function set) *A set of functions $\{f_1, ..., f_k\}$ is independent if $\forall i \forall g : \mathbb{B}^{k-1} \to \mathbb{B}$, we have that f_i is independent of $g(f_1, ..., f_{i-1}, f_{i+1}, ..., f_k)$.*

Theorem 16 (Independent functions) *Let f and g be functions such that there is no $\emptyset \neq t \subseteq X$ such that $\hat{f}(t) \neq 0$ and $\hat{g}(t) \neq 0$. Then f and g are independent.*

Proof. Because the Fourier expansions of f and g do not have a term $\emptyset \neq t \subseteq X$ such that $\hat{f}(t) \neq 0$ and $\hat{g}(t) \neq 0$, we have that $\widehat{(f \wedge g)}(\emptyset)$ depends only on $\hat{f}(\emptyset)$ and $\hat{g}(\emptyset)$, and likewise for $\widehat{(f \wedge \neg g)}(\emptyset)$, $\widehat{(\neg f \wedge g)}(\emptyset)$, and $\widehat{(\neg f \wedge \neg g)}(\emptyset)$.

Using the Fourier expansion of conjunction and straightforward arithmetic, we can show that $|f \wedge g| \cdot |\neg f \wedge \neg g| = (1 - \widehat{f \wedge g}(\emptyset)) \cdot 2^{|X|-1} \cdot (1 - \widehat{f \wedge \neg g}(\emptyset)) \cdot 2^{|X|-1} = (1 - \widehat{\neg f \wedge g}(\emptyset)) \cdot 2^{|X|-1} \cdot (1 - \widehat{\neg f \wedge \neg g}(\emptyset)) \cdot 2^{|X|-1} = |\neg f \wedge g| \cdot |f \wedge \neg g|$. □

We identify a sharing T with the set of functions given by the set of shares in T.

Definition 17 (Cut) *Given a circuit C and a set of sharings $\mathcal{T} = \{T_1, ..., T_n\}$ where each sharing in $\mathcal{T}$ is given as a set of gates in the circuit, we say that $\mathcal{T}$ is a cut of C if for any sharing $T_i \in \mathcal{T}$ and any shares $s_j^1, ..., s_j^d \in T_j$ the set of functions $\{val(T_j)\} \cup \bigcup_{j \in [1,n]} \{s_j^1, ..., s_j^d\}$ is independent.*

Theorem 18 *Let C be a circuit. The set of primary input sharings and output sharings of composable modules is cut in C.*

Proof. The correctness of the theorem follows by the facts that initial sharings are independent, and the output sharing of a composable module can only depend on its inputs which is covered by the Definition 10. □

Given a circuit C and its cut $\mathcal{T}$, one can get the circuit equivalent C' where any sharing $T \in \mathcal{T}$ can be securely relabeled. That is, outputs of a composable module are cut and become circuit outputs, and new primary inputs are created and connected to the fan-out of the module as a sharing of a new variable. The following theorem gives the generalization of the above claim.

Theorem 19 (Relabeling) *Let C is a circuit, $\mathcal{T}$ be a cut in C, and C' is the equivalent of C that is cut along $\mathcal{T}$. Then C is d-probing secure if C' is d-probing secure.*

Proof. Suppose $P' = \{p_1, ..., p_d\}$ be a set of probes in C', and P' is secure. Let P be the corresponding probes in C. If P' is secure, it cannot depend on any sharing value in $\mathcal{T}$. Thus, it depends on at most d shares for any sharing in $\mathcal{T}$. Therefore, P depends on at most d shares of any sharing in $\mathcal{T}$, and by definition of cut, the function defined by $\mathcal{T}$ is independent of any sharing in C. Then, P is secure.

Given C' is secure, all possible tuples P' of up to d probes in C' are secure, so as well as P in C. Therefore, C is d-probing secure. □

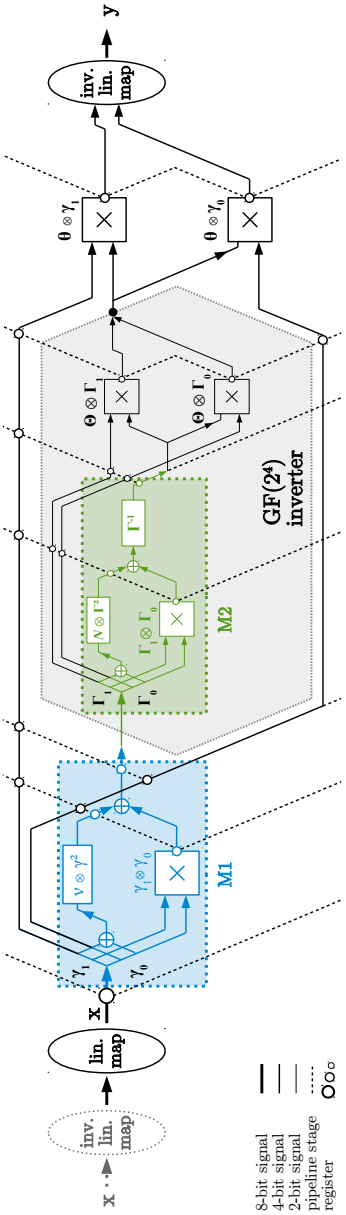

Fig. 4. DOM AES S-box with the two relabeled modules M1 and M2 as used for verification

5 Practical Experiments and Comparisons

To demonstrate the efficiency increase in the formal verification when using our independence notion, we decide to use the generic DOM AES S-box [13] depicted in Figure 4 as an example. This S-box is based on the original design of Canright [7] and was used in many masked AES implementations [4, 5, 8, 9, 11, 13, 17]. Furthermore, the AES S-box has proven to be a suitable and complex benchmark for masking schemes as well as for verification tools [1, 6].

Before the verification approaches of Bloem et al. and Barthe et al. can benefit from our approach, we first need to find suitable modules that fulfill the independence of inputs and outputs. For this purpose, we wrote a pass for the Yosys Open Synthesis Suite [22, 23] that automatically detects and extracts suitable modules. The pass searches for entry points of fresh random masks followed by two register stages (such that the compression to $d + 1$ masks is fulfilled) and tracks the signal flow forth and back to extract the modules. We then use the SAT based checking from Appendix A to verify the independence of the outputs and cut the output ports of the modules. The outputs then become outputs of the top-level module, and new inputs are created that are reconnected to the fan-out of the independent module. This way we found and extracted two suitable modules for relabeling: M1) the circuitry around the first $GF(2^4)$ multiplier, and M2) the first $GF(2^2)$ multiplier and registers inside the $GF(2^4)$ inverter.

Further optimization. To speed up the verification even further we used in addition a trick borrowed from cryptanalysis. Since, e.g., the approach of Bloem et al. uses just an approximation of the Fourier spectrum which becomes already quite complicated after the initial linear transformation of the S-box, we fed the inverse linear transformation of the linear mapping into the S-box. We are allowed to do this since our proofs work for any independent sharing of the inputs which is given because the mapping is linearly independent since it is invertible, and the mapping is applied share-wise to the input sharings. The advantage we get from this, however, is a cleaner representation of the shared input bits after the linear mapping which we can then feed to the tools. We note that we do not cut the linear mapping away from the S-box but rather connect the fan-out of the linear transformation to the signals before the inverse linear transformation.

Results. Table 1 shows the verification results for the relabeled AES S-box up to order five as well as results for the unmodified circuit and from related work. In addition, Table 2 states some statistics on the size and complexity of the relabeled circuits, such as the number of linear and nonlinear gates, and how many secrets and masks were checked. Since our circuit transformations for relabeling introduce new secret and masks, the additional number of secrets and masks are denoted in parentheses for the relabeled circuits. Time spent on checking module independence is negligible compared to the actual verification time and not stated in the results. With relabeling, the verification times for the tool of Bloem et al. (Rebecca) and the tool of Barthe et al. (maskVerif) are

Table 1. Verification results for the DOM AES S-box [13]

Order	*Rebecca* [15]		*maskVerif* [2]	
	w/o relabeling	w/ relabeling	w/o relabeling	w/ relabeling
1	10 h [6]	10 min	3 s [1]	$\leq$ 0.1 s
2	-	-	-	3 s
3	-	-	-	4 m
4	-	-	-	9 h
5	-	-	-	46.6 d

lowered from 10 hours to 10 minutes and from 3 seconds to less than 100 ms for the first-order circuits, respectively. Since there have not been any higher-order results reported in existing works so far, we tried to verify the second-order S-box circuits without relabeling but canceled the experiments after more than one month. With relabeling we can verify the second-order S-box in just about 3 seconds for the maskVerif tool.

Figure 5 illustrates the verification times for different protection orders including results up to the fifth protection order which takes about 46.6 days to be completed.

6 Conclusions and Discussion

In this work, we targeted the more efficient formal verification of circuits. As a first step, we discussed the gap between composability in the more general probing model with glitches and the strong noninterference (SNI) notion. We then introduced a sharing independence notion and demonstrated that this implies composability for first-order probing secure circuits which outputs and inputs are independently shared. Furthermore, we discussed that even for higher-order masking composability results from independence input and output sharings if there exists no feed-forward path for the inputs. In either case, we demonstrated that sharing independence helps to speed up the formal verification in existing tools from Bloem et al. [6] and Barthe et al. [1] by allowing us to relabel independently shared module outputs. We could thus reduce the verification time for a

Table 2. Statistics for the relabeled circuits, numbers in parentheses indicate additional variables induced by cutting of signals

Order	*Gates*			*Variables*	
	linear	nonlinear	reg	secret	mask
1	460	144	278	8 (6)	26 (6)
2	978	324	471	8 (6)	54 (12)
3	1,688	576	700	8 (6)	108 (18)
4	2,590	900	965	8 (6)	180 (24)

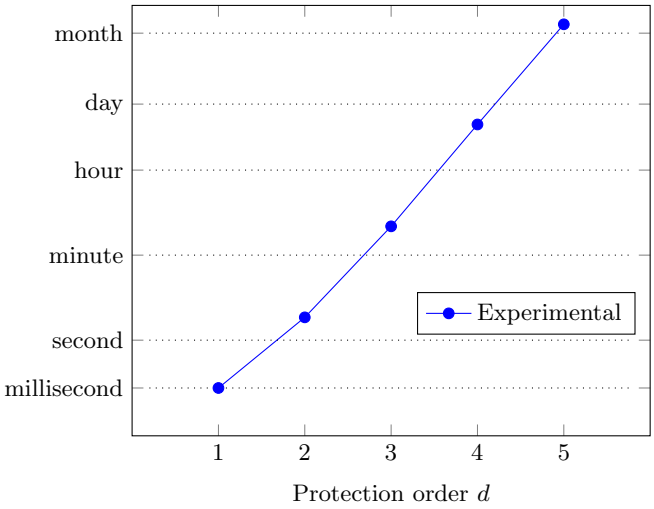

Fig. 5. Protection order versus verification time of DOM AES S-box

first-order DOM masked AES S-box from 10 hours to 8 minutes and from 3 seconds to less than 0.1 seconds respectively. We also presented the first verification results for the higher-order DOM masked AES S-box up to order 5. With these results, we are now for the first time able to prove large circuits for higher-order security directly on the netlist of a masked hardware implementation. Also, we now have formal justification for round-separated analysis of side-channel information of cryptographic implementations which round functions fulfill input and output sharing independence.

Future work. Our work opens the door to some new and interesting research questions.

- Even though relabeling results in a significant speed-up for the higher-order verification, it remains uncertain whether or not composability follows from the sharing independence of input and outputs in the general case. This would allow for separated verification of modules and would speed-up the verification even more.
- For the search for suitable candidates for independent modules, we focused on finding patterns that consist of a combination of an XOR gate with fresh randomness followed by two register stages before the next nonlinear transformation. There exists another thinkable relabeling point that is located inside the last multiplier stage in Figure 4. We did not consider this as an independent module, since using it would mean that we need to change more in the circuitry than just connecting and disconnecting of wires. Nevertheless, this could be another way to speed up the verification if one finds the correct formal justification to do so.

- Existing verification tools are not able to handle feedback loops and do not consider the actual control flow in their verification. Especially first-order circuits for which there exist masking solutions which reuse randomness from different clock cycles or even switch between the source of randomness requires manual treatment. One possible solution is to unroll feedback loops up to a certain clock cycle to analyze the security within a limited number of iterations.
- Another open question is the assessment of the amount of leakage for a given flaw once it is detected. At the moment the solutions of Bloem et al. and Barthe et al. do not support analysis on the severity for a given flaw. The approach of Bloem et al. could be theoretically extended towards the precise calculation of the Fourier spectrum for a given flaw which could then serve as a basis for the quantification of the leakage. However, such calculations could soon become unfeasible due to their high complexity.

Acknowledgements.

We want to thank Sonia Belaïd, Gilles Barthe, Pierre-Alain Fouque, and Benjamin Grégoire for the helpful discussions and support with their *maskVerif* tool [2]. This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 681402). This work has been supported by the Austrian Research Promotion Agency (FFG) via the project IoT4CPS. Furthermore, the work has been supported in part by the Austrian Science Fund (FWF) through project S11406-N23 and project W1255-N23.

References

1. G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, P. Strub, and R. Zucchini. Strong non-interference and type-directed higher-order masking. In *ACM Conference on Computer and Communications Security*, pages 116–129. ACM, 2016.
2. G. Barthe, S. Belaïd, P. Fouque, and B. Grégoire. maskverif: a formal tool for analyzing software and hardware masked implementations. *IACR Cryptology ePrint Archive*, 2018:562, 2018.
3. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. Higher-order threshold implementations. In *ASIACRYPT (2)*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.
4. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. A more efficient AES threshold implementation. In *AFRICACRYPT*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2014.
5. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. Trade-offs for threshold implementations illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.
6. R. Bloem, H. Groß, R. Iusupov, B. Könighofer, S. Mangard, and J. Winter. Formal verification of masked hardware implementations in the presence of glitches. In

EUROCRYPT (2), volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.
7. D. Canright. A very compact s-box for AES. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.
8. T. D. Cnudde, B. Bilgin, O. Reparaz, V. Nikov, and S. Nikova. Higher-order threshold implementation of the AES s-box. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015.
9. T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen. Masking AES with d+1 shares in hardware. In *CHES*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016.
10. S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert. Composable masking schemes in the presence of physical defaults and the robust probing model. *IACR Cryptology ePrint Archive*, 2017:711, 2017.
11. H. Groß, R. Iusupov, and R. Bloem. Generic low-latency masking in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):1–21, 2018.
12. H. Groß and S. Mangard. Reconciling d+1 masking in hardware and software. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 115–136. Springer, 2017.
13. H. Groß, S. Mangard, and T. Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *IACR Cryptology ePrint Archive*, 2016:486, 2016.
14. Y. Ishai, A. Sahai, and D. A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
15. R. Iusupov. REBECCA - Masking verification tool. https://github.com/riusupov/rebecca.
16. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
17. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the limits: A very compact and a threshold implementation of AES. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.
18. S. Nikova, C. Rechberger, and V. Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.
19. R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.
20. J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
21. O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede. Consolidating masking schemes. In *CRYPTO (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.
22. C. Wolf. Yosys Open SYnthesis Suite. http://www.clifford.at/yosys/.
23. C. Wolf and J. Glaser. Yosys - a free verilog synthesis suite. In *Proceedings of Austrochip 2013*, 2013.
24. G. Xiao and J. L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Trans. Information Theory*, 34(3):569–571, 1988.

A Independence checking

We introduce the the method to check order-d sharing independence according to the Definition 9. We encoded a problem into formulas in propositional logic and use SAT solving as a resolution procedure. Intuitively, we encode the semantics of the module in a way described in [6], and connect an XOR gate (checking gate) to any possible subset of up to d signals from each sharing. Then, if the output of the checking gate is 1-order secure, the sharings are d-probing independent.

Let C be a circuit. For each gate g we introduce a set of Boolean variables $X_g = \{x_g \mid x \in X\}$, additionally for each gate in input and output sharings we create a Boolean *activation variable* a_g. For a checking gate g_c we introduce a set of Boolean variables X_{g_c}. We define a formula Ψ to check whether the sharings are d-probing independent. L and N are the sets of linear gates and nonlinear gates, resp. Formula Ψ consist of multiple parts:

$$\Psi = \bigwedge_{g \in I} \Psi_{\text{inp}}(g) \wedge \bigwedge_{g \in N} \Psi_{\text{nl}}(g) \wedge \bigwedge_{g \in L} \Psi_{\text{lin}}(g) \wedge \bigwedge_{g \in R} \Psi_{\text{reg}}(g).$$

The labeling of the inputs is determined by $\mathcal{I}$. For $X' \subseteq X$, we define

$$\psi_g(X') = \bigwedge_{x \in X} \begin{cases} x_g(X') & \text{if } x \in X', \\ \neg x_g(X') & \text{if } x \notin X', \text{ and} \end{cases}$$
$$\Psi_{\text{inp}}(g) = \bigvee_{X' \subseteq X : \widehat{\mathcal{I}(g)}(X') \neq 0} \psi_g(X').$$

To define the behavior of linear and nonlinear gates we define the following auxiliary formulas, where $T = (t_1, \ldots, t_n)$, $U = (u_1, \ldots, u_n)$, and $V = (v_1, \ldots, v_n)$ are ordered sets of variables, and define $\leftrightarrow$ to denote equality.

$$\Psi_{\text{empty}}(T) = \bigwedge_i \neg t_i,$$
$$\Psi_{\text{copy}}(T, U) = \bigwedge_i (t_i \leftrightarrow u_i), \text{ and}$$
$$\Psi_{\text{lin}}(T, U, V) = \bigwedge_i (t_i \leftrightarrow (u_i \oplus v_i)).$$

For a linear gate g with inputs g' and g'', we use the formula

$$\Psi_{\text{lin}}(g) = \Psi_{\text{lin}}(X_g, X_{g'}, X_{g''}),$$

for a nonlinear gate g with inputs g' and g'', we use the formula

$$\Psi_{\text{nl}}(g) = \Psi_{\text{empty}}(X_g) \vee \Psi_{\text{copy}}(X_g, X_{g'}) \vee \Psi_{\text{copy}}(X_g, X_{g''}) \vee \Psi_{\text{lin}}(X_g, X_{g'}, X_{g''}), \text{ and}$$

for a register g withs input g', we simply have $\Psi_{\text{reg}}(g) = \Psi_{\text{copy}}(X_g, X'_g)$.

Also we introduce an integer variables a_{s_sum} for each sharing, and bound it to the attack order d:

$$a_{s\_sum} = \sum_{g_s} \mathrm{Ite}(a_g, 1, 0)$$
$$a_{s\_sum} \leq d.$$

The function $Ite(a_g, 1, 0)$ (if-then-else) converts a Boolean variable to Integer.

For the checking gate we XOR the corresponding inputs:

$$\Psi(g_c) = \bigwedge_{x \in X} x_{g_c} \leftrightarrow \bigoplus_{g \in \mathcal{G}} a_g \wedge x_g.$$

For the transient signals we introduce a second set of variables $X'_g = \{x'_g \mid x \in X\}$. We create a slightly modified set of constraints, where we write Φ' to denote a formula Φ in which each variable x_g has been replaced by x'_g.

$$\Phi = \Phi_{\mathrm{gates}} \wedge \Phi'_{\mathrm{unsafe}}, \text{ where}$$
$$\Phi_{\mathrm{gates}} = \bigwedge_{g \in I} (\Psi_{\mathrm{inp}}(g) \wedge \Psi'_{\mathrm{inp}}(g)) \wedge \bigwedge_{g \in N} (\Psi_{\mathrm{nl}}(g) \wedge \Psi'_{\mathrm{nl}}(g)) \wedge$$
$$\bigwedge_{g \in L} (\Psi_{\mathrm{lin}}(g) \wedge \Psi'_{\mathrm{nl}}(g)) \wedge \bigwedge_{g \in R} \Phi_{\mathrm{reg}}(g),$$

where for a register g with input g', we copy only the original (glitch-free) signals:

$$\Phi_{\mathrm{reg}}(g) = \Psi_{\mathrm{copy}}(X_g, X_{g'}) \wedge \Psi_{\mathrm{copy}}(X_g, X'_{g'}).$$

Finally, we check whether security is violated, that is, whether there is a non-zero Fourier coefficient which contains a secret and no masks:

$$\Phi'_{\mathrm{unsafe}}(g_c) = \bigvee_{s \in S} s'_{g_c} \wedge \bigwedge_{m \in M} \neg m'_{g_c}.$$