Higher-Order Deterministic Masking with Application to Ascon

Vahid Jahandideh, Bart Mennink and Lejla Batina

Radboud University, Nijmegen, The Netherlands
{v.jahandideh,b.mennink,lejla}@cs.ru.nl

Abstract. Side-channel attacks (SCAs) pose a significant threat to the implementations of lightweight ciphers, particularly in resource-constrained environments where masking—the primary countermeasure—is constrained by tight resource limitations. This makes it crucial to reduce the resource and randomness requirements of masking schemes. In this work, we investigate an approach to minimize the randomness complexity of masking algorithms. Specifically, we explore the theoretical foundations of *deterministic* higher-order masking, which relies solely on *offline* randomness present in the initial input shares and eliminates the need for *online* (fresh) randomness during internal computations.
We demonstrate the feasibility of deterministic masking for ciphers such as Ascon, showing that their diffusion layer can act as a refresh subcircuit. This ensures that, up to a threshold number, probes placed in different rounds remain independent. Based on this observation, we propose composition theorems for deterministic masking schemes. On the practical side, we extend the proof of first- and second-order probing security for Ascon's protected permutation from a single round to an arbitrary number of rounds.

Keywords: Side-Channel, Masking, Randomness Complexity, Ascon.

1 Introduction

Side-Channel Attacks and Masking. Side-channel leakages and attacks exploiting these vulnerabilities represent a significant threat to the secure implementation of ciphers, including newly standardized designs like Ascon [2]. A common countermeasure against side-channel attacks is *masking*, which operates in two phases. First, inputs to the cipher, such as the nonce and key, are secret-shared into n shares. Then, the cipher's gates (e.g., AND and XOR) are replaced with special mini-circuits called gadgets. These gadgets process n-shared inputs and generate n-shared outputs.

Masking requires randomness in two distinct contexts: (1) during the initial sharing process, referred to as *offline randomness*, and (2) for the secure operation of gadgets during internal computations, referred to as *online randomness*. Generating high-quality randomness, particularly in the presence of side-channel leakages, is a challenging and resource-intensive task [10,14]. Consequently, considerable research has focused on reducing the randomness requirements for masking [6,21,30].

Lightweight ciphers like Ascon are specifically designed to operate efficiently in resource-constrained environments. This motivates the need for further optimization of protection mechanisms, particularly in reducing their complexity. In this work, we explore a novel approach aimed at minimizing randomness usage. Specifically, we investigate the feasibility of entirely eliminating the dependency on online randomness—a paradigm we term *deterministic masking*.

Deterministic Masking. Proposals for probing-secure deterministic gadgets already exist in the literature [17, 27, 28, 30]. In this work, we aim to expand this concept further by exploring the feasibility of achieving probing security at the circuit level. In general, this goal is challenging: combinations of probing-secure gadgets may become insecure when composed without *refresh gadgets* [15, 29]. Deploying refresh gadgets is a typical solution to address composition issues [5]. However, refresh gadgets require online randomness, which conflicts with the principles of deterministic masking. To overcome this challenge, we demonstrate that the architecture of certain ciphers, such as Ascon, inherently supports deterministic masking. Specifically, the diffusion layer at the end of each round acts as a substitute for refresh gadgets, enabling secure composition without relying on online randomness.

Bricklayer Design of Ciphers. The computations in a cipher are typically organized into multiple rounds, where each round consists of two primary components: a set of non-linear operations, referred to as the *confusion layer*, and a set of linear bit-mixing operations, known as the *diffusion layer*. The non-linear transformations are commonly implemented using smaller functions (S-boxes) that operate in parallel. Daemen and Rijmen [18] introduced the term *bricklayer* to describe this design architecture.

In Figure 1, we illustrate the layout of a deterministic masking scheme for the bricklayer design. Here, $\mathbb{S}$S-box denotes the protected implementation of the S-box, and $\mathbb{S}$Diffusion represents the masked implementation of the diffusion layer.

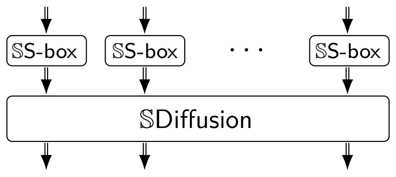

Figure 1: Deterministic masking for the bricklayer architecture, notably in the absence of any refresh gadgets.

Related Works. The widely referenced masking of Ascon, intended for software implementations,[1] is a deterministic masking scheme. However, its security has only been verified for one or two rounds [22], and it remains unclear whether this masking scheme can maintain its security order when targeting more rounds of the cipher.

At the level of single gadgets, Nikova et al. [27] introduced a first-order probing-secure deterministic multiplication gadget with $n = 4$ shares. Building on this foundation, various deterministic masking schemes for S-boxes have been proposed [17, 28, 30].

At the circuit level, Beyne et al. [9] investigated the problem of providing deterministic masking by leveraging the properties of ciphers. Their approach employs linear cryptanalysis, specifically using correlation matrices, to upper-bound the correlation value between two probes placed in different rounds. A low correlation value implies that the probes are uniformly distributed and independent of the secrets. However, this approach is limited to handling two probes and relies on certain independence assumptions regarding the round transformations triggered by round key additions. In Ascon, the focus of this paper, the round transformations are public, and there is no round key involved.

[1] https://github.com/ascon/simpleserial-ascon

1.1 Our Contribution

This work advances both the mathematical foundations and practical applications of deterministic masking schemes. Specifically, our contributions are as follows:

- As our key contribution, we demonstrate that, in bricklayer structures, the diffusion layer functions similarly to a refresh layer by combining the randomness within the gadgets before the start of the next round. This prevents adversaries from leveraging probes across different rounds.

- We demonstrate that the inherent reliance of *simulation*-based approaches and probe-propagation frameworks [5, 11, 23] on online randomness limits their applicability for verifying the security of deterministic masking schemes. To address this limitation, we propose specific tweaks to bridge this gap.

- We establish the requirements for gadgets to achieve first-order probing security in composition and apply these results to round-based ciphers. Importantly, our discussion provides a precise definition of the independence of n-shared values, without imposing restrictions on the corresponding secrets.

- For higher-order probing security, we show that probes at the input and output of the diffusion layer, up to the *linear branch number* [18], remain independent. We extend this observation to the multi-round composition of the diffusion layer.

- We present conditions for achieving higher-order probing security in deterministic masking compositions and apply these conditions to the NIST standard Ascon [2]. Notably, we differentiate the mapping of shares and secrets in this cipher and show that while secrets undergo non-linear transformations in the rounds, shares are subject to linear transformations.

Outline. This paper is organized as follows: Section 2 provides the necessary mathematical background. Section 3 covers the preliminaries. In Section 4, we discuss the limitations of simulation-based approaches and introduce our tweaks to address deterministic masking schemes. Section 5 presents our results for first-order probing security, which are extended to higher-order probing security in Section 6, both in the context of deterministic masking. Finally, the conclusion is given in Section 7.

2 Mathematical Background

In this section, we review the mathematical foundations necessary for verifying probing security. We discuss properties of functions such as bijectivity and explore the linear independence of random variables.

Notation. In this paper, we represent random variables, secrets, and intermediate values with capital letters (e.g., X), while their specific realizations or values are denoted using lowercase letters (e.g., x). Lists of shares and matrices are highlighted with bold letters (e.g., $\mathbf{X}$). The cardinality of a set $\mathcal{X}$ is represented as $|\mathcal{X}|$. The probability distribution of a random variable X is denoted by $\Pr(X)$, and $\Pr(X = x)$ specifies the probability that X takes the value x. Shares of a secret X are indexed as X_i, where the index i starts from one. To indicate that a gate G or circuit C is protected, we append the symbol $\mathbb{S}$ to its name (e.g., $\mathbb{S}\mathsf{G}$ and $\mathbb{S}\mathsf{C}$).

Probability Computation from LUT. Security evaluation of protected circuits often requires the computation of (conditional) probabilities. Here, we describe how such probabilities can be precisely obtained using a lookup table (LUT) representation.

Let random variables X and Y be connected through a deterministic function $Y = F(X, R)$, where R is an r-bit random variable. A LUT with $2^r|X|$ rows represents all possible values taken by X and Y. Additional columns can be appended to this LUT to compute values of a function $G(X, Y)$. Using the LUT, the probability $\Pr(G_1(X, Y) = g_1)$ is computed as:

$$\Pr(G_1(X,Y) = g_1) = \frac{\#\{\text{rows} \mid G_1(X,Y) = g_1\}}{2^r|X|}, \tag{1}$$

and the conditional probability $\Pr(G_1(X, Y) \mid G_2(X, Y))$ is determined by:

$$\Pr(G_1(X,Y) = g_1 \mid G_2(X,Y) = g_2) = \frac{\#\{\text{rows} \mid G_1(X,Y) = g_1 \text{ and } G_2(X,Y) = g_2\}}{\#\{\text{rows} \mid G_2(X,Y) = g_2\}}. \tag{2}$$

Definition 1 (Joint Independence)**.** A set of random variables $\{X_1, \ldots, X_l\}$ is said to be m-jointly independent ($m \leq l$) if, for any subset of size m, $\{X_{i_1}, \ldots, X_{i_m}\}$, the joint probability satisfies:

$$\Pr(X_{i_1}, \ldots, X_{i_m}) = \Pr(X_{i_1}) \cdots \Pr(X_{i_m}). \tag{3}$$

For example, given random binary variables R_1 and R_2, the set $\{R_1, R_2, R_1 \oplus R_2\}$ is 2-jointly independent.

Bijective Mapping and Collisions. A function (also referred to as a mapping) $F\colon \{0,1\}^m \to \{0,1\}^m$ is *bijective* if it establishes a *one-to-one correspondence* between elements of its *domain* and *codomain*. A bijective F is also known as a *permutation* and is invertible. Conversely, if F is not bijective, there exist distinct inputs x_1 and x_2 in its domain such that $F(x_1) = F(x_2)$. In this case, we say that F has a *collision*.

Polynomials and Parity Relations. Given a set of binary and independent random variables $\{X_1, \ldots, X_l\}$, we define binary polynomials as:

$$F_j = \bigoplus_{i=1}^{l} c_i^j X_i,$$

where the constants c_i^j belong to $\{0, 1\}$. The Hamming weight (HW) of a polynomial F_j is defined as $\mathsf{wt}(F_j) = \sum_i c_i^j \in \mathbb{N}$.

In a set of polynomials $\mathcal{F} = \{F_1, \ldots, F_m\}$, we are interested in identifying *parity relations*, and, in particular, the one with the smallest HW. A parity relation is a subset of $\mathcal{F}$ that sums to zero:

$$\bigoplus_{j=1}^{m} d_j F_j = 0,$$

where its HW is given by $\sum_j d_j$. If no such parity relation exists, the m polynomials are said to be *linearly independent*. Since all relations among the polynomials are linear, we conclude that these polynomials (random variables) are m-jointly independent.

For example, let $l = 3$ and $\mathcal{F} = \{X_1 \oplus X_2, X_1 \oplus X_3, X_2 \oplus X_3, X_1 \oplus X_2 \oplus X_3, X_1\}$. In $\mathcal{F}$, there is no parity relation with Hamming weight less than 3; hence, the polynomials are 2-jointly independent.

3 Preliminaries

Circuit View. The computations of a cipher are typically represented by a circuit C, which is uniquely defined as an *acyclic* and *directed* graph consisting of vertices $\mathcal{V}$ and edges $\mathcal{E}$. The elements of $\mathcal{V}$ correspond to *atomic* operations (e.g., XOR and AND) that work collaboratively by passing operands. These operands serve as the connecting members of $\mathcal{V}$ and are listed in $\mathcal{E}$. In hardware terminology, atomic operations are referred to as *gates*, and the corresponding operands are called *wires*. Throughout this text, the terms operation and gate, as well as operand and wire, will be used interchangeably.

Following the standard approach in the field, we assume that a side-channel adversary has knowledge of C and leverages side-channel leakage to infer the values carried by the wires.

Boolean Masking. To protect C from side-channel leakage, the variables in $\mathcal{E}$ are secret-shared using a technique called *Masking.* For a standalone u-bit variable $X \in \mathbb{F}_{2^u}$, masking encodes X with a set of n shares $\mathbf{X} = \{X_1, \ldots, X_n\}$, randomly chosen from $\mathbb{F}_{2^u}$, such that their sum $(\oplus_{i=1}^{n} X_i)$ equals X. We denote $\mathbf{X}$ as an n-sharing and refer to X as the *native* or secret variable.

For a circuit C, masking is achieved by separately encoding its inputs and replacing its gates (elements of $\mathcal{V}$) with mini-circuits called *gadgets.* A gadget is designed to perform the functionality of a gate (or multiple gates) while accepting n-shared inputs and producing n-shared outputs.

For operations that are *additive-affine*[2] in the underlying field, designing an equivalent gadget is straightforward. For instance, in the case of a single-input single-output function $\mathsf{G}\colon \mathbb{F}_{2^u} \to \mathbb{F}_{2^u}$, the corresponding gadget $\mathbb{S}\mathsf{G}\colon (\mathbb{F}_{2^u})^n \to (\mathbb{F}_{2^u})^n$ accepts $\mathbf{X}$ and produces $\mathbf{Y}$ such that $Y_1 = \mathsf{G}(X_1)$ and $Y_i = \mathsf{G}(X_i) \oplus \mathsf{G}(0)$ for $2 \leq i \leq n$. However, for non-linear functions, such as $\mathsf{S\text{-}box}$, the architecture of a gadget is more complex and often case-specific.

Furthermore, while *trivially composing* gadgets may result in functionally correct circuits, it can introduce security vulnerabilities. Composition problems are generally mitigated by inserting *refresh gadgets.* A refresh gadget accepts an n-sharing $\mathbf{X}$ and uses online randomness to compute an n-sharing $\mathbf{Y}$ for the same secret X such that the probability distributions of $\mathbf{X}$ and $\mathbf{Y}$, conditioned on X, are independent. In this case, we say that $\mathbf{X}$ and $\mathbf{Y}$ are two independent n-sharings.

Deterministic Masking. We use this term to describe a gadget or a protected circuit ($\mathbb{S}\mathsf{C}$) that does not consume online (fresh) randomness in its internal computations. Early works by Nikova et al. [27] identified a (first-order) probing secure deterministic multiplication gadget with $n = 4$ shares. Various deterministic masking schemes for $\mathsf{S\text{-}boxes}$ have been presented in [17, 28, 30].

The primary challenge in designing a secure deterministic $\mathbb{S}\mathsf{C}$ lies in the absence of a refresh gadget. Specifically, the *probe propagation* framework (see Section 4), which is a key approach for proving the security of compositions, relies on the use of online randomness, particularly through refresh gadgets. The main contribution of this paper is to establish the groundwork for an alternative approach to probing-secure composition that does not depend on online randomness.

Probing Adversary. The probing model, introduced by Ishai et al. [23], assumes that a side-channel adversary can learn the values carried in up to d wires of a running circuit.

[2]An operation G is additive-affine if, for any x and y in its domain, $\mathsf{G}(x + y) = \mathsf{G}(x) + \mathsf{G}(y) - \mathsf{G}(0)$ holds.

In this model, a gadget (or a protected circuit) is considered secure if a probing adversary with a threshold of d probes cannot deduce any information about the native variables.

To account for physical defaults, the standard probing model has been extended to capture *glitches* and *transition* leakages [12, 20]. However, our focus will remain on the standard probing model, as mitigating glitches and transitions typically does not rely on the availability of fresh randomness.

3.1 Ciphers Blueprint

ciphers are common targets for side-channel attacks, necessitating their protected implementation. These cryptographic primitives typically operate over several *rounds*, with each round applying similar operations on a b-bit *state* $\mathbf{s}$ while preserving its bit length. The round function generally consists of two layers: A linear layer, often referred to as the *diffusion layer*, and a non-linear layer, commonly known as the *confusion layer*.

For most ciphers, including the NIST standard Ascon [2, 19], the structure exhibits additional characteristics:

- As illustrated in Figure 1, the non-linear layer is not implemented as a single b-bit function but rather as a concatenation of similar `S-boxes`, each processing chunks of $\mathbf{s}$ independently.
- The round function acts as a key-less (also referred to as public) *permutation*. A permutation is a bijective mapping from b-bit to b-bit. Consequently, if randomness is injected into the state initially, applying the round function does not create collisions, thus preserving the randomness throughout the process.

3.1.1 Ascon's Structure

Ascon [19] has a bricklayer structure operating on a state of $b = 320$ bits. It employs a *sponge-based* [8] mode of operation for authenticated encryption. The key, tag, and nonce are each 128 bits. As outlined in Figure 2, the encryption process is divided into four phases: (`I`) *Initialization*: The state is initialized with the (private) key K, the (public) nonce N, and a fixed initialization vector (IV). (`II`) *Associated Data Processing*: The state is updated with associated data blocks A_i. Associated data is not confidential and is therefore not encrypted. However, it is absorbed by the sponge to verify its integrity during transit. (`III`) *Plaintext Processing*: Plaintext blocks P_i are injected into the state, and ciphertext blocks C_i are extracted. (`IV`) *Finalization*: The key K is injected again, and the tag T is extracted for authentication.

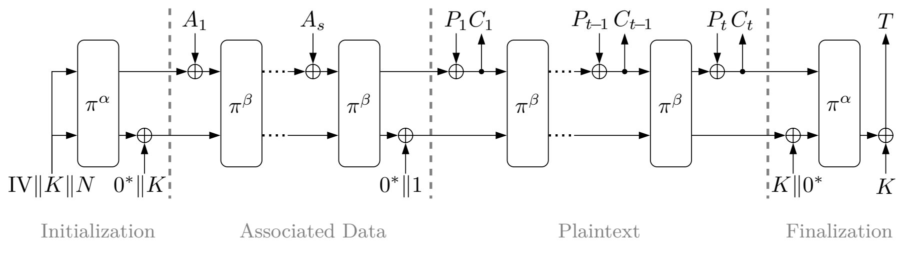

Figure 2: Ascon. π^α and π^β are application of the permutation π in α and β rounds.

Ascon's Permutation. The permutation π is an iterative application of a round transformation on the 320-bit state $\mathbf{s}$. The state is represented as a 5×64 table, $\mathbf{s} = [\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5]^\top$, where each $\mathbf{s}_i$ is 64 bits long. The round transformation comprises three steps: (I) *Addition of Round Constants*: XORs a round-specific 1-byte constant to the state. (II) *Non-linear Layer*: Applies a 5-bit S-box 64 times in parallel. For $1 \le i \le 64$, the S-box is applied to $(\mathbf{s}_1[i], \mathbf{s}_2[i], \mathbf{s}_3[i], \mathbf{s}_4[i], \mathbf{s}_5[i])$, where $[i]$ denotes the i-th bit. (III) *Diffusion Layer*: XOR rotated parts of the state. Specifically, for $1 \le j \le 5$, $\mathbf{s}_j$ is updated as:

$$\mathbf{s}_j \leftarrow \mathbf{s}_j \oplus (\mathbf{s}_j \lll \theta_j) \oplus (\mathbf{s}_j \lll \theta'_j), \tag{4}$$

where the rotation amounts θ_j and θ'_j are specified in [19].

Ascon's S-box. The S-box is χ_5 (defined below) between two thin 5-bit linear and affine layers, as depicted in Figure 3. The round function contains 64 S-boxes, one for each i.

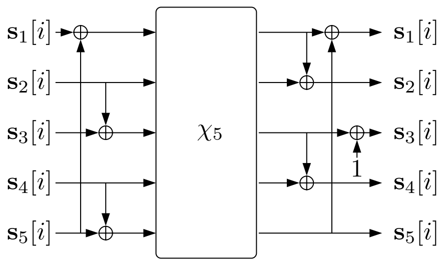

Figure 3: Ascon's S-box.

The Map χ_m. The non-linear map $\chi_m \colon \{0,1\}^m \to \{0,1\}^m$ is defined as follows: For binary input vector $(X^1, \dots, X^m)$, the output Y^i, for $1 \le i \le m$, is computed as:

$$Y^i = X^i \oplus (1 \oplus X^{i+1}) X^{i+2}, \tag{5}$$

where the boundary values X^m and X^{m+1} are substituted by X^1 and X^2, respectively.

The map χ_m was investigated by Daemen in [16]. For odd m, it is bijective. This map is deployed in several important cryptographic algorithms. For instance, χ_5 is used in SHA-3 [1] and Ascon [19]. Notably, in both algorithms, χ_m provides the only source of non-linearity in the round function.

Protecting the Permutation. The addition of the key K at the end of the *initialization* phase and the start of the *finalization* phase, as shown in Figure 2, prevents side-channel state recovery attacks (carried out during Associated Data and Plaintext Processing) from escalating into *key recovery* or *tag forgery* attacks [7, 19].

To specifically mitigate key recovery attacks, only the initialization and finalization phases require protection. Considering that both phases consist of a single invocation of the permutation π, our focus will be on securing the computations of π.

3.2 Composition Problem

Protecting the permutation requires more than simply designing probing-secure $\mathbb{S}$S-box and $\mathbb{S}$Diffusion gadgets and combining them. It is a well-known issue that trivially composing d-probing secure gadgets can still leave them vulnerable to fewer than d probes. This concern has been addressed in several works [5, 15, 29]. However, since the composition challenge is central to our discussion, we pause to clarify it further with an example.

Example 1. Suppose a 3-bit $\mathsf{S\text{-}box}$ is formed by combining χ_3 and a linear layer. Given input bits $\{A, B, C\}$, χ_3 computes the output bits, based on (5), as:

$$\begin{cases} A' = A \oplus (1 \oplus B)C, \\ B' = B \oplus (1 \oplus C)A, \\ C' = C \oplus (1 \oplus A)B. \end{cases}$$

The subsequent linear layer computes $A'' = A' \oplus B'$, $B'' = B'$, and $C'' = 1 \oplus C'$. For this $\mathsf{S\text{-}box}$, we propose the following $\mathbb{S}\mathsf{S\text{-}box}$ at $n = 2$, where $\mathbb{S}\chi_3$ operates as:

$$\begin{cases} A'_1 = [A_1 \oplus (1 \oplus B_1)C_1] \oplus B_2C_1, & A'_2 = [A_2 \oplus (1 \oplus B_1)C_2] \oplus B_2C_2, \\ B'_1 = [B_1 \oplus (1 \oplus C_1)A_1] \oplus C_2A_1, & B'_2 = [B_2 \oplus (1 \oplus C_1)A_2] \oplus C_2A_2, \\ C'_1 = [C_1 \oplus (1 \oplus A_1)B_1] \oplus A_2B_1, & C'_2 = [C_2 \oplus (1 \oplus A_1)B_2] \oplus A_2B_2. \end{cases} \tag{6}$$

The masked linear layer works as:

$$A''_1 = A'_1 \oplus B'_1, \quad A''_2 = A'_2 \oplus B'_2, \quad B''_1 = B'_1, \quad B''_2 = B'_2, \quad C''_1 = 1 \oplus C'_1, \quad C''_2 = C'_2.$$

This $\mathbb{S}\mathsf{S\text{-}box}$ is first-order probing secure. The security claim can be directly proved by showing that each intermediate V is independent of the secrets (A, B, C). That is, $\Pr(V \mid A, B, C) = \Pr(V)$. Verifying this involves enumerating all possible inputs, yielding $2^{3n} = 64$ inputs in this example.

Now, consider a 3-bit permutation $\mathsf{f}^{(r)}$, constructed by applying $\mathsf{S\text{-}box}$ r times, i.e.,

$$\mathsf{f}^{(r)}(A, B, C) = \mathsf{S\text{-}box}(\ldots(\mathsf{S\text{-}box}(A, B, C))\ldots).$$

We show that trivially composing the given $\mathbb{S}\mathsf{S\text{-}box}$ does not result in a secure computation of $\mathbb{S}\mathsf{f}^{(r)}(A_1, A_2, B_1, B_2, C_1, C_2)$ for $r > 1$. For example, by enumerating all input shares, for A''_1 at the output of $\mathbb{S}\mathsf{f}^{(16)}$, we find:

$$\Pr(A''_1 = 0 \mid A = 0, B = 0, C = 1) = 0, \quad \text{whereas} \quad \Pr(A''_1 = 0) = \frac{1}{2}.$$

This shows that A''_1 is *dependent* on native variables, violating first-order probing security.

Although this $\mathsf{S\text{-}box}$ is bijective, the given $\mathbb{S}\mathsf{S\text{-}box}$ is *not* a bijective gadget and has *collisions*. For instance, the inputs:

$$\begin{cases} (A_1 = 1, A_2 = 1, B_1 = 1, B_2 = 1, C_1 = 0, C_2 = 0), \\ (A_1 = 0, A_2 = 0, B_1 = 0, B_2 = 0, C_1 = 1, C_2 = 1), \end{cases}$$

map to the same output. Collisions reduce the output space, consume initial randomness, and make the intermediates of subsequent rounds secret-dependent. For $\mathsf{f}^{(r)}$, collisions increase with r, as shown in Figure 4.

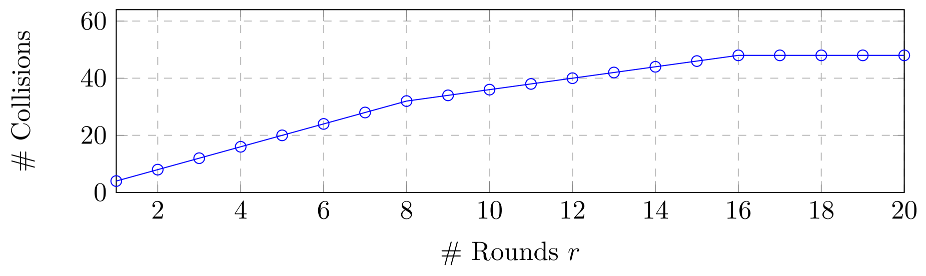

Figure 4: More collisions occur with increasing number of rounds.

□

Role of Online Randomness. A generic solution to composition issues involves incorporating online randomness, particularly through the insertion of refresh gadgets. Online randomness compensates for entropy loss caused by collisions and blocks the *propagation* of adversary probes (detailed in the following section).

4 Simulation and Probe Propagation

Direct verification of d-probing security, which involves checking the independence of any d-tuple of intermediates from native variables, becomes impractical as the sharing order n increases. The *simulation* approach by Ishai et al. [23] addresses this challenge. To prove d-probing security, one shows that the distribution of up to d probes (on intermediates) can be simulated using fewer than n shares of each input.

The simulation approach follows basic rules. Let V_1 and V_2 be intermediates (not necessarily shares of a secret), and let R represent an online randomness variable:

- If a probe P is of the form $P = R \oplus V_1$, and R is not involved in the computation of any other probe, then R *hides* V_1. In this case, the distribution of P can be simulated using an independent random variable R' as $P = R'$, without requiring V_1 for the simulation.

- If a probe P is of the form $P = V_1 \oplus V_2$ or $P = V_1V_2$, both V_1 and V_2 are required for the simulation.

Definition 2 (Probe Propagation)**.** If simulating a probe P requires knowledge of some intermediate V, we say that P propagates to V, and V is called a propagated probe.

Probes always propagate toward inputs. The propagation rules are depicted in Figure 5. Notably, the addition of online randomness, as stated above, halts further propagation. Coron [13] introduced the following generalization to the randomness addition rule.

Probe Elimination [13]. In a set of probes $\mathbf{P}$, let a probe P_i be of the form $P_i = R \oplus P'$, where P' is a function of intermediates other than R. If the randomness R is not involved in any other probe, then P_i can be *eliminated* from $\mathbf{P}$, and its distribution can be simulated using an independent randomness R'. In this case, R hides the intermediates in P'. The probe elimination process can iteratively continue by updating $\mathbf{P} \leftarrow \mathbf{P} - P_i$. The remaining intermediates required to compute the final $\mathbf{P}$ are needed for the simulation. We clarify this approach with an example.

Example 2. Let the sharing order be $n = 2$, and let $\mathbf{A} = \{A_1, A_2\}$ be the shares of the native A. Assume the adversary has the following probes:

$$P_1 = R_1 \oplus A_1A_2 \oplus R_2, \quad P_2 = R_2 \oplus A_1 \oplus A_2, \quad P_3 = A_1,$$

where R_1 and R_2 are randomness variables. Applying the probe elimination approach:

- P_1 is of the form $P_1 = R_1 \oplus P'$, and since R_1 is not involved in P_2 and P_3, we eliminate P_1, updating $\mathbf{P} = \{P_2, P_3\}$.

- In the second iteration, $P_2 = R_2 \oplus P'$, and R_2 is not involved in the computation of the current probes. We eliminate P_2, updating $\mathbf{P} = \{P_3\}$.

The remaining probe $P_3 = A_1$ requires only one input share (A_1) for simulation. Thus, the adversary's probes can be simulated with knowledge of only one input share. □

Composition Rules. The simulation rules are useful for proving the probing security of standalone gadgets. However, they are insufficient for verifying the security of compositions. Barthe et al. [5] introduced the concept of *strong non-interference* (SNI) to address this issue. A gadget is SNI if simulating up to a threshold number of output shares does not require any input shares. Thus, an SNI gadget acts as a barrier, stopping further propagation of probes. Cassiers and Standaert [11] later refined this concept by introducing *probe isolating non-interference* (PINI) gadgets. PINI gadgets are trivially composable and do not require additional refresh gadgets at interfaces for secure composition.

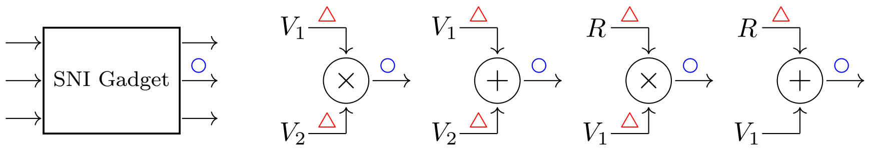

Figure 5: Probe propagation rules. Blue circles represent probes, and red triangles represent propagated probes. V_1 and V_2 are intermediates, and R is a randomness variable.

Simulation Needs Online Randomness. The probe elimination technique, as well as the design of SNI and PINI gadgets, inherently requires randomness.[3] This reliance on randomness limits their deployment for proving the probing security of deterministic masking schemes, where the use of online randomness is avoided. This paper explores tweaks to handle the absence of online randomness. As a first step, we demonstrate that simulation-based security is more restrictive than the core definition of probing security.

4.1 Limitations of Simulation-Based Security

Given a set of probes $\mathbf{P}$ and a set of natives $\mathbf{V}$, *probing security* requires that $\mathbf{P}$ conveys no information about $\mathbf{V}$, i.e.,

$$\Pr(\mathbf{P} \mid \mathbf{V}) = \Pr(\mathbf{P}).$$

Probing security addresses the equivocation about natives and, in some cases, is a broader concept than simulation of the probes (see also Appendix E of the full version of [5]). The following examples clarify this point.

A 2-input Case. At $n = 2$, let $\{A_1, A_2\}$ and $\{B_1, B_2\}$ be shares of secrets A and B. For simulating a probe P of the form $P = A_1 \oplus B_1B_2$, the simulation rules require A_1, B_1, and B_2. Specifically, since both shares of B are required, the approach suggests that P depends on B. However, if the shares of A and B are independent, direct enumeration over all $2^{2n} = 16$ possible inputs reveals that P is independent of A and B. In this case, A_1 hides the secret-dependent term B_1B_2. $\square$

A 3-input Case. Another example involves probing intermediates of $\mathbb{S}\chi_3$ as defined in (6). For instance, a probe on A'_1 results in:

$$P = [A_1 \oplus (1 \oplus B_1)C_1] \oplus B_2C_1.$$

Simulating P requires the shares B_1, B_2, C_1, and A_1. Specifically, both shares of B are required, indicating that the simulation approach cannot prove probing security. However,

[3]If no online randomness is involved, input shares *uniquely* determine each output share. Consequently, simulating (reproducing) output shares without using any of the input shares becomes impossible.

the randomness brought in by A_1 hides the secret-dependent term:

$$(1 \oplus B_1)C_1 \oplus B_2 C_1 = (1 \oplus B)C_1.$$

□

These examples illustrate that probing security is a broader notion than simulation-based security. They also demonstrate how randomness introduced by the shares can be used to achieve probing security. However, this approach has an important prerequisite: the shares of inputs must be independent.

5 First-Order Deterministic Masking

Blinding with input shares is an effective approach to achieving probing security at the level of a single gadget. However, this technique alone is insufficient to establish the security of deterministic masking schemes in the broader context of circuit-level computations. In this section, we address this challenge by developing composition rules for deterministic masking in circuits. Specifically, we focus on circuits that follow a *bricklayer architecture* (see Figure 1).

To begin, we address the case of mitigating a single probe by presenting the following lemma:

Lemma 1. *At a sharing order n, a deterministic masked circuit composed of first-order probing-secure gadgets is itself first-order probing secure if each gadget with m inputs receives m-jointly independent n-sharings as its input.*

Proof. The adversary is assumed to place only one probe, which will necessarily target one of the gadgets in the circuit. Each gadget, under the given conditions, operates with the same input structure as it would in isolation. Consequently, any single probe—regardless of its placement—remains independent of the native secrets. This independence guarantees the circuit's first-order probing security. □

Recall that $\mathbf{X} = \{X_1, \ldots, X_n\}$ is an n-sharing for secret X if $\mathbf{X}$ is distributed uniformly over all assignments satisfying $X = \bigoplus_{i=1}^{n} X_i$. Moreover, two n-sharings $\mathbf{X}$ and $\mathbf{Y}$ are independent if the collection of any $n-1$ shares from $\mathbf{X}$ and $n-1$ shares from $\mathbf{Y}$ are $(2n-2)$-jointly independent. This notion can be extended to an arbitrary number of n-sharings.

We intentionally place no restrictions on the corresponding secrets. The secrets can be dependent, identical, or even known (such as initialization vectors (IV) or nonces). For future reference, let us make the notion of independence of n-sharings more concrete.

Definition 3 (Independence of n-sharings)**.** A set of n-sharings $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ are independent if the collection of any $n-1$ shares of each $\mathbf{X}_i$ are $[m \times (n-1)]$-jointly independent.

Bijectivity and Being n-Sharing are Different. The bijectivity of a gadget in our deterministic masking setup is essential for preserving the input randomness. However, bijectivity alone does not guarantee that the gadget will produce n-shared outputs when provided with n-shared inputs. We clarify this distinction with the following illustrative example.

Example 3. Assume gadget $\mathbb{S}\mathsf{G}_i$, operating on an input n-sharing $\mathbf{X}$, computes its output as $\mathbf{Y} = \{X_1 \oplus X_i, X_2, \ldots, X_n\}$. That is, it adds share X_i to share X_1. While this gadget is bijective, its output is not n-sharing, which compromises its compositional security. For instance, we observe:

$$\mathbf{Y} = \mathbb{S}\mathsf{G}_2\left(\mathbb{S}\mathsf{G}_3\left(\ldots\left(\mathbb{S}\mathsf{G}_n(\mathbf{X})\right)\right)\right),$$
$$Y_1 = \bigoplus_{i=1}^{n} X_i = X.$$

Lemma 1 in Bricklayer Designs. Here, we examine the applicability of Lemma 1 in the practical context of bricklayer designs. As previously discussed in Section 3.1, certain ciphers, including Ascon, employ the bricklayer design. In this design, a b-bit state $S = S^1 \parallel \cdots \parallel S^t$ is processed in t chunks of m-bit length ($b = t \times m$) as follows:

$$\begin{aligned} S &\leftarrow \mathsf{S\text{-}box}(S^1) \parallel \cdots \parallel \mathsf{S\text{-}box}(S^t), \\ S &\leftarrow \mathsf{Diffusion}(S). \end{aligned} \tag{7}$$

In the protected version, the initial b-bit state S is replaced with an (nb)-bit n-shared state $\mathbf{S}$, where each bit of S is separately encoded. For masking, $\mathsf{S\text{-}box}$ is replaced with the $\mathbb{S}\mathsf{S\text{-}box}$ gadget. This gadget is bijective and operates on m-jointly independent n-shared inputs to produce m-jointly independent n-shared outputs, for which explicit realizations exist. The diffusion layer, being additive-affine, is straightforward to mask. The protected circuit functions as follows:

$$\begin{aligned} \mathbf{S} &\leftarrow \mathbb{S}\mathsf{S\text{-}box}(\mathbf{S}^1) \parallel \cdots \parallel \mathbb{S}\mathsf{S\text{-}box}(\mathbf{S}^t), \\ \mathbf{S} &\leftarrow \mathbb{S}\mathsf{Diffusion}(\mathbf{S}). \end{aligned} \tag{8}$$

These operations iterate over multiple rounds. Our goal is to demonstrate that, in each round, the inputs to the gadgets satisfy the requirements of Lemma 1. Consequently, if these gadgets are first-order probing secure, the composition remains $d = 1$ secure for any number of rounds.

Lemma 2. *For the given bricklayer design, at any round, the inputs of each* $\mathbb{S}\mathsf{S\text{-}box}$ *are m-independent n-sharings, and the inputs of* $\mathbb{S}\mathsf{Diffusion}$ *are b-independent n-sharings.*

Proof. After round r, let the state be S^r, and the n-shared state be $\mathbf{S}^r = \{S_1^r, \ldots, S_n^r\}$, where S_i^r are the shares of the secret S^r, and we have $S^r = \bigoplus_{i=1}^{n} S_i^r$. Our aim is to show that any $n-1$ collection of shares S_i^r are distributed uniformly, which implies that $\mathbf{S}^r$ is an n-sharing for the secret S^r, as depicted in Figure 6.

We emphasize that we do not claim any specific distribution of S^r, as it is deterministically computed from the (non-random) initial state S. For example, in Ascon (as shown in Figure 2), S is derived by concatenating a fixed initialization vector (IV), a private key K, and a public nonce N.

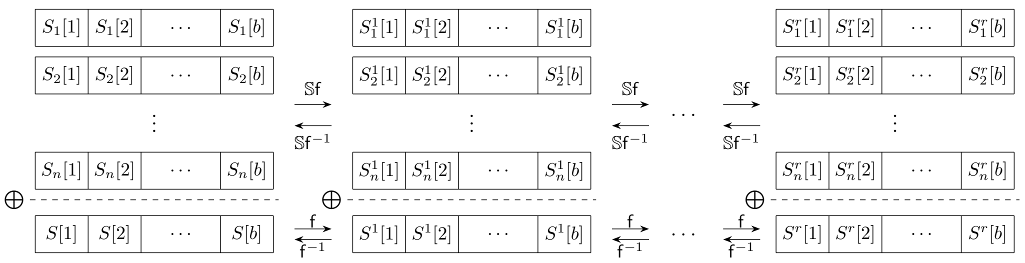
$S_1[1]$ $S_1[2]$ $\cdots$ $S_1[b]$
$S_2[1]$ $S_2[2]$ $\cdots$ $S_2[b]$
$S_n[1]$ $S_n[2]$ $\cdots$ $S_n[b]$
$S[1]$ $S[2]$ $\cdots$ $S[b]$
$\mathbb{S}\mathsf{f}$
$\mathbb{S}\mathsf{f}^{-1}$
f
f^{-1}
$S_1^1[1]$ $S_1^1[2]$ $\cdots$ $S_1^1[b]$
$S_2^1[1]$ $S_2^1[2]$ $\cdots$ $S_2^1[b]$
$S_n^1[1]$ $S_n^1[2]$ $\cdots$ $S_n^1[b]$
$S^1[1]$ $S^1[2]$ $\cdots$ $S^1[b]$
$S_1^r[1]$ $S_1^r[2]$ $\cdots$ $S_1^r[b]$
$S_2^r[1]$ $S_2^r[2]$ $\cdots$ $S_2^r[b]$
$S_n^r[1]$ $S_n^r[2]$ $\cdots$ $S_n^r[b]$
$S^r[1]$ $S^r[2]$ $\cdots$ $S^r[b]$
$\oplus$

Figure 6: State and its shared variant evolving over r rounds.

Without loss of generality, we show that the first $n-1$ shares of the state, i.e., S_1^r to S_{n-1}^r, are uniformly distributed over b-bits. Specifically, they can take arbitrary values s_1 to s_{n-1}, while the initial (unshared) state S is fixed to $S = s$, and consequently, S^r will be a corresponding value s^r. Under these settings, we write:

$$\begin{aligned}
&\Pr\left(S_1^r = s_1, \ldots, S_{n-1}^r = s_{n-1} \mid S^r = s^r\right) = \\
&\Pr\left(S_1^r = s_1, \ldots, S_{n-1}^r = s_{n-1}, S_n^r = \bigoplus_{i=1}^{n-1} s_i \oplus s^r\right) = \\
&\Pr\left(\mathbf{S} = \mathbb{S}\mathsf{f}^{-r}(s_1, \ldots, s_{n-1}, \bigoplus_{i=1}^{n-1} s_i \oplus s^r), \quad S = \mathsf{f}^{-r}(S^r)\right) = \frac{1}{2^{b(n-1)}},
\end{aligned}$$

where $\mathbf{S}$ is the sharing of the initial state S, for which each $n-1$ collection of shares is uniformly distributed (due to masking) over b-bit values. The function $\mathbb{S}\mathsf{f}^{-1}$ is the inverse of the masked round operations in Equation (8), and f^{-1} is the inverse of the round operations in Equation (7). Both inverses exist since the (masked) round operations are bijective.

Taking similar steps, we can demonstrate that the inputs to $\mathbb{S}$Diffusion are also n-sharings. Furthermore, if $\mathbf{S}^r$ is the concatenation of b-jointly independent n-sharings, then any subset of $\mathbf{S}^r$ with m columns (see Figure 6) is m-jointly independent n-sharings. $\square$

5.1 Ascon's First-Order Deterministic $\mathbb{S}$S-box Gadgets

Dae-$\mathbb{S}\chi_5$ Gadget. Daemen et al. [17] proposed a two-share, first-order secure $\mathbb{S}\chi_5$ function, which we label as Dae-$\mathbb{S}\chi_5$. Its security for Ascon's S-box was formally verified using the maskVerif tool [3,4].[4] The scheme was implemented by members of the Ascon development team for software protection of Ascon.[5] The implementation, as explained in [22], is *bit-sliced* and includes useful features such as *share rotation* to enhance side-channel robustness.

The Dae-$\mathbb{S}\chi_5$ gadget requires one bit of randomness. However, we still classify it as deterministic because no online randomness is required for internal computations, and subsequent rounds reuse the initial randomness. Let $\{A, B, C, D, E\}$ denote the input natives, and recall our notation: the shares of a native A are denoted as A_1 and A_2. Let $\{R_1, R_2\}$ be variables initialized to $R_1 = r_j$ and $R_2 = r_j$. For $1 \leq j \leq 64$, r_j is a random bit, with one such bit allocated for each of the 64 gadgets. The computations of Dae-$\mathbb{S}\chi_5$ are arranged as follows:

$$\begin{array}{lll}
R_1 \oplus\!\!\leftarrow \overline{D_1}E_1 \oplus D_2E_1, & R_2 \oplus\!\!\leftarrow \overline{D_1}E_2 \oplus D_2E_2, & \\
E_1 \oplus\!\!\leftarrow \overline{A_1}B_1 \oplus A_2B_1, & E_2 \oplus\!\!\leftarrow \overline{A_1}B_2 \oplus A_2B_2, & \\
B_1 \oplus\!\!\leftarrow \overline{C_1}D_1 \oplus C_2D_1, & B_2 \oplus\!\!\leftarrow \overline{C_1}D_2 \oplus C_2D_2, & \\
D_1 \oplus\!\!\leftarrow \overline{E_1}A_1 \oplus E_2A_1, & D_2 \oplus\!\!\leftarrow \overline{E_1}A_2 \oplus E_2A_2, & \\
A_1 \oplus\!\!\leftarrow \overline{B_1}C_1 \oplus B_2C_1, & A_2 \oplus\!\!\leftarrow \overline{B_1}C_2 \oplus B_2C_2, & \\
C_1 \oplus\!\!\leftarrow R_1, & C_2 \oplus\!\!\leftarrow R_2, \quad R_1 \leftarrow R_2. &
\end{array} \tag{9}$$

Here, the operation $X \oplus\!\!\leftarrow Y \oplus Z$ means Y is first added to X, followed by the addition of Z to the updated X. The notation $\overline{X}$ represents $1 \oplus X$. The final value of R_1 will be used as r_j in the following round.

[4] The maskVerif architecture used at the time of the security verification was based on [4]. The newer version described in [3] employs simulation-based rules for security checks and is thus unable to confirm the probing security of Dae-$\mathbb{S}\chi_5$ or the corresponding $\mathbb{S}$S-box.

[5] https://github.com/ascon/simpleserial-ascon

The Dae-$\mathbb{S}\chi_5$ gadget is designed to ensure *transition security* in addition to probing security. Thanks to the use of auxiliary random variables, the gadget is bijective: the 2^{11} possible inputs map to 2^{11} distinct outputs.

The GitHub implementation also linearly mixes the auxiliary randomness bits of the gadgets at the output of each round. However, this mixing does not appear to provide any additional benefit. The presented description already satisfies the requirements of Lemma 1 and consequently leads to a $d = 1$-secure masking of ASCON's permutation.

SM-$\mathbb{S}\chi_5$ Gadgets. Shahmirzadi and Moradi [30], using a computerized search approach, identified instances of $(n = 2, d = 1)$ and $(n = 3, d = 2)$ $\mathbb{S}\chi_5$ gadgets. These gadgets notably require no auxiliary randomness and provide resistance to glitches alongside probing security. We label these gadgets as SM1-$\mathbb{S}\chi_5$ and SM2-$\mathbb{S}\chi_5$, respectively.

The SM1-$\mathbb{S}\chi_5$ and SM2-$\mathbb{S}\chi_5$ gadgets are bijective and produce n-shared independent outputs when given n-shared independent inputs. Consequently, according to Lemma 1, combining SM1-$\mathbb{S}\chi_5$ with preceding and succeeding affine layers—as is the case for ASCON's S-box—preserves first-order security. Furthermore, the resulting S-box also satisfies the requirements of Lemma 1, thereby enabling deterministic masking for ASCON's permutation.

6 Higher-Order Probing Security

Ensuring higher-order probing security for deterministic masking in the presence of multiple probes is a complex task. It requires more than the bijectivity of the individual composing gadgets, as the interactions between them can introduce vulnerabilities. To illustrate, consider the serial combination of $d = 2$ secure gadgets shown in Figure 7. When this combination is instantiated with SM2-$\mathbb{S}\chi_5$ gadgets, it can be demonstrated that two probes—one placed at the input of the first $\mathbb{S}$S-box and the other at the output of the rth ($r \gg 1$) $\mathbb{S}$S-box—are statistically dependent on the underlying secrets. Consequently, this serial combination does not maintain $d = 2$ probing security.

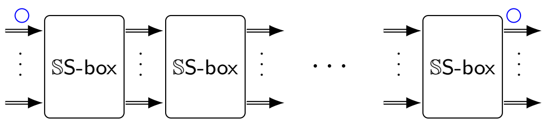

Figure 7: Serial combination of r gadgets. Blue circles indicate probe locations.

However, in bricklayer architectures, we can address the composition challenge by showing that the diffusion layer, up to a certain number of probes, acts as a refresh layer and blocks the adversary's attempt to gain benefit from spanning probes across different cipher rounds. This motivates taking a closer look at the structure of the diffusion layer.

6.1 Properties of the Diffusion Layer

A diffusion layer is an invertible linear transformation that operates on a b-bit state S. This transformation can be represented using matrix multiplication as $S_{1\times b} \leftarrow S_{1\times b}\mathbf{M}_{b\times b}$. The primary design goal of the diffusion layer is to combine state bits through the XOR operation. The cryptographic effectiveness of the diffusion layer is typically evaluated using two metrics: the linear and differential *branch numbers*, denoted respectively by $\mathcal{B}_l$ and $\mathcal{B}_d$.

Definition 4 (Branch Numbers [18])**.** The branch numbers $\mathcal{B}_l$ and $\mathcal{B}_d$ are defined as:

$$\mathcal{B}_l = \min_{S \neq 0^b} \left[\mathsf{wt}(S) + \mathsf{wt}(S\mathbf{M}^\top)\right], \quad \mathcal{B}_d = \min_{S \neq 0^b} \left[\mathsf{wt}(S) + \mathsf{wt}(S\mathbf{M})\right], \tag{10}$$

where $\mathbf{M}^\top$ is the transpose of $\mathbf{M}$, and wt computes the number of non-zero (active) elements in its input vector.

The metric $\mathcal{B}_d$ identifies the minimum Hamming weight of input/output pairs and specifies the least number of active output bits resulting from a single active input bit. Conversely, $\mathcal{B}_l$ determines the minimum Hamming weight of parity relations between inputs and outputs. For this work, the $\mathcal{B}_l$ metric is of greater relevance.

Cryptographic View. The metrics $\mathcal{B}_l$ and $\mathcal{B}_d$ are directly related to the strength of a cipher against linear and differential cryptanalysis, respectively. Higher values for these branch numbers indicate increased resistance to such attacks. However, achieving higher branch numbers typically comes at the cost of increased computational complexity.

Definition 5 (Complexity of Diffusion Layer [24])**.** The average number of XOR operations required to compute each output bit is defined as the complexity of the diffusion layer.

For Ascon, the linear and differential branch numbers of the diffusion layer are both 4 [19]. Specifically, each input bit affects three output bits, and no parity relation with fewer than four bits exists. This property can be verified by inspecting Equation (4). It is noteworthy that, for Ascon, the branch number is relatively low.[6] With the relatively low branch number, the diffusion layer's computational complexity is also very low: each output bit is computed using just 2 XOR operations.

Polynomial View. By representing the b binary random variables of the input state as $\{X_1, \ldots, X_b\}$, we can express the output bits of the diffusion layer as polynomials:

$$f_j = \bigoplus_{i=1}^{b} m_i^j X_i, \tag{11}$$

for $1 \le j \le b$, where the (binary) coefficients are elements of $\mathbf{M}$, defined as $m_i^j = \mathbf{M}[i,j]$. Identifying $\mathcal{B}_l$ then translates into finding a parity relation in the set

$$\mathcal{F} = \{X_1, \ldots, X_b, f_1, \ldots, f_b\}$$

with the smallest Hamming weight. A parity relation is a subset of $\mathcal{F}$ that sums to zero (see Section 2). More precisely, we show that:

$$\mathcal{B}_l = \min_{|\mathcal{C}|} \left[\bigoplus_{j \in \mathcal{C}} \mathcal{F}[j] = 0 \right], \tag{12}$$

where $\mathcal{F}[j]$ denotes the jth entry of the set $\mathcal{F}$, and $\mathcal{C}$ is a non-empty set of indices. To prove this claim, we write:

$$\begin{aligned} \mathcal{B}_l &= \min_{S \neq 0^b} \left[\mathsf{wt}(S) + \mathsf{wt}(S\mathbf{M}^\top) \right] \\ &= \min_{S \neq 0^b} \left[\mathsf{wt}(S \| S\mathbf{M}^\top) \right] \\ &= \min_{S \neq 0^b} \mathsf{wt}\left(S \times \left[\mathbf{I} \mid \mathbf{M}^\top \right] \right). \end{aligned}$$

[6] Since a single active input can create at most b active outputs, branch numbers are upper-bounded by $b+1$. In practice, this upper bound (or values close to it) is achievable for a special class of $\mathbf{M}$ matrices described by maximum distance separable (MDS) codes [18].

In coding theory, the outputs of $S \times \left[\mathbf{I} \mid \mathbf{M}^\top\right]$ are *codewords*, where $\mathbf{G} = \left[\mathbf{I}_{b\times b} \mid \mathbf{M}^\top\right]$ is the *generator* matrix of the code. Corresponding to $\mathbf{G}$, there is a *parity check* matrix $\mathbf{H} = [\mathbf{M} \mid \mathbf{I}_{b\times b}]$ [25], such that any codeword $C \in \{0,1\}^{2b}$ satisfies:

$$\mathbf{H}^\top C = \mathbf{0}_{b\times 1}.$$

Additionally, any $2b$-bit vector C satisfying $\mathbf{H}^\top C = \mathbf{0}$ is a codeword. This allows us to write:

$$\min_{S\neq 0^b} \mathsf{wt}(S\mathbf{G}) = \min_{\{C\neq 0^{2b}, \mathbf{H}^\top C = \mathbf{0}\}} \mathsf{wt}(C).$$

By simplifying further, we obtain:

$$\min_{\{C\neq 0^{2b}, \mathbf{H}^\top C = \mathbf{0}\}} \mathsf{wt}(C) = \min_{|\mathcal{C}|} \left[\bigoplus_{\mathrm{col}\in\mathcal{C}} \mathbf{H}[:\,,\mathrm{col}] = \mathbf{0} \right].$$

The notation $\mathbf{H}[:\,,\mathrm{col}]$ means a vector containing all the rows at the given column. We can also show that:

$$\mathcal{F}[2b-j] = [X_1, \ldots, X_b] \times \mathbf{H}[:\,, j],$$

which means the sum of columns $\mathbf{H}[:\,,\mathrm{col}]$ in some set $\mathrm{col} \in \mathcal{C}$ equals:

$$\bigoplus_{j\in\mathcal{C}} \mathcal{F}[2b-j] = \bigoplus_{j\in\mathcal{C}'} \mathcal{F}[j],$$

and this completes the proof.

6.1.1 Self-Composition of the Diffusion Layer

The evaluation of the diffusion mapping is just one part of the computations in a single cipher round. A cipher typically operates over multiple rounds, with non-linear operations interleaved between successive diffusion layers. To gain insight into how parity relations evolve with an increasing number of rounds, we temporarily disregard the presence of non-linear layers and study the composition of the diffusion transformation itself, as illustrated in Figure 8. The composition is computed as $(S\mathbf{M})\mathbf{M} = S\mathbf{M}^2$. In general, the composition over r rounds is directly represented by the matrix $\mathbf{M}^r$.

Definition 6 (Order of the Diffusion Layer)**.** The *order* of the diffusion layer is defined as the number of rounds r for which the layer composes with itself to produce the identity mapping. Formally, it is the smallest integer r such that $\mathbf{M}^r = \mathbf{I}$.

Composing the diffusion layer introduces new parity relations between the input S and the outputs $S\mathbf{M}^r$ (for varying r). Even before reaching the order, these new relations may have weights smaller than $\mathcal{B}_l$. More specifically, these relations arise in the $(r+1)b$-element set of polynomials:

$$\mathcal{F}^r = \{X_1, \ldots, X_b, f_1, \ldots, f_b, f_{b+1}, \ldots, f_{2b}, \ldots, f_{(r-1)b+1}, \ldots, f_{rb}\},$$

where, for $0 \leq \alpha < r$ and $1 \leq \beta \leq b$, $f_{\alpha b+\beta}$ is defined as:

$$f_{\alpha b+\beta} = \bigoplus_{i=1}^{b} \mathbf{M}^{\alpha+1}[\beta, i] X_i.$$

We denote the minimum weight of parity relations in the set $\mathcal{F}^r$ as $\mathcal{B}_l^r$. Note that $\mathcal{B}_l^r$ is a decreasing function of the number of rounds r, and at the order of the layer, it reaches the minimum value $\mathcal{B}_l^r = 2$.

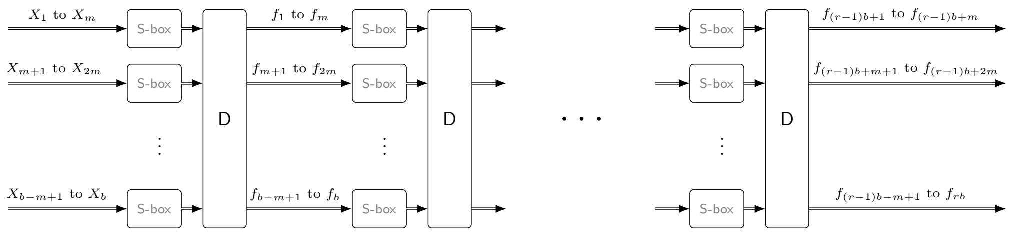

Figure 8: Outputs of the diffusion layer in r rounds expressed as polynomials of the input, where S-boxes are replaced with identity functions.

Example 4. Let $b = 15$ and the diffusion be defined by the following relation:

$$\mathbf{X} \leftarrow \mathbf{X} \oplus (\mathbf{X} \lll 1) \oplus (\mathbf{X} \lll 3), \tag{13}$$

where $\mathbf{X}$ is a vector of 15 bits. Using a computer program, we verified that this map is invertible, has an order of 15, and its $\mathcal{B}_l$ is 4.[7]

For this map, we constructed the set $\mathcal{F}^r$ and computed the corresponding $\mathcal{B}_l^r$ by exhaustively testing all subsets of $\mathcal{F}^r$ with fewer than $\mathcal{B}_l$ entries. We observed that, up to the order, the minimum weight of parity relations does not fall below the initial branch number. □

We note that for diffusion layers used in ciphers, the order is expected to be practically high. Moreover, $\mathcal{B}_l^r = \mathcal{B}_l$ is typically maintained for sufficiently high values of r in practical applications.

6.2 Joint Independence Across the Diffusion Layer

Consider a toy 3-bit linear transformation with the following input-output relation:

$$(X, Y, Z) \to (X \oplus Y, Y \oplus Z, X \oplus Y \oplus Z).$$

For this map, we have $\mathcal{B}_l = 3$. In this example, we can directly verify that the knowledge of a single input bit does not determine any of the output bits. More precisely, any two variables chosen from the inputs and outputs are jointly independent. The following lemma generalizes this observation.

Lemma 3. *Let $\mathcal{B}_l$ denote the branch number of a diffusion layer. Then, $\mathcal{B}_l - 1$ probes placed on the inputs and outputs of* Diffusion *are jointly independent. Alternatively, the collection of all probes on the inputs and outputs is $(\mathcal{B}_l - 1)$-jointly independent.*

Proof. Assume these probes (variables) are dependent. Then, there must exist a parity relation between the inputs and outputs with fewer than $\mathcal{B}_l$ terms. This contradicts the definition of $\mathcal{B}_l$. □

Lemma 3 can be extended to the composition of diffusion layers: any $\mathcal{B}_l^r - 1$ probes placed on the inputs and outputs of the diffusion layer in r rounds are jointly independent.

[7]This map can be expressed with *circulant matrices*, for which there is a straightforward approach to compute the order and inverse [26].

6.3 Our Higher-Order Composition Rule

Thus far, we have proved that the diffusion layer allows for some joint independence of output natives, provided that the inputs are unknown, uniform, and independent. However, this result, in its current form, is not particularly useful, as we cannot guarantee that the inputs always satisfy these requirements. For instance, some natives, such as IV and nonce (or their combinations in the initial rounds), are fixed or public. By transitioning to the masked environment and working with n-shared inputs (instead of natives), we can make a more practical and meaningful statement:

Lemma 4. *Any $\mathcal{B}_l - 1$ collection of n-sharings at the input and output of* $\mathbb{S}$Diffusion *are jointly independent. Alternatively, the collection of all input and output n-sharings are $(\mathcal{B}_l - 1)$-jointly independent n-sharings.*

Proof. Let $\mathcal{L} = \{\mathbf{X}_1, \ldots, \mathbf{X}_{\mathcal{B}_l - 1}\}$ be a collection of n-sharings. Their dependency would imply that for a subset $\mathcal{C} \subset \mathcal{L}$, we have: $\bigoplus_{\mathbf{X} \in \mathcal{C}} \mathbf{X} = 0$, which, by summing over all share indices, further implies: $\bigoplus_{\mathbf{X} \in \mathcal{C}} X = 0$, where the relation holds over the native random variables. However, by the definition of $\mathcal{B}_l$, no parity relation among the native random variables can have fewer than $\mathcal{B}_l$ terms. Hence, no dependency can exist within $\mathcal{L}$.

It is important to note that the independence of n-sharings (Definition 3) does not impose any restrictions on their corresponding natives. For instance, this lemma holds even if all the input natives are zero. $\square$

6.4 Our Higher-Order Composition Rule

We are now ready to introduce our main composition result for deterministic masking schemes in bricklayer architectures.

Theorem 1 (Composition Theorem)**.** *In a bricklayer architecture operating in multiple rounds with d-probing secure* $\mathbb{S}$S-box*es, as illustrated in Figure 9. If any set of l* $\mathbb{S}$S-box*es*

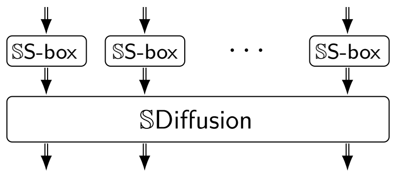

Figure 9: Bricklayer architecture with $\mathbb{S}$S-boxes and $\mathbb{S}$Diffusion.

receive a jointly independent n-shared set of inputs, the probing security order D of the combination satisfies:

$$\min\{d, l\} \leq D \leq d.$$

Proof. First, assume all probes are placed only on $\mathbb{S}$S-boxes, and no intermediates of $\mathbb{S}$Diffusion are probed. Let the probes spread over $\mathbb{S}$S-boxes in a set $\mathcal{T}$. If $|\mathcal{T}| \leq l$, the inputs to these $\mathbb{S}$S-boxes will remain statistically identical to their standalone situation. Thus, if each $\mathbb{S}$S-box has fewer than d probes, these probes will be independent of the secrets. For $l \geq d$, the value of l does not limit the probing security order of the composition, so $D = d$. For $l < d$, we have $l \leq D \leq d$.

Next, consider the case where intermediates of $\mathbb{S}$Diffusion are probed. Using an argument similar to probe propagation (see Section 4), we transfer these probes to the $\mathbb{S}$S-boxes. For example, consider a probe in $\mathbb{S}$Diffusion represented as $P = X \oplus Y$, where X and Y are outputs of two different $\mathbb{S}$S-boxes. If at least one of these $\mathbb{S}$S-boxes is in $\mathcal{T}$, we add one probe to X and one to Y and remove P. In this case, if $|\mathcal{T}|+1 \leq l$, then probes on $\mathbb{S}$S-boxes remain independent of secrets.

However, if neither $\mathbb{S}$S-box is probed, we add a probe to X, include the corresponding $\mathbb{S}$S-box in $\mathcal{T}$, and claim that X, being uniformly distributed, hides Y in $P = X \oplus Y$, provided $|\mathcal{T}| \leq l$. In general, each probe on $\mathbb{S}$Diffusion adds at most one $\mathbb{S}$S-box to $\mathcal{T}$, and every $\mathbb{S}$S-box in $\mathcal{T}$ corresponds to at least one probe. Hence, the probing security order satisfies:

$$\min\{d, l\} \leq D \leq d.$$

□

Using Lemma 4, for two consecutive rounds (containing one diffusion layer in between) with m-bit input/output S-boxes, we derive the following upper bound:

$$l \leq \frac{\mathcal{B}_l - 1}{m}.$$

For the special case of the Ascon cipher, where the m inputs of the S-box come from m distinct sets of bits (one from each of the 64-bit rows), we have a better bound:

$$l \leq \mathcal{B}_l - 1.$$

We expect that for the value of l, two consecutive rounds act as a bottleneck, and adding a limited number of rounds does not decrease the value of l further.

6.5 Application to Ascon

The Diffusion layer in Ascon is additive-affine, which simplifies the design of its masked counterpart at any order. For a $d = 2$ probing-secure $\mathbb{S}$S-box, we adopt the construction proposed in [22].[8] Since χ_5 is the only non-linear component of the S-box (illustrated in Figure 3), we focus on $\mathbb{S}\chi_5$ gadgets, which we designate as Gig-$\mathbb{S}\chi_5$.

Gig-$\mathbb{S}\chi_5$ Gadget. Let $\{A, B, C, D, E\}$ represent the input native variables. In our notation, the shares of a native variable such as A are denoted as $\{A_1, A_2, A_3\}$. Similar to the Dae-$\mathbb{S}\chi_5$ gadget, there is one auxiliary n-shared input native $R = 0$, with shares defined as follows:

$$R_2 = r_j, \quad R_3 = r_j \oplus r'_j, \quad R_1 = R_2 \oplus R_3,$$

where r_j and r'_j are randomly chosen bits for $1 \leq j \leq 64$. No online randomness is consumed during internal computations, and randomness is recycled across subsequent rounds. The computations for this gadget can be structured as follows:

$$\begin{array}{lll}
R_1 \oplus\!\!\leftarrow D_1E_2 \oplus \overline{D_1}E_1 \oplus D_1E_3, & R_2 \oplus\!\!\leftarrow D_2E_3 \oplus \overline{D_2}E_2 \oplus D_2E_1, & R_3 \oplus\!\!\leftarrow D_3\overline{E_1} \oplus D_3|E_3 \oplus D_3E_2, \\
E_1 \oplus\!\!\leftarrow A_1B_2 \oplus \overline{A_1}B_1 \oplus A_1B_3, & E_2 \oplus\!\!\leftarrow A_2B_3 \oplus \overline{A_2}B_2 \oplus A_2B_1, & E_3 \oplus\!\!\leftarrow A_3\overline{B_1} \oplus A_3|B_3 \oplus A_3B_2, \\
B_1 \oplus\!\!\leftarrow C_1D_2 \oplus \overline{C_1}D_1 \oplus C_1D_3, & B_2 \oplus\!\!\leftarrow C_2D_3 \oplus \overline{C_2}D_2 \oplus C_2D_1, & B_3 \oplus\!\!\leftarrow C_3\overline{D_1} \oplus C_3|D_3 \oplus C_3D_2, \\
D_1 \oplus\!\!\leftarrow E_1A_2 \oplus \overline{E_1}A_1 \oplus E_1A_3, & D_2 \oplus\!\!\leftarrow E_2A_3 \oplus \overline{E_2}A_2 \oplus E_2A_1, & D_3 \oplus\!\!\leftarrow E_3\overline{A_1} \oplus E_3|A_3 \oplus E_3A_2, \\
A_1 \oplus\!\!\leftarrow B_1C_2 \oplus \overline{B_1}C_1 \oplus B_1C_3, & A_2 \oplus\!\!\leftarrow B_2C_3 \oplus \overline{B_2}C_2 \oplus B_2C_1, & A_3 \oplus\!\!\leftarrow B_3\overline{C_1} \oplus B_3|C_3 \oplus B_3C_2, \\
C_1 \oplus\!\!\leftarrow R_1, & C_2 \oplus\!\!\leftarrow R_2, & C_3 \oplus\!\!\leftarrow R_3.
\end{array}$$

In this notation, $X|Y$ represents the OR operation between bits X and Y, and the sequence of operations (from left to right) is critical for maintaining probing security. This structure has been formally verified to provide $d = 2$ probing security for the $\mathbb{S}\chi_5$ gadget and its corresponding Ascon $\mathbb{S}$S-box.

The gadget has been verified to be bijective, mapping $2^{5\times3+2} = 2^{17}$ distinct inputs to 2^{17} distinct outputs. Additionally, the gadget and the associated $\mathbb{S}$S-box produce independent 3-shared outputs when provided with independent 3-shared inputs.

[8]We use the variant implemented at https://github.com/ascon/simpleserial-ascon.

Randomness Reuse. For the next round, the randomness variables at the output of 64 gadgets are recycled. Let $\mathbf{R}_2$ represent the collection of R_2 variables at the output of the gadgets, and $\mathbf{R}_3$ denote the corresponding R_3 random variables. We propose a recycling process as follows:

$$\mathbf{R}_2 \leftarrow \mathbf{R}_2 \oplus (\mathbf{R}_2 \lll 7) \oplus (\mathbf{R}_2 \lll 48), \mathbf{R}_3 \leftarrow \mathbf{R}_3 \oplus (\mathbf{R}_3 \lll 19) \oplus (\mathbf{R}_3 \lll 34), \mathbf{R}_1 \leftarrow \mathbf{R}_2 \oplus \mathbf{R}_3.$$

This transformation is bijective and preserves the 64×2 bits of input randomness. The mappings are similar to the diffusion layer and maintain their branch number $\mathcal{B}_l = 4$ up to their order. For the selected rotation amounts, the order (Definition 6) is 64, which represents the maximum possible value (see [26] for the properties of these maps).[9]

Linear Representation of Output Shares. Upon closer inspection of Gig-$\mathbb{S}\chi_5$, we observe that at a fixed secret state, each of the output shares can be expressed as a linear function of the input shares. For instance, for the shares of E at the output of the gadget, we have:

$$E_1 \leftarrow E_1 \oplus A_1 B \oplus B_1, \quad E_2 \leftarrow E_2 \oplus A_2 B \oplus B_2, \quad E_3 \leftarrow E_3 \oplus A_3 B \oplus B_3.$$

This linear representation similarly applies to the shares of R at the output. We leverage this linearity to track the independence of inputs in subsequent rounds, as required by Theorem 1.

Let us denote all the $nb = 3 \times (5+1) \times 64$ shares at the cipher's input as the set of variables $\{X_1, X_2, \ldots, X_{nb}\}$. By following the linearity of mappings, we can create polynomial descriptions for input shares at different rounds. Let Y_t^r, for $1 \leq t \leq nb$, be an input share at round r. For this share, we can write:

$$Y_t^r = \bigoplus_{i \in \mathcal{I}_t^r} X_i \oplus \bigoplus_{j \in \mathcal{J}_t^r} X_j f_{j,t}^r(S), \tag{14}$$

where $\mathcal{I}_t^r \cap \mathcal{J}_t^r = \emptyset$, and S represents the initial (unshared) state of the cipher. The functions $f_{j,t}^r$ are nonlinear, and we track the maximum number of times native variables are multiplied within them. We have constructed such linear descriptions for all shares across different numbers of rounds.

Considering the symmetry of the cipher, to investigate the second-order probing security of the masking, we demonstrate that the set of n-shared inputs to the first $\mathbb{S}$S-box and the set of n-shared inputs to all other $\mathbb{S}$S-boxes in subsequent rounds are independent (as per Definition 3). To this end, we reorder the polynomials derived in (14) and isolate variables X_1 to X_{18}, which correspond to the inputs of the first $\mathbb{S}$S-box.

We verify that for each $\mathbb{S}$S-box, the set of input shares are blinded by distinct initial shares X_i. More concretely, to show that a share Y_t^r is independent of X_1 to X_{18}, we identify an X_i in its polynomial description such that $i \in \mathcal{I}_t^r$ and $i \notin \{1, 2, \ldots, 18\}$. To prove independence for multiple Y_t^r shares, we demonstrate that each is blinded with a distinct X_i term, following a method similar to the probe elimination approach in Section 4.

It is important to note that up to $n-1$ shares of each initial native are random. For effective blinding, care must be taken not to use all n shares of a single native input.

We have applied the described method for $r \leq 5$ rounds and verified the independence of inputs to the $\mathbb{S}$S-boxes from X_1 to X_{18}. For rounds $r > 5$, the sets $\mathcal{I}^r$ are empty, and we have:

$$Y_t^r = \bigoplus_{j \in \mathcal{J}_t^r} X_j f_{j,t}^r(S), \tag{15}$$

[9]The transformation implemented in the GitHub code for recycling randomness is similar but simpler. It computes $\mathbf{R}_1 \leftarrow (\mathbf{R}_2 \lll 7) \oplus (\mathbf{R}_3 \lll 13)$ for the first and second 32 gadgets separately. This alternative transformation is bijective with a branch number of 2 and an order of 16.

where $f^r_{j,t}$ is expected to be of degree r. To continue the blinding approach, we require at least one non-zero $f^r_{j,t}(S)$. In such cases, the corresponding X_j with $j \notin \{1, 2, \ldots, 18\}$ blinds Y^r_t. At any fixed state S, there is a very high probability that at least one of the $f^r_{j,t}(S)$ bits, for $19 \leq j \leq nb$, is non-zero.

7 Conclusion

This work underscores the potential of ciphers to achieve higher-order probing security through deterministic masking, solely leveraging offline randomness encapsulated in the initial input shares. Our analysis demonstrates how a diffusion layer, attributed by linear branch number $\mathcal{B}_l$, effectively mixes inputs to generate seemingly fresh n-sharings for the subsequent round. Crucially, we establish that despite the absence of fresh randomness in subsequent rounds, the limited probing capability of an adversary ensures that this lack of freshness remains indistinguishable. These results pave the way for resource-efficient masking schemes that balance theoretical rigor with practical feasibility, advancing the state of side-channel attack countermeasures.

References

[1] SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015-08-04 2015). https://doi.org/https://doi.org/10.6028/NIST.FIPS.202

[2] Ascon-Based Lightweight Cryptography Standards for Constrained Devices, (NIST initial public draft) (2024), https://csrc.nist.gov/pubs/sp/800/232/ipd

[3] Barthe, G., Belaïd, S., Cassiers, G., Fouque, P., Grégoire, B., Standaert, F.: maskVerif: Automated Verification of Higher-Order Masking in Presence of Physical Defaults. In: Sako, K., Schneider, S.A., Ryan, P.Y.A. (eds.) Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11735, pp. 300–318. Springer (2019). https://doi.org/10.1007/978-3-030-29959-0_15, https://doi.org/10.1007/978-3-030-29959-0_15

[4] Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y.: Verified Proofs of Higher-Order Masking. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. pp. 457–485. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)

[5] Barthe, G., Belaïd, S., Dupressoir, F., Fouque, P.A., Grégoire, B., Strub, P.Y., Zucchini, R.: Strong Non-Interference and Type-Directed Higher-Order Masking. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 116–129 (2016)

[6] Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness Complexity of Private Circuits for Multiplication. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 616–648. Springer (2016)

[7] Bellizia, D., Bronchain, O., Cassiers, G., Grosso, V., Guo, C., Momin, C., Pereira, O., Peters, T., Standaert, F.X.: Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography A Practical Guide Through the Leakage-Resistance Jungle.

In: Crypto. Santa Barbabra, United States (Aug 2020), https://hal.science/hal-02901380

[8] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge Functions. In: Ecrypt Hash Workshop 2007 (May 2007)

[9] Beyne, T., Dhooghe, S., Zhang, Z.: Cryptanalysis of Masked Ciphers: A Not So Random Idea. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 817–850. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_27, https://doi.org/10.1007/978-3-030-64837-4_27

[10] Cassiers, G., Masure, L., Momin, C., Moos, T., Moradi, A., Standaert, F.X.: Randomness Generation for Secure Hardware Masking – Unrolled Trivium to the Rescue. IACR Communications in Cryptology **1**(2) (2024). https://doi.org/10.62056/akdkp2fgx

[11] Cassiers, G., Standaert, F.X.: Trivially and Efficiently Composing Masked Gadgets With Probe Isolating Non-Interference. IEEE Transactions on Information Forensics and Security pp. 2542–2555 (2020). https://doi.org/10.1109/TIFS.2020.2971153

[12] Cassiers, G., Standaert, F.X.: Provably Secure Hardware Masking in the Transition- and Glitch-Robust Probing Model: Better Safe than Sorry. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 136–158 (Feb 2021). https://doi.org/10.46586/tches.v2021.i2.136-158, https://tches.iacr.org/index.php/TCHES/article/view/8790

[13] Coron, J.S.: Formal Verification of Side-Channel Countermeasures via Elementary Circuit Transformations. In: Preneel, B., Vercauteren, F. (eds.) Applied Cryptography and Network Security. pp. 65–82. Springer International Publishing, Cham (2018)

[14] Coron, J.S., Greuet, A., Zeitoun, R.: Side-Channel Masking with Pseudo-Random Generator. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology – EUROCRYPT 2020. pp. 342–375 (2020)

[15] Coron, J.S., Prouff, E., Rivain, M., Roche, T.: Higher-Order Side Channel Security and Mask Refreshing. In: Moriai, S. (ed.) Fast Software Encryption. pp. 410–424. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)

[16] Daemen, J.: Cipher and Hash Function Design Strategies based on linear and differential cryptanalysis. Ph.D. thesis, Katholieke Universiteit Leuven (1995), https://cs.ru.nl/~joan/papers/JDA_Thesis_1995.pdf

[17] Daemen, J., Dobraunig, C., Eichlseder, M., Groß, H., Mendel, F., Primas, R.: Protecting against Statistical Ineffective Fault Attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 508–543 (2020). https://doi.org/10.13154/TCHES.V2020.I3.508-543, https://doi.org/10.13154/tches.v2020.i3.508-543

[18] Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020). https://doi.org/10.1007/978-3-662-60769-5, https://doi.org/10.1007/978-3-662-60769-5

[19] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight Authenticated Encryption and Hashing. Journal of Cryptology **34** (2021). https://doi.org/10.1007/s00145-021-09398-9

[20] Faust, S., Grosso, V., Merino Del Pozo, S., Paglialonga, C., Standaert, F.X.: Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(3), 89–120 (Aug 2018). https://doi.org/10.13154/tches.v2018.i3.89-120, https://tches.iacr.org/index.php/TCHES/article/view/7270

[21] Faust, S., Paglialonga, C., Schneider, T.: Amortizing Randomness Complexity in Private Circuits. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 781–810. Springer International Publishing (2017)

[22] Gigerl, B., Mendel, F., Schläffer, M., Primas, R.: Efficient Second-Order Masked Software Implementations of Ascon in Theory and Practice, NIST LWC Workshop (2023)

[23] Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003. pp. 463–481. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)

[24] Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. IACR Trans. Symmetric Cryptol. **2017**(4), 130–168 (2017). https://doi.org/10.13154/TOSC.V2017.I4.130-168, https://doi.org/10.13154/tosc.v2017.i4.130-168

[25] Lin, S., Costello, D.J.: Error Control Coding, Second Edition. Prentice-Hall, Inc., USA (2004)

[26] Liu, M., Sim, S.M.: Lightweight MDS Generalized Circulant Matrices. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 101–120. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_6, https://doi.org/10.1007/978-3-662-52993-5_6

[27] Nikova, S., Rechberger, C., Rijmen, V.: Threshold Implementations Against Side-Channel Attacks and Glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security. pp. 529–545. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[28] Piccione, E., Andreoli, S., Budaghyan, L., Carlet, C., Dhooghe, S., Nikova, S., Petrides, G., Rijmen, V.: An Optimal Universal Construction for the Threshold Implementation of Bijective S-Boxes. IEEE Trans. Inf. Theory **69**(10), 6700–6710 (2023). https://doi.org/10.1109/TIT.2023.3287534, https://doi.org/10.1109/TIT.2023.3287534

[29] Reparaz, O.: A note on the security of Higher-Order Threshold Implementations. Cryptology ePrint Archive, Paper 2015/001 (2015), https://eprint.iacr.org/2015/001

[30] Shahmirzadi, A.R., Moradi, A.: Second-Order SCA Security with almost no Fresh Randomness. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 708–755 (2021)