IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 3, pp. 114–140. DOI:10.46586/tches.v2022.i3.114-140

Composable Gadgets with Reused Fresh Masks

First-Order Probing-Secure Hardware Circuits with only 6 Fresh Masks

David Knichel[1] and Amir Moradi[2]

[1] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
firstname.lastname@rub.de
[2] University of Cologne, Institute for Computer Science, Germany
firstname.lastname@uni-koeln.de

Abstract. Albeit its many benefits, masking cryptographic hardware designs has proven to be a non-trivial and error-prone task, even for experienced engineers. Masked variants of atomic logic gates, like AND or XOR – commonly referred to as gadgets – aim to facilitate the process of masking large circuits by offering free composition while sustaining the overall design's security in the d-probing adversary model. A wide variety of research has already been conducted to (i) find formal properties a gadget must fulfill to guarantee composability and (ii) construct gadgets that fulfill these properties, while minimizing overhead requirements. In all existing composition frameworks like NI/SNI/PINI and all corresponding gadget realizations, the security argument relies on the fact that each gadget requires individual fresh randomness. Naturally, this approach leads to very high randomness requirements of the resulting composed circuit. In this work, we present *composable gadgets with reused fresh masks* (COMAR), allowing the composition of any first-order secure hardware circuit utilizing only 6 fresh masks in total. By construction, our newly presented gadgets render individual fresh randomness unnecessary, while retaining free composition and first-order security in the robust probing model. More precisely, we give an instantiation of gadgets realizing arbitrary XOR and AND gates with an arbitrary number of inputs which can be trivially extended to all basic logic gates. With these, we break the linear dependency between the number of (non-linear) gates in a circuit and the randomness requirements, hence offering the designers the possibility to highly optimize a masked circuit's randomness requirements while keeping error susceptibility to a minimum.

Keywords: Side-Channel Analysis · Masking · Probing Security · Composability · COMAR

1 Introduction

Masking. The increasing amount of easily accessible devices creates a permanent surge in demand for highly effective countermeasures against physical attacks, causing Side-Channel Analysis (SCA) attacks to retain their topical relevance since its first seminal description in [Koc96, KJJ99]. Since then, a considerable amount of effort has been put into mitigating the threat originating from a wide variety of divers side channels, ranging from timing [Koc96], power consumption [KJJ99], electromagnetic (EM) emanations [GMO01], or temperature and heat dissipation [HS13]. Among a multitude of proposed methods, *masking* has evolved to be a promising technique to achieve SCA resilience due to is solid theoretical background rooted in the concepts of secret sharing. Despite the strong effort committed to masking cryptographic primitives [ISW03, Tri03, NRS11, RBN+15, GMK17, GM18], many of the proposed schemes have proven to suffer from invalid assumption or design flaws [MMSS19], eventually compromising their practical security.

Received: 2022-01-15 Accepted: 2022-03-15 Published: 2022-06-08

Formal Adversary Models. In order to prevent these bugs, researchers have started to lay focus on establishing formal adversary models which aim to accurately capturing the adversary's capabilities and the circuit behavior in the real world, while being sufficiently abstract to enable (automated) evaluation of SCA resilience on an algorithmic level [ISW03, BDF$^+$17, PR13, FGP$^+$18]. As a consequence, these models, on the one hand, enable detecting security flaws in an early stage of the design process. For this, a wide variety of tools have been proposed [BGI$^+$18, BBC$^+$19, KSM20, BDM$^+$20, BBD$^+$16, BGR18, BBD$^+$15, CGLS21]. On the other hand, they allow following a more systematic approach when constructing provable-secure masking schemes. The ISW d-probing model – initially introduced in [ISW03] – and its extension to model physical defaults contributing to data leakage in hardware implementations [FGP$^+$18], has turned out to be an adversary model well suited for finding implementations that are provably resilient against SCA attacks due to both, its sufficiently high abstraction and the existing reduction into the *Noisy Leakage Model* [DDF14], which is considered to be the closest to reality with respect to accurately modeling leakage behavior.

Composable Gadgets. As it has proven to be a hard task to mask circuits realizing large functions for high security orders, a new line of research has emerged aiming to formally define composability notions which enable the construction of masked circuits in a divide-and-conquer fashion. Here, the goal is to derive masked realizations of atomic logic gates – typically AND and XOR – that fulfill the properties specified by the composability notion, and in so doing, guaranteeing security in the d-probing model, even when interconnected into a larger circuit. Hence, this reduces the task of finding (higher-order) masked versions of large circuits to constructing masked, atomic logic gates that are in conformity with the security notions and then modularly constructing the overall circuit based on these masked gates – commonly referred to as *gadgets*. In the context of the d-probing model, Non-Interference (NI)/Strong Non-Interference (SNI) [BBD$^+$15, BBD$^+$16] and Probe-Isolating Non-Interference (PINI) [CS20] were introduced as such notions, where SNI ironed out flaws of NI whose properties were not sufficient to guarantee composability. As the scope of SNI was originally limited to single-output gates, Cassiers et al. generalized it in [CS20], but at the same time introduced PINI as an elegant notion to construct composable gadgets which further reduces implementation overheads and allows trivial realization, i.e., share-wise application, of linear functions. Accompanied by the introduction of this wide variety of notions, many concrete gadget constructions have been proposed, either realizing small logic gates [ISW03, BDM$^+$20, BBD$^+$16, CS20, CGLS21, CS21], or even arbitrary logic functions [KSM22].

Randomness Reduction. As gadgets typically realize atomic logic functions like a simple 2-input AND gate, and fresh randomness is required for each of these gadgets, the composition to entire cipher designs results in a significant randomness overhead being introduced to the circuit. As fresh randomness needs to be provided by a source, i.e., Pseudo-Random Number Generators (PRNGs), randomness requirements directly translate into area consumption on a chip and hence into an increase in costs of production. Consequently, it is an interesting research topic to elaborate on how randomness overhead of composable gadgets can be further reduced. One possible solution is to extend the functionalities of the gadgets while preserving individual randomness requirements. In [KSM22], a methodology was presented that allows the construction of first-order-secure gadgets realizing arbitrary Boolean functions with only one random bit per output. Although this certainly leads to randomness reductions compared to the utilization of 2-input gadgets when constructing large masked circuits, it also significantly increases the area requirements of the implementation.

Another approach is to minimize the randomness requirements per gadget. In [GSM$^+$19], Gross et al. presented a methodology to derive a first-order-secure implementation from AND and XOR gadgets requiring two fresh random bits in total – including the initial sharing. This approach is restricted to software implementations, i.e., does not guarantee SCA resilience in the presence of physical defaults [WM18]. Another drawback is that interconnection of gadgets is not trivial, i.e., the designer has to follow a certain set of rules when composing the gadgets.

Our Contributions. In this work, we present COMAR, a methodology for achieving free composition of hardware gadgets that enable construction of arbitrary first-order-secure hardware implementations in the d-probing model under glitches, utilizing – apart from the initial sharing – only 6 fresh random bits in total. In this context, we construct freely-composable gadgets, realizing 2-input XOR and AND gates which can be trivially converted to all atomic logic gates. We further show how to extend such gadgets to cover an arbitrary number of inputs at the cost of investing more fresh masks which can still be reused by other gadgets. We demonstrate that our constructions lead to a significant reduction with respect to randomness overhead introduced into the design compared to existing schemes, offering the same composability and security guarantees. Our work hence paves the ground for enabling the designer to optimize for different overhead parameters when masking a cryptographic implementation, while keeping design constraints and error susceptibility as low as possible. We further practically verify our findings by means of case studies and leakage assessments.

Outline. We first present a brief recap of all necessary concepts and methodologies in Section 2. In particular, we give an introduction to the robust d-probing model and all well-established composability notions. In Section 3, we present our novel methodology for constructing gadgets with constant randomness requirements, and argue about their first-order security and composability, before we discuss the practical implications of our work by means of different case studies in Section 4. After presenting a dedicated comparison to the state of the art in Section 5 and conducting an exemplary leakage assessment in Section 6, we conclude our work in Section 7.

2 Preliminaries

2.1 Notations

While we denote random variables by capital letters, e.g., $X \in \mathbb{F}_2$ is a binary random variable, we use bold capital letters, e.g., $\mathbf{X}$, for sets containing random variables. The i-th input to a function is identified by X_i, while superscripts are used to denote share indices when dealing with masked functions. Consequently, X^i is the i-th share of X. Moreover, $\mathbf{X}_i$ denotes the set of all elements in $\mathbf{X}$ with subscript i. When masking a function $\mathsf{F} : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ with t shares per input, the set containing all input shares is given as $Sh(\mathbf{X}) = [X_0^0, X_0^1, \dots, X_0^{t-1}, X_1^0, \dots, X_{n-1}^{t-1}]$. In the same manner, for a set of share indices $\mathbf{I} \subseteq [0, \dots, t-1]$, $Sh(\mathbf{X})^{\mathbf{I}}$ denotes the set of all input shares $X_{0 \leq i < n}^{s \in \mathbf{I}}$. Eventually, drawing a value X uniformly and at random from a set $\mathcal{R}$ is denoted as $X \overset{\$}{\leftarrow} \mathcal{R}$.

2.2 Boolean Masking

Rooted in the concept of secret sharing, a *Boolean Masking* of a secret $X \in \mathbb{F}_2^n$ is a set $\mathbf{X} \in \mathbb{F}_2^{n \times s}$ of s independent secret shares $X^i \in \mathbb{F}_2^n$, $0 \leq i < s$, such that $X = \bigoplus_{i=0}^{s-1} X^i$. The secret shares are commonly derived by sampling $X^i \overset{\$}{\leftarrow} \mathbb{F}_2^n$, for $0 \leq i < s-1$ and deriving the remaining share as $X^{s-1} = X \oplus \left(\bigoplus_{i=0}^{s-2} X^i \right)$. This directly implies that, in order to achieve d-th order security, all sensitive data has to be split in at least $d+1$ shares, i.e., two shares in the case of first-order security.

2.3 Circuit Model

As derived in [ISW03] and later in its extension [FGP⁺18], any stateful, deterministic circuit C can be modeled as a Directed Acyclic Graph (DAG) $\mathsf{G_C} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V}$ being the set of vertices and $\mathcal{E}$ the set of edges of $\mathsf{G_C}$. The edges represent wires carrying elements of $\mathbb{F}_2$ while the vertices model combinational gates such as AND and XOR or memory gates, i.e., registers. On any circuit invocation, registers output the previous input to the gate while storing the current input for the next invocation. Eventually, $\mathsf{G_C}$ realizes a Boolean function $\mathsf{F} : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$.

Encoded Circuit Model. To precisely define the adversary's capabilities when probing a masked circuit, a more fine-grained breakdown of the circuit becomes necessary. As defined in [AIS18], a *circuit compiler* consists of three algorithms. The COMPILE algorithm is deterministic and takes a circuit C as input and outputs a randomized (masked) circuit $\tilde{\mathsf{C}}$. The probabilistic ENCODE algorithm takes as input $\mathbf{X}$ and outputs the encoded input $\tilde{\mathbf{X}}$. In our case, this corresponds to performing Boolean masking as described in Section 2.2. DECODE is a deterministic algorithm, eventually taking encoded data $\tilde{\mathbf{Y}}$ and outputting $\mathbf{Y}$. In the case of Boolean masking, this can be simply achieved by building the XOR-sum of all shares, i.e., unsharing.

Given these three algorithms, data processing in a shared manner can be described by computing $\mathbf{Y} \leftarrow \mathsf{DECODE} \circ \tilde{\mathsf{C}} \circ \mathsf{ENCODE}(\mathbf{X})$ for $\mathbf{Y} \leftarrow \mathsf{C}(\mathbf{X})$. It is important to highlight that the adversary's probing capabilities are limited to only collect observations in $\tilde{\mathsf{C}}$, while the actual execution of the ENCODE and DECODE operations stay hidden. In other words, these operation are known to the adversary, i.e., no obscurity, but they cannot be probed.

2.4 Probing Security

2.4.1 d-Probing Model

The traditional ISW d-probing model [ISW03] grants the attacker the ability to probe up to d wires in the circuit and hence to observe the values carried by them. Modeling the circuit behavior relies on the assumption that at any given time, each wire carries a stable signal defined by the combinatorical function contributing to it. As a result, a circuit is considered secure in the d-probing model, if, and only if, observing the joint distribution of up to d probes reveals no information about any sensitive data.

2.4.2 Robust Probing Model

The traditional ISW d-probing model is mainly relevant for verifying the security of software implementations, as the assumption of wires carrying stable signals has proven to be false for hardware implementations due to physical defaults additionally contributing to the circuit's information leakage. In [FGP⁺18], Faust et al. introduced the *robust* d-probing model, broadening the scope of the d-probing model to accurately model (i) glitches, i.e., differences in signal delays, possibly causing unintentional share recombinations, (ii) data transitions occurring at registers, and (iii) coupling, i.e., dependencies between adjacent wires in a circuit.

Glitches are captured by the robust probing model in a worst-case manner, meaning that a probe is not restricted to observe the stable, and functionally intended, value of a wire, but is also able to observe any value contributing to its computation up to the last

Algorithm 1 glitch-extend

Input: Non-extended probe P

Output: Glitch-extended probes $\mathbf{P}^{ext}$

```
1: if $P$ is placed on an output of a combinational gate then
2:     $\mathbf{P}^{ext} \leftarrow \bigcup_{0 \le i < n} \textsf{glitch-extend}(P_i)$    ▷ where $\{P_0, \ldots, P_{n-1}\}$ are all inputs of the driving gate
3: else if $P$ is placed on an output of a register or on a primary input then
4:     $\mathbf{P}^{ext} \leftarrow \{P\}$
5: end if
```

register stage or primary input. This is formalized in Definition 1, where a standard probe on a wire is recursively extended by either extending all input probes of the driving gates or by returning the standard probe itself if it probes a register output or a primary input. For a set of probes, the corresponding extended set of probes is built straightforwardly by the union of all individual extended probes.

Definition 1 (Glitch-Extended Probes)**.** A set of standard probes $\mathbf{P} = \{P_0, P_1, ..., P_{d-1}\}$ is glitch-extended through $\mathbf{P}^{ext} = \bigcup_{0 \le i < d} \textsf{glitch-extend}(P_i)$, where the procedure of glitch-extend is given in Algorithm 1 and $\mathbf{P}^{ext}$ is called the set of glitch-extended probes with respect to $\mathbf{P}$.

2.5 Composability Notions

Previous efforts have shown that direct masking of large and complex functions is a hard task, especially for higher orders. This is why a new stream of research has emerged, dealing with developing composability notions which aim at defining sufficient properties a masked sub-circuit – in this context, commonly referred to as *gadgets* – must fulfill in order to guarantee security in the (robust) d-probing model of a larger, composed circuit. All these notions focus on limiting the *propagation* of probes which intuitively describes, how leakage is traversed backwards throughout a circuit [CS20], originating from the position where the probe is initially placed and which is directly connected to the concept of probe simulatability [BBP+16]. Probes placed on a gadget are restricted to propagate only into a limited set of input shares, allowing to draw conclusions about probe propagation into primary inputs of the larger, composed circuit.

2.6 Probe Simulatability

Probe simulatability helps to formalize probe propagation, i.e., dependencies between probes placed on the encoded circuit and input shares. It can be formally defined as follows.

Definition 2 (Perfect Probe Simulation)**.** Given a set of (extended) probes $\mathbf{P}$ with cardinality $|\mathbf{P}| = t$ placed on a masked circuit $\tilde{\mathsf{C}}$, $\mathbf{P}$ is said to be *perfectly simulatable* by a set of input shares $\mathbf{S}$ iff there exist a simulator SIM, such that for any value for the inputs of $\tilde{\mathsf{C}}$, the joint probability distribution over $\mathbf{P}$ and $\mathsf{SIM}(\mathbf{S})$ are equal, where $\mathsf{SIM}(\mathbf{S}) : \mathbb{F}_2^{|\mathbf{S}|} \mapsto \mathbb{F}_2^t$ with input $\mathbf{S} \subseteq Sh(\mathbf{X})$ is a probabilistic polynomial time (p.p.t.) simulator.

With the help of this formal definition, we can give a definition for the common composability notions in the following.

2.6.1 Non-Interference/Strong Non-Interference

As introduced in [BBD+15], NI does not differentiate between internal and output probes and limits probe propagation likewise for both. Due to the lack of any distinction between internal and output probes placed on a masked circuit, NI has proven to be non-sufficient to guarantee composability. As a remedy, its definition was adjusted by Barthe et al. in [BBD+16], resulting in the notion of SNI where probes placed on the output of a gadget are not allowed to propagate into any input share at all.

2.6.2 Probe-Isolating Non-Interference

As the definition of SNI only covers single-output gadgets in its original form, Cassiers et al. extended it in [CS20] to be applicable to multiple-output gadgets as well. At the same time, they introduced the notion of PINI as an alternative to guarantee composability. Its advantages stem from its reduced overhead and enabling trivial construction, i.e., share-wise application, of masked linear functions without the need for any fresh randomness and without introducing any additional latency into the design.

Borrowed from Domain-Oriented Masking (DOM) [GMK16], share domains where introduced and probe propagation of output probes was restricted to only occur within the same share domain, while internal probes are limited to propagate within a single, but arbitrary, share domain.

Definition 3 (d-Probe-Isolating Non-Interference (PINI))**.** Let $\mathbf{P_I}$ be the set of internal probes with $|\mathbf{P_I}| = t_1$. Further, let $\mathbf{I_O}$ be the index set (share domains) assigned to the output wires probed by $\mathbf{P_O}$ with $|\mathbf{I_O}| = t_2$ and $\mathbf{P_O}$ containing all output probes.

A masked circuit $\tilde{\mathsf{C}}$ provides d-Probe-Isolating Non-Interference iff for every set of probes $\mathbf{P} = \mathbf{P_I} \cup \mathbf{P_O}$ with $t_1 + t_2 \leq d$, there exists a set $\mathbf{I_I}$ of circuit input share indices with $|\mathbf{I_I}| \leq t_1$ such that $\mathbf{P}$ can be perfectly simulated by $\mathbf{S} = Sh(\mathbf{X})^{\mathbf{I_I} \cup \mathbf{I_O}}$.

2.7 Hardware Private Circuits

Along with the introduction of this multitude of notions, several concrete realizations of composable gadgets have been proposed. As a limitation, these gadgets mostly cover only atomic gates – like 2-input AND or XOR – as finding efficient constructions, which are provably secure in the (robust) d-probing model under a sufficient composability notion, is a hard task for large functions. The only exception poses GHPC [KSM22], which is a methodology for constructing first-order secure and trivial composable gadgets for arbitrary, vectorial Boolean functions requiring only one bit of fresh randomness per coordinate function, but leaves a relatively large area footprint. Nevertheless, all known gadgets leverage the introduction of gadget-individual fresh randomness to guarantee the confirmatory to the corresponding probe propagation restrictions and hence composability. As complete cipher designs are composed of many of these fundamental gates (AND-XOR), this directly implies a significant randomness overhead for the implementation of full encryption/decryption functions. A concrete instantiation of a gadget realizing a simple AND gate and fulfilling trivial composability by means of the PINI notion is given as HPC2 which is initially presented in [CGLS21]. HPC2 contains two register stages, introducing a latency of two clock cycles regardless of the security order. In Figure 1, a schematic representing an exemplary circuit composed of first-order HPC2 AND gadgets is depicted, where each gadget is supplied with an individual fresh random bit, already resulting in a total of 7 fresh random bits for this simple example. Note that every share-wise implementation of a linear function is already PINI-conform. Hence, the XOR operation does not introduce any additional randomness or latency into the design.

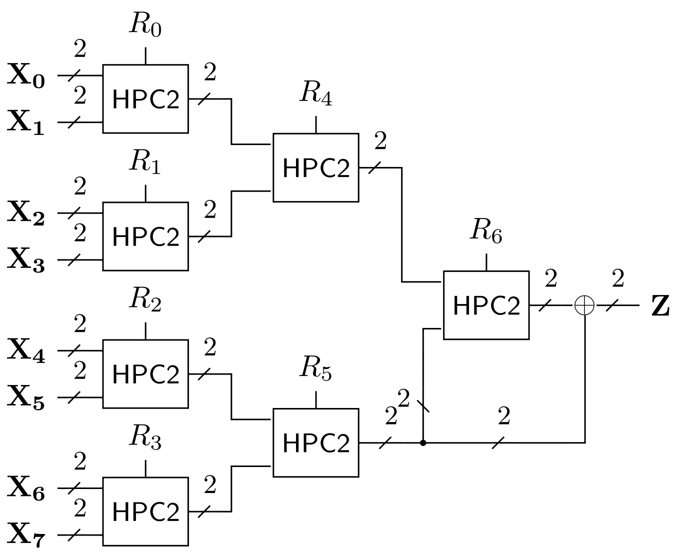


Figure 1: Example for a circuit composed of first-order HPC2 AND gadgets.

2.8 Automated Generation of Masked Hardware

Composable hardware gadgets are well suited for automated masking of hardware designs. The underlying methodology is to simply synthesize a given unprotected design utilizing a restricted library including only those cells for which composable gadgets exist, before substituting each of the resulting cells with the corresponding gadget. If these gadgets provably guarantee secure composition, the resulting netlist will be secure in the (robust) probing model.

AGEMA [KMMS22] is a software tool for realizing this transformation from an unprotected netlist to a protected one. For this, the Verilog netlist is represented as a graph before it is translated into a Mealy machine, i.e., a combinational circuit and a single main register stage. The designer can then specify which part of the netlist should be masked, and which family of gadgets should be instantiated. In the *naive* approach integrated in AGEMA, the structure of the given netlist stays unchanged, and only the cells are substituted by their corresponding gadgets. Of course, since these gadgets introduce additional latency, pipelining or clock gating is automatically applied to ensure the correct functionality of the circuit. This process elegantly ensures security in the glitch-extended robust probing model while being less error-prone than manually applying the masking at the algorithmic level. In this work, we utilize AGEMA to construct the masked ciphers for our case studies, i.e., we provide our gadget definitions to AGEMA when selecting the naive approach. This intentional choice allows fair comparison with several case studies given in the original work [KMMS22].

3 Technique

3.1 Overview

Our objective is to reduce the randomness requirements of gadget-composed cipher designs in hardware. As given in Section 2.7, each 2-input HPC2 AND gadget requires one individual fresh mask bit which is necessary to guarantee composability under the PINI notion and can thus not be nullified. Therefore, pursuing our goal, we introduce COMAR, a methodology for constructing first-order secure gadgets, which allows us to reuse the same

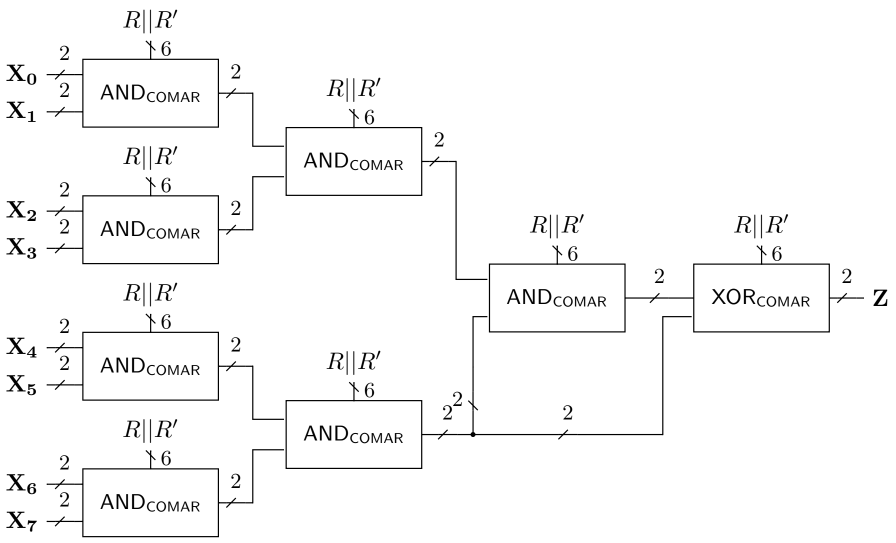


Figure 2: Exemplary circuit composed of COMAR gadgets.

randomness in each gadget, drastically reducing the randomness requirements compared to existing designs which rely on gadget-individual masks. Thanks to the composability of COMAR gadgets, their composition to larger circuits leads to d-probing secure designs (for $d = 1$) under glitch-extended probes.

Naturally, the randomness requirements of HPC2 (and also other composable gadgets) scale with the size and complexity of the target function which is to be masked, i.e., the number of 2-input non-linear gates, while our approach only demands for 6 random bits regardless of the underlying function.

In Figure 2, the same circuit as presented in Section 2.7 is depicted where every input is split into two shares, but now the circuit is realized by instantiating our new COMAR gadgets instead of HPC2 AND. In contrast to HPC2, each of our 2-input AND gadgets needs 6 random masks, instead of only a single one, but as these fresh masks can be reused for every gadget, it is directly obvious that the break-even-point with respect to fresh randomness requirements of our approach compared to HPC2 lies at a number of 6 AND gadgets. Meaning that if a circuit contains more than 6 AND gates, our approach requires less fresh randomness compared to HPC2. This threshold is actually fulfilled for every real-world logic circuit. Figure 2 shows an example to demonstrate this break-even-point. Compared to Figure 1, our approach here needs only 6 instead of 7 bits. Naturally, this advantage becomes more significant for larger circuit.

Note that, in contrast to PINI gadgets, a trivial, i.e., share-wise, realization of an XOR gadget is not possible in case of COMAR. Here, the $\mathsf{XOR}_{\mathsf{COMAR}}$ needs 6 random mask bits, which are the same as those 6 bits reused in every other COMAR gadget. It also needs two register stages, increasing the latency requirements for composed circuits. Nonetheless, there are usecases where it is favorable for designer to trade additional latency against significantly less randomness requirements. We give more detail about this trade-off in Section 4 and Section 5.

3.2 On the Necessity of Introducing Fresh Randomness into Gadgets

Usually, fresh randomness is introduced into gadgets wherever probe propagation is to be stopped into certain inputs of the gadget. Often, introducing fresh randomness conveniently allows simulation of a wire independent of any other value contributing to its distribution.

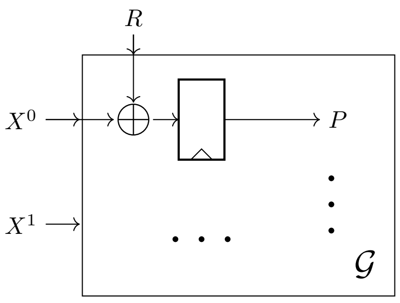


Figure 3: Example probe on a gadget.

Considering the simple example given in Figure 3, a probe on the output of the register can be perfectly simulated tossing a fair coin if R is a fresh random bit, i.e., R is certainly independent of any other wire in the design – in particular independent of X^0. Therefore, $P = R \oplus X^0$ can be perfectly simulated by $P \xleftarrow{\$} \mathbb{F}_2$ which is formalized by Theorem 1.

The situation changes if X^0 depends on R. We assume $X^0 = X \oplus R$ and $X^1 = R$ as stable input signals. In this case, P is not perfectly simulatable using only X^0, as $P = (X \oplus R) \oplus R = X$. To simulate this, we need both input shares X^0 and X^1.

Theorem 1. *If a wire W of a gadget $\mathcal{G}$ is statistically independent of R, $P = \mathsf{Reg}[W \oplus R]$ can be simulated by flipping a coin, i.e., $P \xleftarrow{\$} \mathbb{F}_2$.*

Proof. It holds that $Pr[W \oplus R = 1] = Pr[W = 1] + Pr[R = 1] - 2 \cdot Pr[W = 1 \vee R = 1]$. If now W is statistically independent of R, it follows that $Pr[W \oplus R = 1] = Pr[W = 1] + Pr[R = 1] - 2 \cdot Pr[W = 1]Pr[R = 1]$. It is directly observable that if R is uniformly distributed over $\mathbb{F}_2$, then $Pr[W \oplus R = 1] = Pr[W = 1] + \frac{1}{2} - 2 \cdot \frac{Pr[W=1]}{2} = \frac{1}{2}$, regardless of the value of $Pr[W = 1]$. □

3.3 COMAR

Before presenting the details of COMAR gadgets, we define Simple Sharing below, which is required to understand the underlying concept.

Definition 4 (Simple Sharing)**.** A sharing $\mathbf{F} \in \mathbb{F}_2^n$ of $F \in \mathbb{F}_2$ is called Simple Sharing iff $\exists! i \in [0, n-1]$ with $F^i = F \oplus \bigoplus_{\forall j \neq i} R_j$ and $F^{\forall j \neq i} = R_j \xleftarrow{\$} \mathbb{F}_2$, i.e., $n-1$ shares are set to individual random masks and one share is set to the unmasked value F added by the sum of all $n-1$ masks.

3.3.1 AND

Figure 4 represents the first-order COMAR 2-input AND gate. Based on the above given definitions, every signal at the output of a gate is simply shared with the same mask bit R, e.g., $(A^0, A^1) : (A \oplus M, M)$, and $(B^0, B^1) : (B + M, M)$. Therefore, each input A and B should be first refreshed using R_0 and R_1 respectively. We further require 4 fresh mask bits R'_0 to R'_3 to blind the non-linear monomials. As shown, the shared output is formed as $(C^0, C^1) : (AB \oplus M, M)$ with $M = R'_0 \oplus R'_1 \oplus R'_2 \oplus R'_3$, i.e., satisfying simple sharing given in Definition 4. Further, placing any probe at the output sharing does not propagate to any input shares.

In total, the gadget has a latency of 2 clock cycles and requires 6 fresh mask bits. This is clearly larger than HPC2 2-input AND gadget (Section 2.7), but all our gadgets can receive the same fresh masks $\langle R_0, R_1, R'_0, R'_1, R'_2, R'_3 \rangle$. In other words, independent of the

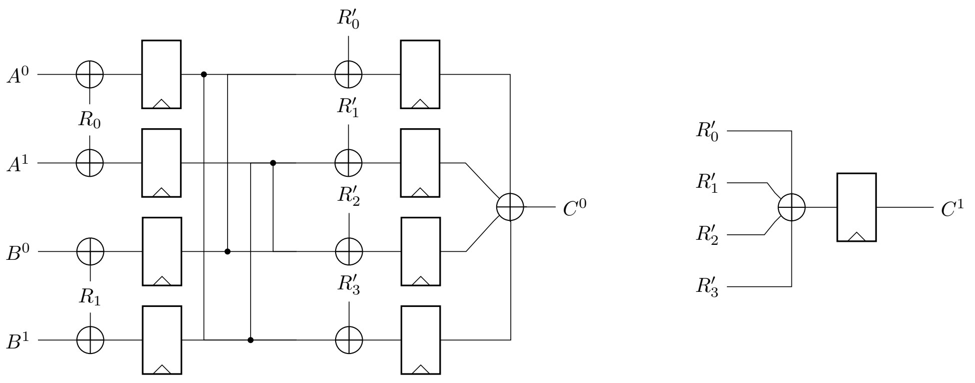


Figure 4: COMAR first-order 2-input AND gate ($\mathsf{AND}_{\mathsf{COMAR}}$).

Algorithm 2 COMAR first-order n-input AND ($\mathsf{AND}^n_{\mathsf{COMAR}}$)

Input: shares $(A^0_0, A^1_0), \ldots, (A^0_{n-1}, A^1_{n-1})$ s.t. $A_{0\leq j<n} = A^0_j \oplus A^1_j$ and fresh masks $(R_j)_{0\leq j<n}$ and $(R'_i)_{0\leq i<2^n}$

Output: shares (C^0, C^1) s.t. $C^0 \oplus C^1 = \prod_j A_j$

```
1: for $\forall j \in \{0,\ldots,n-1\}$ do                          ▷ input sharing refresh
2:     $A'^0_j \leftarrow \mathsf{Reg}[A^0_j \oplus R_j]$
3:     $A'^1_j \leftarrow \mathsf{Reg}[A^1_j \oplus R_j]$
4: end for
5: for $\forall i \in \{0,\ldots,2^n-1\}$ do
6:     $C'_i \leftarrow \mathsf{Reg}\Big[\prod_j A'^{\mathsf{bit}_j(i)}_j \oplus R'_i\Big]$      ▷ $\mathsf{bit}_j(i)$: $j$-th bit of binary representation of $i$
7: end for
8: $C^0 \leftarrow \bigoplus_{0\leq i<2^n} C'_i$
9: $C^1 \leftarrow \mathsf{Reg}\big[\bigoplus_{0\leq i<2^n} R'_i\big]$
```

size of the circuit, and the number of instantiated 2-input AND gates, only 6 fresh masks are required.

This scheme can be extended to AND operations with a higher number of inputs. The corresponding pseudo-code is given in Algorithm 2. For an n-input AND gate, n fresh mask bits are required to refresh the sharing of n inputs, and 2^n fresh mask bits for blinding the non-linear monomials. In short, in a circuit with at most n-input AND gates, $n + 2^n$ fresh mask bits are required which are reused by all corresponding gadgets.

Note that since the second shares A^1, B^1 and C^1 in all such AND gadgets are the same, for the sake of performance, the associated gates and registers (i.e., lines 3 and 9 of Algorithm 2 and $\mathsf{Reg}[A^1 \oplus R_0]$, $\mathsf{Reg}[B^1 \oplus R_1]$, and $\mathsf{Reg}[R'_0 \oplus \ldots \oplus R'_3]$ in Figure 4) do not need to be repeated for every gadget.

3.3.2 XOR

We should highlight that the share-wise application of XOR operations in HPC2 naturally fulfill the composability requirements and can be easily cascaded. However, this is not the case for COMAR, since the output of every gadget is shared by the same mask, and XORing them would clearly lead to unmasked values, i.e., first-order leakage. Therefore,

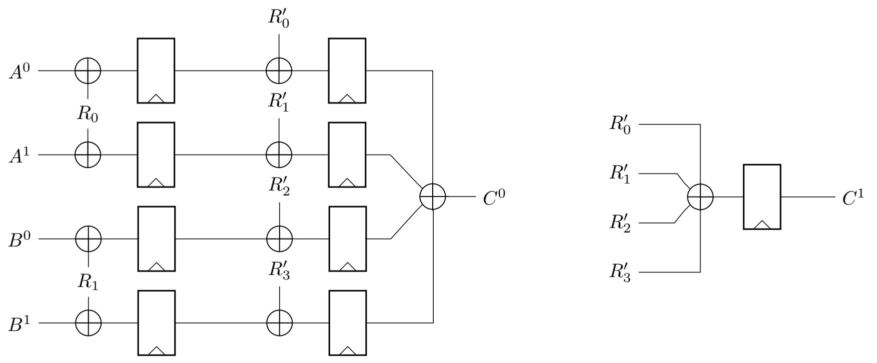


Figure 5: COMAR first-order 2-input XOR gate ($\mathsf{XOR}_{\mathsf{COMAR}}$).

Algorithm 3 COMAR first-order n-input XOR ($\mathsf{XOR}^n_{\mathsf{COMAR}}$)

Input: shares $(A^0_0, A^1_0), \ldots, (A^0_{n-1}, A^1_{n-1})$ s.t. $A_{0\le j<n} = A^0_j \oplus A^1_j$ and fresh masks $(R_j)_{0\le j<n}$ and $(R'_i)_{0\le i<2n}$

Output: shares (C^0, C^1) s.t. $C^0 \oplus C^1 = \bigoplus_j A_j$

```
1: for $\forall j \in \{0,\ldots,n-1\}$ do                ▷ input sharing refresh
2:     $A'^0_j \leftarrow \mathsf{Reg}[A^0_j \oplus R_j]$,
3:     $A'^1_j \leftarrow \mathsf{Reg}[A^1_j \oplus R_j]$
4: end for
5: for $\forall j \in \{0,\ldots,n-1\}$ do
6:     $C'_{2j} \leftarrow \mathsf{Reg}\left[A'^0_j \oplus R'_{2j}\right]$,    $C'_{2j+1} \leftarrow \mathsf{Reg}\left[A'^1_j \oplus R'_{2j+1}\right]$
7: end for
8: $C^0 \leftarrow \bigoplus_{0\le i<2n} C'_i$
9: $C^1 \leftarrow \mathsf{Reg}[\bigoplus_{0\le i<2n} R'_i]$
```

in contrast to HPC2, we constructed an XOR gadget, whose block diagram is shown in Figure 5 and whose structure is very similar to that of the $\mathsf{AND}_{\mathsf{COMAR}}$. Originating from another gadget's output, both inputs A and B are shared with the same $M = R'_0 \oplus \ldots \oplus R'_3$, before they are refreshed by R_0 and R_1 respectively, which are the same R_0 and R_1 used in $\mathsf{AND}_{\mathsf{COMAR}}$ gadgets. In the next stage, each of the refreshed shares is XORed with one $R'_{i\in\{0,\ldots,3\}}$, also being the same as those applied in $\mathsf{AND}_{\mathsf{COMAR}}$ gadget. Therefore, the output shares of the XOR gadget become $C^0 = A \oplus B \oplus M$ and $C^1 = M$, i.e., satisfying the definition of a Simple Sharing with the same M as used for all other inputs and outputs of other gadgets.

As a disadvantage, its area footprint is higher than the HPC2 XOR and it requires two register stages. However, similar to $\mathsf{AND}^2_{\mathsf{COMAR}}$, we can extend the construction of $\mathsf{XOR}_{\mathsf{COMAR}}$ to support XORing a higher number of operands. The corresponding pseudo-code is given in Algorithm 3. Similar to the $\mathsf{AND}_{\mathsf{COMAR}}$, the gates and registers associated to the second share, i.e., $\mathsf{Reg}[A^1 \oplus R_0] \oplus R'_1$, $\mathsf{Reg}[B^1 \oplus R_1] \oplus R'_3$, $C^1 = \mathsf{Reg}[R'_0 \oplus \ldots \oplus R'_3]$ and lines 3 and 9 in Algorithm 3 do not need to be repeated for every gadget.

3.3.3 Proofs

Below, we provide necessary theorems to prove the security of our constructed COMAR gadgets when composed to build larger circuits.

Assumption 1. *Every COMAR gadget with n inputs is supplied with two sets of fresh masks $\left(R\right)_{0\leq j<n}$ and $\left(R'\right)_{0\leq i<n'}$ with $n' = 2^n$ for $\mathsf{AND}^n_{\mathsf{COMAR}}$ and $n' = 2n$ for $\mathsf{XOR}^n_{\mathsf{COMAR}}$. For every evaluation of the gadget, each of these $n + n'$ fresh masks is individually drawn from a uniform distribution at random and is independent of other $n + n' - 1$ bits.*

Theorem 2. *If each R_j is statistically independent of A^0_j and A^1_j, for every $0 \leq j < n$, $\mathsf{AND}^n_{\mathsf{COMAR}}$ provides first-order security under the Probe-Isolating Non-Interference notion.*

Proof.

Input to the first register stage: A probe on the input to the first register stage observes the distribution over $\left(A^{l\in\{0,1\}}_j, R_j\right)$. As A^l_j is independent of R_j, this can be simulated by A^l_j and choosing $R_j \xleftarrow{\$} \mathbb{F}_2$. Hence, an extended probe only propagates into A^l_j.

Input to the second register stage: An extended probe on the input to the second register stage reveals the distribution over $\left(A'^{bit_0(i)}_0 \oplus R_0, A'^{bit_1(i)}_1 \oplus R_1, \dots, A'^{bit_{n-1}(i)}_{n-1} \oplus R_{n-1}, R'_i\right)$ for $i \in \{0, \dots, 2^n - 1\}$. Directly following Theorem 1, each such an observation, so-called O, can be simulated by sampling $O \xleftarrow{\$} \mathbb{F}^{n+1}_2$.

Output: An extended probe on the output observes the distribution over $\big(C'_0, C'_1, \dots, C'_{2^n-1}\big)$, with $C'_i = \prod\limits_j A'^{\mathsf{bit}_j(i)}_j \oplus R'_i$. Since $\prod\limits_j A'^{\mathsf{bit}_j(i)}_j$ is independent of R'_i, this can be simulated by sampling the observation $O \xleftarrow{\$} \mathbb{F}^{(2^n)}_2$, due to Theorem 1.

□

Theorem 3. *If each R_j is statistically independent of A^0_j and A^1_j, for every $0 \leq j < n$, $\mathsf{XOR}^n_{\mathsf{COMAR}}$ provides first-order security under the Probe-Isolating Non-Interference notion.*

Proof.

Input to the first register stage: A probe on the input to the first register stage is similar to that on an $\mathsf{AND}^n_{\mathsf{COMAR}}$, which is covered by Theorem 2.

Input to the second register stage: A probe on the input to the second register stage observes the distribution over $\left(A^{bit_0(i)}_{\lfloor i/2\rfloor} \oplus R_{\lfloor i/2\rfloor}, R'_i\right)$ for $i \in \{0, \dots, 2n-1\}$. Due to the independence of $R_{\lfloor i/2\rfloor}$ and R'_i, following Theorem 1, each such an observation can be simulated by sampling $O \xleftarrow{\$} \mathbb{F}^2_2$.

Output: A glitch-extended probe on the output observes the distribution over $\left(C'_0, C'_1, \dots, C'_{2n-1}\right)$, with $C'_i = A^{bit_0(i)}_{\lfloor i/2\rfloor} \oplus R_{\lfloor i/2\rfloor} \oplus R'_i$. Since each $A^{l\in\{0,1\}}_j$ is independent of R_j (the assumption of the theorem), this can be simulated by sampling the observation $O \xleftarrow{\$} \mathbb{F}^{(2n)}_2$, due to Theorem 1.

□

Assumption 2. *In a circuit composed of only COMAR AND gadgets with at most n inputs and COMAR XOR gadgets with at most m inputs, all gadgets receive the same set of fresh masks $\left(R\right)_{\max(n,m)}$ and $\left(R'\right)_{\max(2^n,2m)}$.*

Theorem 4. *Every masked circuit composed of COMAR gadgets is first-order secure in the glitch-extended d-probing model with $d = 1$.*

Proof.
Let $\mathsf{C}_{\mathsf{COMAR}}$ be a masked (sub)-circuit with m outputs realizing the Boolean function $\mathsf{F} : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$, $\mathsf{F} = \langle \mathsf{F}_0, \mathsf{F}_1, \ldots, \mathsf{F}_{m-1} \rangle$ with each F_i being a coordinate function $\mathbb{F}_2^n \mapsto \mathbb{F}_2$, solely composed of COMAR gadgets.

- $\mathsf{AND}^m_{\mathsf{COMAR}} \circ \mathsf{C}_{\mathsf{COMAR}}$. First, we consider the case where the inputs to an $\mathsf{AND}^m_{\mathsf{COMAR}}$ are the outputs of a circuit composed of $\mathsf{XOR}_{\mathsf{COMAR}}$ and $\mathsf{AND}_{\mathsf{COMAR}}$, i.e., only 2-input gadgets. A probe placed on such an interconnection hence observes distributions of one of the following forms:

$$\left(\mathsf{F}'_i \oplus R'_0, \mathsf{F}''_i \oplus R'_1, \mathsf{F}'''_i \oplus R'_2, \mathsf{F}''''_i \oplus R'_3\right) \quad \text{or} \quad \left(R'_0 \oplus R'_1 \oplus R'_2 \oplus R'_3\right),$$

  with $\mathsf{F}_i = \mathsf{F}'_i \oplus \mathsf{F}''_i \oplus \mathsf{F}'''_i \oplus \mathsf{F}''''_i$ when the output of either an $\mathsf{AND}_{\mathsf{COMAR}}$ or an $\mathsf{XOR}_{\mathsf{COMAR}}$ is probed. These probed signals are the input of the next $\mathsf{AND}^m_{\mathsf{COMAR}}$ as well which are blinded by the corresponding R_i. Following Assumption 2, the above-given probed values are independent of R_i; hence, their simulatability is implied by Theorem 2.

  The same holds when m outputs of $\mathsf{C}_{\mathsf{COMAR}}$ are provided by larger gadgets, i.e., $\mathsf{AND}^{n_1>2}_{\mathsf{COMAR}}$ and $\mathsf{XOR}^{n_2>2}_{\mathsf{COMAR}}$. More precisely, a probe placed on an output of $\mathsf{C}_{\mathsf{COMAR}}$ observes distributions which are blinded by R'_j with $0 \leq j < 2^n_1$ for $\mathsf{AND}^{n_1}_{\mathsf{COMAR}}$ or $0 \leq j < 2n_2$ for $\mathsf{XOR}^{n_2}_{\mathsf{COMAR}}$. Since the first stage of $\mathsf{AND}^m_{\mathsf{COMAR}}$ blinds the inputs with the corresponding R_i, following Assumption 2 and Theorem 2, their simulatability is implied.

- $\mathsf{XOR}^m_{\mathsf{COMAR}} \circ \mathsf{C}_{\mathsf{COMAR}}$. Here, we consider the case where the outputs of $\mathsf{C}_{\mathsf{COMAR}}$ are given to an $\mathsf{XOR}^m_{\mathsf{COMAR}}$. Since the first stage of $\mathsf{XOR}^m_{\mathsf{COMAR}}$ is identical to that of $\mathsf{AND}^m_{\mathsf{COMAR}}$, the statements given above for $\mathsf{AND}^m_{\mathsf{COMAR}} \circ \mathsf{C}_{\mathsf{COMAR}}$ holds true here as well.

□

3.3.4 Other Gates

Other gadgets, e.g., NOT, NAND, OR, NOR, and XNOR, can be easily constructed. In order to maintain the simple sharing, negation should be performed on the first share. More precisely, $(\overline{X^0}, X^1)$ should be the output of the NOT gadget. The same holds for NAND and XNOR. Since $A|B = \overline{\overline{A}\,\overline{B}}$, an OR gadget is constructed by placing a NOT gate at the first share of all inputs and the output of an AND gadget. Therefore, all statements and proofs given in Section 3.3.3 are valid for other COMAR gadgets constructed as explained above.

3.3.5 Discussions

If a circuit is made by only NOT and 2-input AND, NAND, OR, NOR, XOR, and XNOR gates, its first-order secure variant requires 6 fresh mask bits independent of its size and the number of gates. When larger gadgets (i.,e., with a higher number of inputs) are employed, naturally the circuit needs more fresh mask bits as a trade-off between the latency and the amount of demand for fresh randomness. If the circuit instantiates non-linear COMAR gadgets with at most n inputs and linear COMAR gadgets with at most m inputs, the number of required fresh mask bits for the entire circuit is $\max(n, m) + \max(2^n, 2m)$. As stated, the drawback is that the XORs would introduce additional register stages compared to the equivalent first-order HPC2 circuit, whose required number of fresh mask bits equals the number of 2-input non-linear gates. Further, note that HPC2 gadgets are only available for 2-input gates and hence there is no trade-off possible at all. We like to highlight that supporting n-input gadgets is a valuable feature of COMAR, as it allows a designer to structure a circuit differently, i.e., allowing gadgets with a larger or smaller number of inputs, hence adjusting the masked variant towards the specific randomness, area and latency requirements of the use case.

4 Case Studies

In order to examine the performance and security of our scheme, we have considered five cases studies including the round-based implementations of the full cipher encryption functions of AES-128 [DR20], Skinny64-64 [BJK+16], CRAFT [BLMR19], LED-64 [GPPR11], and Midori-64 [BBI+15].

To this end, we have constructed HDL specifications of COMAR gadgets and utilized the recently introduced open-source tool AGEMA [KMMS22], which translates the gate-level netlist of an unprotected implementation to a masked description by exchanging the gates with their corresponding masked gadgets (see Section 2.8). In order to apply AGEMA to our case studies, we first constructed the custom library of AGEMA, where the definition of the gadgets are given, i.e., the I/O port names, number of fresh masks and number of register stages of every gadget. We further used the netlist of the unprotected cores which are given in the case studies of AGEMA available through GitHub[1]. Direct application of AGEMA using COMAR gadgets, would construct the masked circuits correctly, but would result in individual fresh masks being given to each gadget. However, the underlying idea of COMAR is to reuse the fresh masks, i.e., giving the same fresh masks to all gadgets. Therefore, we slightly modified the source code of AGEMA to support this feature. For the sake of simplicity and to highlight the maximum possible randomness optimization, we covered just 2-input COMAR gadgets. This implies that only 6 fresh masks are used for the entire design, which are given to all COMAR gadgets.

After applying our modified version of AGEMA on the aforementioned unprotected full cipher designs we synthesized the resulting circuits with Design Compiler and NanGate 45nm standard cell library. The corresponding results are depicted in Table 1 showing the number of required fresh mask bits, the latency cycles, the critical path delay, and the area footprint of all designs. In order to enable comparison to the state of the art, we further covered the performance figures of the same designs realized utilizing HPC2, GHPC, and $\mathsf{GHPC}_{\mathsf{LL}}$ gadgets. The last two cases are introduced in [KSM22], which – similar to COMAR– are limited to first order. They can highly reduce the latency, particularly the $\mathsf{GHPC}_{\mathsf{LL}}$ variant at cost of a relatively high demand for randomness. Note that in all case studies, we applied the masking on all inputs of the circuit, i.e., in case of encryption both plaintext and key are masked which results in applying the masking on the key schedule of the ciphers as well.

The benefit of COMAR with respect to the number of required fresh masks is obvious. In the most advantageous case, 680 fresh mask bits are required by the round-based AES-128 HPC2 design reduces to 6 bits in COMAR. The critical path delay of the COMAR circuits is also most of the time less than the equivalent HPC2 counterparts, allowing high clock frequencies. On the downside, the added latency of COMAR circuits is significantly more than the HPC2 circuits. This is somehow expected, since – as stated in Section 3 – each $\mathsf{XOR}_{\mathsf{COMAR}}$ gadget introduces two latency cycles. Therefore, in designs with a high number of cascaded XOR gates, the latency of the COMAR circuit would potentially be high. As it can be seen in Table 1, the COMAR circuits of AES-128 and LED-64, which have a strong diffusion layer made by several XORs, have a higher added latency compared to the other case studies.

[1]https://github.com/Chair-for-Security-Engineering/AGEMA

For the sake of completeness, we included the performance figures of some first-order AES designs in Table 1, which have not been designed based on composable gadgets, and it is not straightforward to prove their (robust) probing security when considering the entire encryption function as a whole. Nevertheless, we give a detailed discussion on comparison with the start of the art in Section 5.

4.1 Area Overhead

At first glance, COMAR circuits lead to a higher area overhead, which is shown in Table 1 by pure area. However, for the sake of a fair comparison, the area requirement of a design should include the part needed to realize the sources for generating the fresh random masks. This fact is usually ignored by the state of the art, and different designs are compared ignoring the area required for generating the fresh masks (see e.g., [CRB+16, MMW18, SM21b]). The reason behind such a simplification lies in the nonexistence of a suitable cost function. More precisely, the cost (e.g., required area or energy) of generating a single-bit fresh mask updated every clock cycle is not well-known. Here, we try to include this open question into our consideration by giving an intuition for possible costs of randomness creation.

Randomness Generation. In order to guarantee security, i.e., simulatability of every wire independent of any secret, each fresh mask bit has to be drawn independently from a uniform distribution over $\mathbb{F}_2$, and the adversary should not have control over or knowledge about the fresh masks. Note that, this is not an assumption specific to our work, but a general assumption of Boolean masking. For the sake of simplicity, we can for example assume that an adversary places a probe on $X^0(X^1 \oplus R)$. If R is drawn from a biased distribution, for example with $Pr[R = 0] = 0.6$ instead of 0.5, X^0X^1 will be observed 20% more often than $X^0\overline{X^1}$, trivially leaking information about $X = X^0 \oplus X^1$. Hence, violating the assumption of fresh masks being drawn from a uniform distribution can naturally cause leakage. This is why the required randomness is commonly generated by a PRNG seeded with a random initial input. The independence of the generated bits, i.e., the PRNG's non-predictability, and the discussed uniformity requirement may be assessed following [BRS+].

Looking at the state of the art[2], some works used Linear Feedback Shift Registers (LFSRs), which are randomly seeded during the device power-up. Naturally, a single individual LFSR should be employed for each required fresh mask bit, i.e., sharing an LFSR between multiple parts of the circuit would potentially lead to weaknesses, and hence security degradation, as the underlying masking schemes commonly suppose the independence of fresh masks given to different gadgets (in contrary to the core idea of COMAR). As an example, 31-bit LFSRs with feedback polynomial $x^{31} + x^{28} + 1$ are used in multiple works [KMMS22, SM21b, MMW18, KSM22] as each one can highly efficiently be realized in FPGAs. Each such an LFSR costs 286 GE when synthesized for ASIC using NanGate 45nm library. If a larger period is desired, one can instantiate 64-bit LFSRs with feedback polynomial $x^{64} + x^{63} + x^{61} + x^{60} + 1$. This translates to 565 GE. It should be noted that the majority of the area required for an LFSR is due to the registers. The combinational circuit is made by a few XORs, and the delay of the circuit is at minimum, i.e., that of 2 or 3 XORs and the setup and hold time of registers.

Some other works like [CRB+16] used a reduced-round version of a cipher e.g., PRINCE [BCG+12] in Output FeedBack (OFB) mode implemented in an unrolled

[2]Excluding those which generated the fresh masks on a PC and fed the cryptographic module with all required fresh masks.

Table 1: Performance figures of first-order round-based full cipher encryption functions, excluding PRNGs.

Scheme	**Fresh Random** [bit/cycle]	**Latency** [cycle] added	full	**Critical Path Delay** [ns]	**Area** [GE]
AES-128					
HPC2	680	8	99	2.04	52 597
GHPC	680	8	99	1.48	67 193
GHPC_{LL}	2720	4	55	2.28	52 450
COMAR	6	42	473	1.23	140 214
[SBM21a][a]	8		216		14 256
[SM21a][a]	0		246	6.25	7 136
[Sug19][a]	0^b		266		17 100
[SBHM20][c]	976		10		157 500
Skinny64-64					
HPC2	64	4	165	0.55	6 895
GHPC	64	2	99	0.80	22 850
GHPC_{LL}	1024	1	66	0.85	18 705
COMAR	6	22	759	0.58	22 090
CRAFT					
HPC2	256	8	288	0.94	15 680
GHPC	64	2	96	0.75	22 106
GHPC_{LL}	1024	1	64	0.81	15 748
COMAR	6	14	480	0.61	23 369
LED-64					
HPC2	64	4	165	1.98	7 691
GHPC	64	2	99	1.58	22 904
GHPC_{LL}	1024	1	66	1.84	17 382
COMAR	6	42	1419	0.93	31 163
Midori-64					
HPC2	256	8	153	1.10	17 801
GHPC	64	2	51	1.05	23 901
GHPC_{LL}	1024	1	34	1.08	19 493
COMAR	6	16	289	0.80	36 580

[a] Based on a byte-serial design architecture, and not using NanGate 45nm library.
[b] Using changing of the guards.
[c] Using a masked dual-rail pre-charge logic, and not based on NanGate 45nm library.

fashion. Since no detailed information about the number of covered rounds and the security of such random numbers is given, we cannot predict the corresponding area requirements. Alternatively, Sponge-Based PRNGs are known as a suitable candidate passing the statistical tests proposed by NIST [BDPA10].

Naturally, different variants of Keccak are suggested to be used in Sponge-Based PRNGs, depending on the number of required random bits and the desired period. Such

Table 2: Performance figures of different PRNGs.

Variant		**Bitrate***	**Delay**	**Area**
		[bit/cycle]	[ns]	[GE]
LFSR 31-bit		1	0.17	286
LFSR 64-bit		1	0.18	565
Keccak$-f[25]$	round-based	15/12	0.61	360
Keccak$-f[25]$	unrolled	15	7.10	4.370
Keccak$-f[25]$	unrolled pipeline	15	0.62	4 320
Keccak$-f[50]$	round-based	30/14	0.74	1 154
Keccak$-f[50]$	unrolled	30	9.97	9 136
Keccak$-f[50]$	unrolled pipeline	30	0.76	16 156
Keccak$-f[100]$	round-based	60/16	1.11	2 137
Keccak$-f[100]$	unrolled	60	17.31	20 692
Keccak$-f[100]$	unrolled pipeline	60	1.12	34 192
Keccak$-f[200]$	round-based	136/18	1.14	4 173
Keccak$-f[200]$	unrolled	136	19.84	43 714
Keccak$-f[200]$	unrolled pipeline	136	1.16	75 114
Keccak$-f[400]$	round-based	336/20	1.05	7 989
Keccak$-f[400]$	unrolled	336	20.43	98 846
Keccak$-f[400]$	unrolled pipeline	336	1.06	159 780
Keccak$-f[800]$	round-based	736/22	1.11	16 209
Keccak$-f[800]$	unrolled	736	23.79	205 250
Keccak$-f[800]$	unrolled pipeline	736	1.13	356 598
Keccak$-f[1600]$	round-based	1536/24	1.04	31 361
Keccak$-f[1600]$	unrolled	1536	24.27	466 682
Keccak$-f[1600]$	unrolled pipeline	1536	1.06	752 664

* For all Keccak variants we considered $c = \min(0.4 \times \text{state size}, 64)$, providing a resistance of at most 2^c against state recovery.

designs allow skimming some bits as random values through a squeezing process, i.e., after applying the Keccak permutation function. However, the permutation function is made by a couple of Keccak round functions depending on the size of the employed variant, i.e., the state size. For example, Keccak$[r = 96, c = 104]$ with a 200-bit state and a bitrate of 96 bits, which provides a resistance of about 2^{104} against state-recovery attacks[3], uses Keccak$-f[200]$ as the permutation function involving 18 rounds of the Keccak round function. This means that a typical round-based implementation of such a PRNG would provide 96 random bits every 18 clock cycles, which are obviously not suitable to be used as fresh masks that should be updated every clock cycle. One solution is to realize the unrolled version of such a design, i.e., one clock cycle to fully apply Keccak$-f[200]$ on the state. This highly increases the area requirements as well as the delay of the entire

[3]Note that LFSRs do not provide security against state recovery. Knowing l consecutive outputs of an LFSR with an l-bit state would lead to full recovery of the shift register. Nevertheless, it is still unknown whether security against state recovery in the context of SCA security under probing security model is required.

Table 3: Area footprint of first-order round-based full cipher encryption functions, including PRNGs.

Scheme	Area [GE]		
	LFSR 31-bit	LFSR 64-bit	Keccak, Variant
AES-128			
HPC2	247 281	437 205	409 195, [800]
GHPC	261 673	451 393	423 791, [800]
$\mathsf{GHPC_{LL}}$	830 370	1 589 250	1 557 778, [1600]×2
COMAR	141 932	143 608	144 534, [25]
Skinny64-64			
HPC2	25 218	43 093	82 009, [200]
GHPC	41 154	59 010	97 964, [200]
$\mathsf{GHPC_{LL}}$	311 569	597 265	771 369, [1600]
COMAR	23 808	25 483	26 410, [25]
CRAFT			
HPC2	88 973	160 473	175 460, [400]
GHPC	40 410	58 266	97 220, [200]
$\mathsf{GHPC_{LL}}$	308 612	594 308	768 412, [1600]
COMAR	25 087	26 762	27 689, [25]
LED-64			
HPC2	26 014	43 889	82 805, [200]
GHPC	41 208	59 064	98 018, [200]
$\mathsf{GHPC_{LL}}$	310 246	595 942	770 046, [1600]
COMAR	32 881	34 557	35 483, [25]
Midori-64			
HPC2	91 094	162 595	177 581, [400]
GHPC	42 205	60 061	99 015, [200]
$\mathsf{GHPC_{LL}}$	312 357	598 053	772 157, [1600]
COMAR	38 298	39 974	49 900, [25]

design. To mitigate the delay, pipeline registers can be added at each unrolled round function, realizing an unrolled pipeline design. To give an overview of the performance of such PRNGs, we constructed and synthesized different variants with different design architectures whose results are shown in Table 2. We would like to refer to [SBHM20], where a PRNG based on Keccak$-f$ is used. The authors did not fully explain which Keccak variant has been integrated, but based on the given statements, their PRNG had an internal state of 650-1050 bits, they were able to fetch up to 976 bits every clock cycle, and the PRNG (requiring 14.8 GE) did not impact critical path in their design. It is not fully clear, but we guess that at every clock cycle only one Keccak round function has been applied, not a full Keccak$-f$.

Finally, we considered the area required for the generation of mask bits by each full-cipher HPC2, GHPC, $\mathsf{GHPC_{LL}}$, and COMAR design, and listed their area footprint for different choices of PRNGs in Table 3. This clearly shows that COMAR circuits outperform other gadget-based designs with respect to the area overhead with only one exception, i.e., when using 31-bit LFSRs, LED-64 HPC2 design has the smallest area footprint. As stated, this is due to the extensive number of XORs in the diffusion layer of LED-64, for which several $\mathsf{XOR}_{\mathsf{COMAR}}$ gadgets should be instantiated.

5 Comparison with the State of the Art

5.1 Algorithmic-Level Masking

There exists a variety of works – examples being [SM21a, SBM21a, Sug19] – aiming to achieve high optimization when masking a certain design architecture of a particular cipher like AES. We in the following refer to these approaches as *algorithmic-level* masking. The general advantage of these approaches is that they usually result in implementations with globally optimized overhead requirements. However, as an disadvantages, they are bound to a certain architecture of a cipher and transferring the approach to other designs is not trivially possible or not possible at all. Another major drawback of these schemes is that they are often based on ad-hoc engineering and that there exist no formal security and composability proofs of the final design in a whole. This is due to the fact that existing verification tools like SILVER [KSM20] cannot cope with large hardware circuits. These works usually present experimental evaluations indicating the security of the design. However, their (possible) non-conformity with the formal notions in the probing security model may result in insecure implementations when experienced on different hardware platform or with more accurate measurement setup.

In [SM21a], the authors present a technique to mask a 2-input AND gate where no fresh randomness is needed. They further applied the same technique on S-boxes for different ciphers. However, we would like to highlight that the presented AND realization is not directly composable, and hence special care has to be taken when constructing larger circuits. This manual process might be error-prone and is not well-suited for automated masking of arbitrary implementations. In short such the masked AND gate of [SM21a] is first-order probing secure, but it cannot be used as a gadget in composed circuits. Below we give a counterexample.

Let us assume the target circuit should compute $D \leftarrow \overline{B} \mid (A \;\&\; B)$, with $\mid$ and & denoting the OR and AND operations respectively. This can be realized through computing $C \leftarrow \mathsf{AND}(A, B)$ and $D \leftarrow \overline{\mathsf{AND}(B, \overline{C})}$ by utilizing the AND gadgets of [SM21a]. Note that inversion of a masked value can be realized by inverting one of its shares, i.e., $\overline{C} = (C^0, \overline{C^1})$ or $\overline{C} = (\overline{C^0}, C^1)$. Following the authors' definition in [SM21a], we get

$$\begin{aligned} C^0 &\leftarrow \big[[\overline{A^0}B^0 \oplus B^0] \oplus [\overline{A^0}B^1]\big], & C^1 &\leftarrow \big[[A^1B^0] \oplus [A^1B^1 \oplus B^1]\big], \\ D^0 &\leftarrow \big[[\overline{B^0}C^0 \oplus C^0] \oplus [\overline{B^0}\,\overline{C^1}]\big], & D^1 &\leftarrow \big[[B^1C^0] \,\overline{\oplus}\, [B^1\overline{C^1} \oplus \overline{C^1}]\big], \end{aligned}$$

where square brackets denote values stored into registers. Placing a single probe at the first part of D^0, i.e., the output of the register storing $[\overline{B^0}C^0 \oplus C^0]$, leads to

$$\begin{aligned} \overline{B^0}C^0 \oplus C^0 = B^0C^0 = B^0\big((\overline{A^0}B^0 \oplus B^0) \oplus (\overline{A^0}B^1)\big) &= \overline{A^0}B^0 \oplus B^0 \oplus \overline{A^0}B^0B^1 \\ &= A^0B^0 \oplus \overline{A^0}B^0B^1 = P. \end{aligned}$$

This leaks information about B, since if $B = 1$, i.e, $(B^0, B^1) = (0, 1)$ or $(B^0, B^1) = (1, 0)$, there is only one fullfilling assignment to $A^0B^0 \oplus \overline{A^0}B^0B^1 = 1$, namely $(A^0, B^0, B^1) = (1, 1, 0)$, hence $Pr[P = 1|B = 1] = 1/4$ instead of $1/2$. We verified this by SILVER [KSM20]. Note that this is an arithmetic issue not originating from glitches; hence, placing extra registers in the circuit would not have any effect on this vulnerability.

Furthermore, no formal security proof has been given in [SM21a] for the robust probing security of the full cipher implementations. Although several experimental leakage assessments were performed, we would like to highlight that those experimental assessments only allow to make security statements under a specific setup, whereas the probing security model abstracts from a certain setup in order to attest a design's general SCA resilience.

In another work [SBM21a], the authors efficiently applied two-share Boolean masking on the cubic bijections of the decomposed AES S-box. Their design requires 16 fresh mask bits per S-box, which can be reduced to 8 bits via pipelining. Their aim was to fit such masked cubic functions into BRAM of FPGAs, although they gave ASIC performance results of a byte-serial design in the eprint version of the paper [SBM21b]. Similar to [SM21a], the security of the S-Box has been examined by SILVER, but no statement about the security of the encryption function in a whole can be provided. This becomes more relevant in serialized architectures, where various parts of the circuit are active in different clock cycles and several multiplexers decide which modules' output should be stored in which registers. Hence, there are more locations, where the designer may unintentionally violate the probing security requirements.

Other algorithmic-level approaches, like the one presented in [Dae17], can also lead to good results. The technique, so-called changing of the guards, helps to provide uniformity in $td + 1$ Threshold Implementations when the masked S-Box does not have a uniform output sharing. The underlying concept is conceptually very different to gate-level masking and composable security. Hence, it cannot be directly compared to our approach as it does not offer integration into automated masking tools, but need careful engineering when used in entire cipher designs. For example, we refer to [Sug19], where the same technique has been carefully applied on each module of a tower-field representation of the AES S-Box to overcome their non-uniform output sharing. Similar to many other works, the robust probing security of such a design has not been yet proven, e.g., by means of any tools.

Another promising work is the low-latency masked AES presented in [SBHM20], where the underlying concept is based a masked dual-rail pre-charge logic, called LMDPL. This allowed the authors to make large combinational circuits by LMDPL gates without instantiating register between the gates, hence leading to low-latency masked circuits. The scheme can be seen as a gate-level approach, but the application of LMDPL gates in a circuit requires specific pre-computed values which should be generated by a dedicated module called "mask table generator" designed based on the algorithm of the targeted circuit. The authors have constructed a round-based implementation, and examined its resistance by experimental evaluations showing first-order leakage after 400 million traces. As elaborated in Section 4.1, a form of $\mathrm{Keccak} - f$ has been used as the PRNG to generate fresh masks required for the mask table generator module. As the authors themselves stated, the technique might not exhibit the same level of resistance if it is used in algorithms and designs with smaller and/or fewer S-boxes.

5.2 Gate-Level Masking

Gate-level masking approaches, like in this work, on the other hand, deal with constructing masked variants of composable subcircuits which guarantee the security of the final implementation as a whole and under the (robust) probing security model, regardless of the underlying cipher or the design architecture. These approaches are well suited for automated masking of any unprotected implementation [KMMS22], while usually coming at the cost of higher overheads (area and/or latency). However, their benefits compared to algorithmically-masked, manually-crafted and optimized designs are (i) the ability to prove the security of the resulting circuit, and (ii) the possibility for any engineer to use existing tools in order to construct secure masked circuits without requiring extensive expertise, as the tools and composable gadgets automatically mask any given netlist, preventing engineering flaws which may possibly compromise an implementation's practical security.

Next to NI/SNI [BBD$^+$15, BBD$^+$16], PINI [CS20] was introduced as a formal notion to guarantee gadget composabality. As linear operation can be trivially realized in the PINI framework and the PINI-secure HPC2 AND gadget [CGLS21] introduces less overhead than existing SNI-gadgets, utilizing HPC2 gadgets is currently the most efficient way for gadget-level masking in hardware when only considering basic 2-input logic gates. A PINI-based approach for transforming any vectorial function into a first-order secure and trivial composable gadget is proposed in [KSM22]. However this approach comes with a high area demand if the considered function is large.

The main disadvantage of all NI/SNI/PINI-gadgets is that each gadget of a composed circuit requires individual and fresh randomness, naturally increasing overhead size with circuit complexity. COMAR decouples this relation by enabling the reuse of the same 6 random mask bits for every gadget in the circuit. This way, every implementation can be automatically transformed into a masked variant using only 6 individual masks in total, drastically reducing the overall randomness requirement of the resulting top-module circuit compared to other gate-level masking schemes. Of course, as for COMAR, linear operations introduce additional register stages, applying COMAR gadgets will result in a higher number of clock cycles needed for the masked implementation.

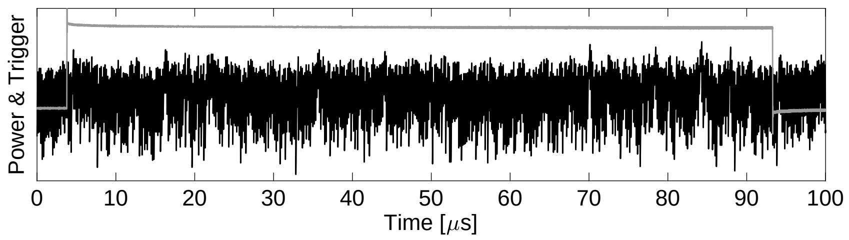
We further like to highlight that we present AND and XOR COMAR gadgets for an arbitrary number of inputs, whereas HPC2 is restricted to the realization of only 2-input non-linear gates. Hence, COMAR can be beneficial if the given unprotected implementation is optimized for 3-bit, 4-bit, or even larger gates. Although the majority of the currently available S-box implementations of different ciphers are optimized for 2-input non-linear gates, there is emerging research in this direction [BDK$^+$21].

With COMAR, we do not aim for an overall better solution, but for offering designers an alternative when area overhead (which directly translates into production cost) is to be reduced while additional latency is acceptable.

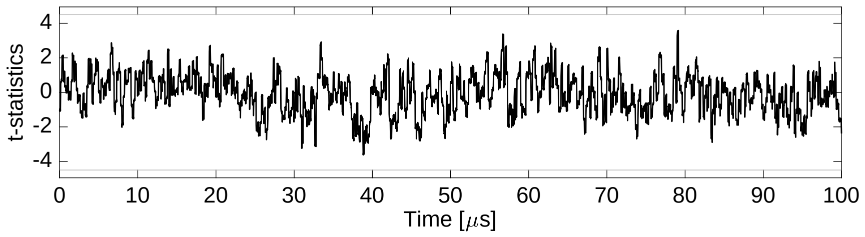
6 Analyses

As the first analysis step, we examined COMAR gadgets and S-boxes of the case studies given in Section 4 by SILVER, an open-source tool verifying masked hardware circuits under different security notions, including the glitch-extended probing model. Since our work is limited to the first security order, the evaluation runtime of SILVER is rather small for our cases. Therefore, we could even examine large circuits, e.g., 4 AES S-boxes followed by a MixColumns composed of COMAR gadgets. Supporting the theory and proofs given in Section 3, SILVER verified first-order security of all our constructions. Since evaluation of larger circuits, particularly those with a sequential loop (as in our case studies), is not feasible with SILVER, we have conducted FPGA-based experiments given as follows. Note, that these experiments, in contrast to other algorithmic-level masking approaches (see Section 5.1), are given for the purpose of presenting a complete work while security in the probing model is already guaranteed due to the gadget composability proven in Section 3.3.3.

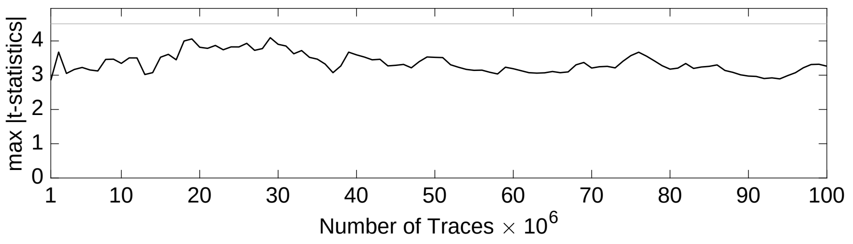
Setup. For this experimental analysis, we consider our COMAR AES-128 round-based encryption function, given in Section 4, where both plaintext and key are masked. For the generation of the fresh masks we instantiated 6 individual 31-bit LFSRs as given in Section 4.1. We implemented the design on the target FPGA of a SAKURA-G SCA-evaluation board [SAK], i.e., a Xilinx Spartan-6 FPGA. We have further collected the corresponding power consumption traces by measuring the voltage drop of a shunt resistor placed in the VDD path of the target FPGA, amplified by 10 dB and then sampled by a digital oscilloscope at a sampling rate of 500 MS/s while the targeted AES design was being operated by a stable clock source at a frequency of 6 MHz. As given in Table 1, the full encryption of the AES design takes 473 clock cycles, resulting in relatively long traces to cover the entire encryption process (see Figure 6a).



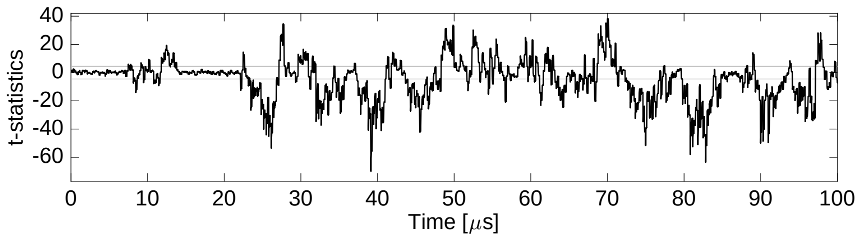
(a) A sample trace



(b) first-order t-test



(c) first-order t-test over number of traces



(d) second-order univariate t-test

Figure 6: FPGA-based analysis of first-order round-based AES-128 encryption design, composed of COMAR gadgets, using 100 million traces.

As the evaluation technique, we applied the common and well-known fixed versus random t-test [CDMG$^+$13], for which the power consumption traces are measured while the circuit is supplied with either a fixed (but masked) plaintext or a random one while the key (also masked) is kept constant for all measurements. In order to avoid false positive/negative evaluations, we followed the procedure given in [SM15] and collected 100 million traces. The analysis results are depicted in Figure 6, confirming our expectations, i.e., first-order security of the design but not second-order. Following [SM15], the second-order t-test was performed by evaluating the distribution over each sample point individually, i.e., univariate, and selecting the second central moment (variance) as the distinguisher between the fixed and random distributions. As we already identified significant leakage for the univariate second-order case, we did not extend our evaluation to multivariate analysis, i.e., the distribution over combination of different sample points.

We should highlight that the underlying cryptographic algorithm of the case study would not have an affect on the result of this evaluation. This is because – as given in Section 4 – AGEMA considers the given design as a Mealy machine, and replaces the gates with the given gadgets. At the end, the equivalent circuit with COMAR gadgets is constructed. In other words, examining other case studies presented in Section 4 would have led to the same analysis report. We additionally have examined our COMAR Skinny64-64 design, whose results are omitted for the sake be brevity. The HDL of our designs including the specification of the COMAR gadgets and the case studies of Section 4 can be found in the GitHub: https://github.com/Chair-for-Security-Engineering/COMAR.

7 Conclusions

In this work, we presented a new set of gadgets – COMAR– offering security and free composability in the glitch-extended robust d-probing model for $d = 1$ and requiring only 6 fresh random bits in total to mask any circuit in its entirety, whereas comparable related works always introduce a linear dependency between the number of (non-linear) gates, i.e., circuit size, and the randomness requirements. With a constant number of 6 fresh random bits when utilizing 2-input COMAR gadgets, we enable the designer to highly optimize for randomness, while the achieved, free composability keeps the design-error susceptibility minimized. Moreover, we extended our gadget definitions to realize masked variants of multiple-input gates, giving the designer the opportunity to adjust a masked circuit's latency and randomness requirements to a specific use case. We further gave formal arguments with respect to the composability and security of our constructions in the first-order (robust) probing security model and practically confirmed our findings by means of an exemplary leakage assessment. Eventually, we discussed our results on the basis of several case studies. Although our approach exceeds latency requirements when compared to related approaches, we show that our methodology outperforms them with respect to the area footprint when a fair comparison is made, i.e., when further considering the area overhead introduced by the randomness source.

To conclude, we think that COMAR offers a valuable increase in the designer's flexibility with respect to different design metrics and use cases, while it remains an interesting question whether a randomness optimization technique can also be found for higher-order composable gadgets. Unfortunately, the same concept cannot be easily transferred to higher orders. When more than one probe is allowed, and two gadgets which reuse the same fresh masks have simple output sharing (defined in Definition 4), two probes are also enough to reveal the XOR difference between unmasked output of the corresponding gadgets. However, there might be other possible techniques to partially reuse some fresh masks in certain circuits, which certainly deserve more attention and more research effort.

Acknowledgments

The work described in this paper has been supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and through the project 393207943 “Security for Internet of Things with Low Energy and Low Power Consumption (GreenSec)".

References

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private Circuits: A Modular Approach. In *CRYPTO 2018*, volume 10993 of *Lecture Notes in Computer Science*, pages 427–455. Springer, 2018.

[BBC$^+$19] Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and François-Xavier Standaert. maskVerif: Automated Verification of Higher-Order Masking in Presence of Physical Defaults. In *ESORICS 2019*, volume 11735 of *Lecture Notes in Computer Science*, pages 300–318. Springer, 2019.

[BBD$^+$15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified Proofs of Higher-Order Masking. In *EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 457–485. Springer, 2015.

[BBD$^+$16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong Non-Interference and Type-Directed Higher-Order Masking. In *CCS 2016*, pages 116–129. ACM, 2016.

[BBI$^+$15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In *ASIACRYPT 2015*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.

[BBP$^+$16] Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness Complexity of Private Circuits for Multiplication. In *EUROCRYPT 201*, volume 9666 of *Lecture Notes in Computer Science*, pages 616–648. Springer, 2016.

[BCG$^+$12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BDF$^+$17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model. In *EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 535–566, 2017.

[BDK$^+$21] Anubhab Baksi, Vishnu Asutosh Dasu, Banashri Karmakar, Anupam Chattopadhyay, and Takanori Isobe. Three Input Exclusive-OR Gate Support for Boyar-Peralta's Algorithm. In *INDOCRYPT 2021*, volume 13143 of *Lecture Notes in Computer Science*, pages 141–158. Springer, 2021.

[BDM$^+$20] Sonia Belaïd, Pierre-Évariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff. Tornado: Automatic Generation of Probing-Secure Masked Bitsliced Implementations. In *EUROCRYPT 2020*, volume 12107 of *Lecture Notes in Computer Science*, pages 311–341. Springer, 2020.

[BDPA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-Based Pseudo-Random Number Generators. In *CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2010.

[BGI$^+$18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal Verification of Masked Hardware Implementations in the Presence of Glitches. In *EUROCRYPT 2018*, volume 10821 of *Lecture Notes in Computer Science*, pages 321–353. Springer, 2018.

[BGR18] Sonia Belaïd, Dahmun Goudarzi, and Matthieu Rivain. Tight Private Circuits: Achieving Probing Security with the Least Refreshing. In *ASIACRYPT 2018*, volume 11273 of *Lecture Notes in Computer Science*, pages 343–372. Springer, 2018.

[BJK$^+$16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BLMR19] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. *IACR Trans. Symmetric Cryptol.*, 2019(1):5–45, 2019.

[BRS$^+$] Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Stefan Leigh, M Levenson, M Vangel, Nathanael Heckert, and D Banks. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD.

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

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware Private Circuits: From Trivial Composition to Full Verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

[CRB$^+$16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 Shares in Hardware. In *CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. *IEEE Trans. Information Forensics and Security*, 15:2542–2555, 2020.

[CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably Secure Hardware Masking in the Transition- and Glitch-Robust Probing Model: Better Safe than Sorry. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):136–158, 2021.

[Dae17] Joan Daemen. Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharing. In *CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 137–153. Springer, 2017.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[DR20] Joan Daemen and Vincent Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition.* Information Security and Cryptography. Springer, 2020.

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

[GM18] Hannes Groß and Stefan Mangard. A unified masking approach. *J. Cryptogr. Eng.*, 8(2):109–124, 2018.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In *TIS@CCS 2016 Vienna*, page 3. ACM, 2016.

[GMK17] Hannes Groß, Stefan Mangard, and Thomas Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In *CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2017.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In *CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[GSM$^+$19] Hannes Groß, Ko Stoffelen, Lauren De Meyer, Martin Krenn, and Stefan Mangard. First-Order Masking with Only Two Random Bits. In *TIS@CCS 2019*, pages 10–23. ACM, 2019.

[HS13] Michael Hutter and Jörn-Marc Schmidt. The Temperature Side Channel and Heating Fault Attacks. In *CARDIS 2013*, volume 8419 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2013.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KMMS22] David Knichel, Amir Moradi, Nicolai Müller, and Pascal Sasdrich. Automated Generation of Masked Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):589–629, 2022.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER - Statistical Independence and Leakage Verification. In *ASIACRYPT 2020*, Lecture Notes in Computer Science. Springer, 2020.

[KSM22] David Knichel, Pascal Sasdrich, and Amir Moradi. Generic Hardware Private Circuits - Towards Automated Generation of Composable Secure Gadgets. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1), 2022.

[MMSS19] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-Resistant Masking Revisited or Why Proofs in the Robust Probing Model are Needed. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):256–292, 2019.

[MMW18] Lauren De Meyer, Amir Moradi, and Felix Wegener. Spin Me Right Round Rotational Symmetry for FPGA-Specific AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):596–626, 2018.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptol.*, 24(2):292–321, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against Side-Channel Attacks: A Formal Security Proof. In *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[RBN$^+$15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In *CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

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

[SBHM20] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson. Low-Latency Hardware Masking with Application to AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):300–326, 2020.

[SBM21a] Aein Rezaei Shahmirzadi, Dusan Bozilov, and Amir Moradi. New First-Order Secure AES Performance Records. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(2):304–327, 2021.

[SBM21b] Aein Rezaei Shahmirzadi, Dusan Bozilov, and Amir Moradi. New First-Order Secure AES Performance Records. *IACR Cryptol. ePrint Arch.*, page 37, 2021.

[SM15] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[SM21a] Aein Rezaei Shahmirzadi and Amir Moradi. Re-Consolidating First-Order Masking Schemes Nullifying Fresh Randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):305–342, 2021.

[SM21b] Aein Rezaei Shahmirzadi and Amir Moradi. Second-Order SCA Security with almost no Fresh Randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):708–755, 2021.

[Sug19] Takeshi Sugawara. 3-Share Threshold Implementation of AES S-box without Fresh Randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):123–145, 2019.

[Tri03] Elena Trichina. Combinational logic design for AES subbyte transformation on masked data. *IACR Cryptol. ePrint Arch.*, 2003:236, 2003.

[WM18] Felix Wegener and Amir Moradi. A Note on Transitional Leakage When Masking AES with Only Two Bits of Randomness. *IACR Cryptol. ePrint Arch.*, 2018:1117, 2018.