Clustering Approach for Higher-Order Deterministic Masking

Vahid Jahandideh, Jan Schoone, and Lejla Batina

Radboud University, Nijmegen, The Netherlands
v.jahandideh@cs.ru.nl,jan.schoone@ru.nl,lejla@cs.ru.nl

Abstract. We present a novel scheme for securely computing the AND operation, without requiring additional online randomness. Building on the work of Nikova et al., our construction extends security beyond first order while ensuring a uniform output distribution and resilience against glitches up to a specified threshold. This result addresses a longstanding open problem in side-channel-resistant masking schemes.
Our approach is based on a new method of share clustering, inspired by finite affine geometry, enabling simultaneous consideration of both security and uniformity. Furthermore, we demonstrate how this clustering-based framework can be applied to higher-order protection of ciphers like Ascon under a fully deterministic masking regime. By eliminating the need for online randomness within the protected circuit, our work expands the practical scope of efficient and higher-order masking schemes for resource constraint applications.

Keywords: Boolean masking · Cryptography · Finite geometry · Multiplication gadgets · Side-channel analysis

1 Introduction

Side-Channel Threat. Implementations of cryptographic algorithms often leak information through side channels, making them vulnerable to *Side-Channel Attacks* (SCAs). These attacks pose a significant risk even when the underlying schemes are secure in a black-box model. A prominent countermeasure is *masking*, which splits sensitive variables (e.g., keys, plaintexts) into n *shares* using *offline* randomness, and then performs computations on these shares. Whenever necessary, *online* randomness is introduced to prevent unintended combination of shares [4, 8]. Since masking heavily depends on the availability and quality of randomness, generating large amounts of fresh randomness—especially under side-channel threats [11, 14]—is challenging and expensive. Consequently, extensive research focuses on reducing the reliance on online randomness [6,24,37]. An extreme case, and the one we pursue, is *deterministic* masking, which requires no online randomness [28].

Deterministic Masking. Masking is implemented by replacing basic operations (e.g., AND, XOR) with small circuits called *gadgets*, which take n-shared inputs

and produce n-shared outputs. Naively composing these gadgets is functionally correct but can inadvertently mix shares in a way that poses flaws in security [35]. A common countermeasure is to insert *refresh gadgets* between gadgets, but they consume online randomness and thus are avoided in deterministic masking. Consequently, two challenges arise: (1) designing deterministic gadgets that rely only on the randomness contained in the shares, and (2) securely composing them without refresh gadgets.

Deterministic Gadgets. The design of deterministic gadgets has been explored in several papers. Nikova et al. [31] introduced a deterministic gadget for the AND operation, while Daemen et al. [18] presented a first-order probing-secure gadget for the map χ_5 (the source of nonlinearity in several ciphers). Although the latter solely uses input shares for masking security, it still requires auxiliary randomness. Later, Shamirzadi and Moradi [37] proposed a deterministic masking scheme for χ_5 that avoids auxiliary randomness entirely and mitigates glitch-based leakages. Moving beyond simple operations, Piccione et al. [33] developed a systematic method for building deterministic gadgets for bijective S-boxes with first-order probing security.

Deterministic Composition. A more challenging task is to securely compose deterministic gadgets; even combining two individually secure gadgets may reduce the overall security [15,35]. However, for round-based ciphers such as Ascon [22], Jahandideh et al. [28] provided a composition rule. As depicted in Figure 1, a typical round function contains an S-box layer and a diffusion layer. Their main insight is that this diffusion layer, by mixing the shares of multiple gadgets, behaves like a refresh gadget and thus enables secure composition of rounds without online randomness.

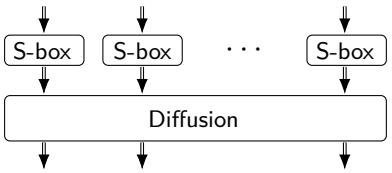

Fig. 1: Round transformation in a cipher.

Probing Security. Ishai et al. [27] formalized the notion of a probing adversary who can inspect up to d intermediate values during computation. A masking scheme is called *d-probing secure* if any set of d or fewer probes reveals no information about the secrets.

Glitch-Extended Probing. In hardware implementations, transient effects known as *glitches* can leak additional information [30]. To address this, Faust et al.

[23] proposed *glitch-extended probing*, wherein a single probe of the form $P = \mathsf{F}(X_i, Y_j, \ldots)$ reveals all inputs to F within a clock cycle. A standard mitigation involves inserting *synchronization layers* to limit how many inputs can interact at once [10, 23], thereby reducing glitch-induced leakage. Notably, this approach does not depend on online randomness.

1.1 An Open Problem

In masked implementations, most of online randomness is consumed within nonlinear operations. The simplest such operation is the AND function on two variables. As a fundamental result, Ishai et al. [27] presented a masked AND gadget that uses $n(n-1)/2$ random values to achieve $d = \lfloor (n-1)/2 \rfloor$ probing security. Subsequent works have increased the achievable security to $d = n - 1$ [36] while reducing randomness requirements to $n^2/4 + \mathcal{O}(n)$ [6]. Meanwhile, Belaïd et al. [6] proved it is impossible to attain $d = n - 1$ without online randomness. Their result, however, does not exclude the possibility of achieving $d < n - 1$ deterministically: Nikova et al. [31] provided a $(n = 4, d = 1)$ deterministic AND gadget. Yet, whether deterministic AND gadgets exist for higher (n, d) remains an open problem.

1.2 Our Contribution

We address the longstanding open problem of constructing a deterministic masked AND operation that achieves $d > 1$ probing security while preserving glitch resistance and output uniformity—all without requiring online (fresh) randomness. Unlike prior approaches that rely on exhaustive search, our method systematically clusters shares using finite affine geometry.

In terms of computational complexity, our multiplication gadget requires n field multiplications and guarantees $d = \sqrt{n} - 1$ probing security. This is comparable to state-of-the-art schemes that achieve $d = n - 1$ probing security at the cost of n^2 multiplications and $n^2/4 + \mathcal{O}(n)$ online randomness [6]. For a detailed comparison of our AND algorithms, see Table 1. Note that in our case n should be a square of a prime power. Hence, concerning probing security orders below 10, our approach works for the following values of (n, d): $(4, 1), (9, 2), (16, 3), (25, 4), (49, 6), (64, 7), (81, 8)$.

In addition, we propose a deterministic masking circuit for the map χ_m that achieves up to $d = \sqrt{n} + 1$ probing security. We verify the gadget's bijectivity for $n = 4$ by exhaustive enumeration. This $n = 4$ case is particularly relevant because it yields third-order security and applies to χ_m in Ascon [2, 22] (the winner of the NIST Lightweight Cryptography competition) as well as other widely used cryptographic primitives like SHA-3 [1] and Xoodoo [19]. Moreover, using recent results by Jahandideh et al. [28], we analyze the probing security of a masked Ascon implementation that incorporates our proposed χ_m gadget.

For a comparison of our χ_m masking with existing approaches, see Table 2. Table 3 contrasts deterministic methods for masking Ascon.

AND Algorithm	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
ISW [27]	n	$\frac{n(n-1)}{2}$	$\lfloor \frac{n-1}{2} \rfloor$	✓	$2(n^2-n)$	n^2	×	✓
Rivain and Prouff [36]	n	$\frac{n(n-1)}{2}$	$n-1$	✓	$2(n^2-n)$	n^2	×	✓
Belaïd et al. [6]	n	$\approx \frac{n^2+2n}{4}$	$n-1$	✓	$\approx \frac{7n^2}{4}$	n^2	×	✓
Nikova et al. [31]	4	0	1	✓	24	4	✓	×
De Cnudde et al. [21]	6	7	2	✓	29	36	✓	×
$\mathbb{S}$AND-DN [This work]	n	0	$\sqrt{n}-1$	×	$2(n-\sqrt{n})$	n	✓	×
$\mathbb{S}$AND-DU [This work]	n	0	$\sqrt{n}-1$	✓	$2\sqrt{n}(n-1)$	n	✓	×

Table 1: Comparison of masking algorithms used for secure computation of AND operation. The description of the columns is as follows: (A): number of shares; (B): required online randomness; (C): probing security order; (D): uniform output; (E): number of XORs; (F): number of field multiplications; (G): glitch resistant; (H): does not require input independence.

Candidate	(A)	(B)	(C)	(D)	(E)	(F)	(G)
Unprotected [16]	1	0	0	odd m	✓	$2m$	m
Bilgin et al. [9]	4	0	1	5	✓	80	30
Daemen et al. [18]	2	1 bit	1	5	×	22	20
Shahmirzadi and Moradi [37]	3	0	2	5	✓	80	45
$\mathbb{S}_{\chi_m}$ [This work]	4	0	3	5 & 3	✓	85 ($m=5$)	80 ($m=5$)

Table 2: Comparison of dedicated masked algorithms for the maps χ_m. The description of the columns is as follows: (A): sharing order n; (B): required online randomness; (C): probing security order; (D): values of m; (E): uniform output (i.e., mn-bit permutation); (F): number of XORs; (G): number of ANDs.

Candidate	(A)	(B)	(C)	(D)
Daemen et al. [18]	2	✓	1	Verified for one round (Using maskVerif [3])
Gigerl et al. [26]	3	✓	2	Verified for one round (Using CoCo [25])
This work	4	✓	3	Proved for two rounds, heuristic reasoning for more rounds

Table 3: Comparison of deterministic masking of Ascon. The description of the columns is as follows: (A): sharing order. (B): not requiring online randomness (C): probing security order. (D): type of the security proof.

A central component of our approach is a novel grouping strategy that simultaneously ensures both uniformity and security. Founded in finite geometry, this technique has the potential to benefit other masking schemes beyond our specific construction. While Petrides [32] also explored a grouping-based approach,

their method was limited to achieving second-order probing security, whereas our approach generalizes beyond this constraint.

Outline. Section 3 introduces our clustering approach and its properties. Building on these insights, Sections 4 and 5 details the construction of multiplication and Toffoli gadgets. Section 6 applies these results to the map χ_m, and Section 7 outlines our proposal for a third-order deterministic masking of Ascon.

2 Preliminaries

Notation. Uppercase letters (e.g., X) denote Random Variables (RVs), while lowercase letters (e.g., x) represent parameters and realizations of RVs. Boldface letters (e.g., $\mathbf{X}$) indicate a list of RVs, such as a list of shares of a secret, or a set of probes, and blackboard bold letters (e.g., $\mathbb{X}$) denote domains. The probability that an RV X takes a value x is denoted as $\Pr(X{=}x)$.

In this paper, circuits and functions are denoted with a sans-serif font (e.g., X). To refer to the masked counterpart of an algorithm, we append $\mathbb{S}$ to its name (e.g., $\mathbb{S}\mathsf{X}$). The notation $|\mathbf{X}|$ denotes the length of a list $\mathbf{X}$, and a list of shares with indices in $\mathbf{L}$ is represented as $\mathbf{X}|_{\mathbf{L}}$. Given a list $\mathbf{X}$, the i-th element in that list is denoted by $\mathbf{X}[i]$.

2.1 Boolean Masking

Let X be a random variable in the finite field $\mathbb{F}_{2^u}$. To achieve a probing security order $d = n - 1$, Boolean masking secret-shares X into an n-tuple (or list) of *shares*, denoted by

$$\mathbf{X} = [X_0, X_1, \ldots, X_{n-1}].$$

Each share X_i is randomly chosen from $\mathbb{F}_{2^u}$ such that their sum (under $\oplus$, the addition in $\mathbb{F}_{2^u}$) reconstructs X, i.e.,

$$X = \oplus_{i=0}^{n-1} X_i.$$

We call $\mathbf{X}$ an n-sharing of X. The parameter u denotes the bit-width of the field, with $u = 1$ for the binary case $\mathbb{F}_2$.

For a circuit C, Boolean masking is applied by n-sharing each input variable and replacing every gate with its corresponding masked counterpart (commonly called a *gadget*). The variables in C can be classified as either *secret* or *native*, where even publicly known values (e.g., nonces or IVs) are still n-shared for consistency and security.

Gadgets. Consider a gate $Y = \mathsf{G}(X)$ corresponding to an affine operation, such as XOR or NOT. Constructing the corresponding gadget, denoted $\mathbb{S}\mathsf{G}$, is

straightforward. To obtain an n-sharing $\mathbf{Y} = [Y_0, \ldots, Y_{n-1}]$ from an n-sharing $\mathbf{X} = [X_0, \ldots, X_{n-1}]$, we can define

$$Y_i = \begin{cases} \mathsf{G}(X_i), & \text{if } i = 0, \\ \mathsf{G}(X_i) \oplus \mathsf{G}(0), & \text{if } 1 \le i \le n-1. \end{cases}$$

For other gates, such as AND, securely computing the output shares is more challenging. Although several constructions for arbitrary-order $\mathbb{S}\mathsf{AND}$ exist [7,12, 27,39], these generally require online randomness. The only known *deterministic* $\mathbb{S}\mathsf{AND}$ is the scheme by Nikova et al. [31] for parameters $(n, d, u) = (4, 1, 1)$. In this scheme, the gadget takes two 4-sharings $\mathbf{X} = [X_0, X_1, X_2, X_3]$ and $\mathbf{Y} = [Y_0, Y_1, Y_2, Y_3]$ of secret random variables X and Y, and produces a 4-sharing $\mathbf{Z} = [Z_0, Z_1, Z_2, Z_3]$ such that $Z = XY$, where Z is the secret shared by $\mathbf{Z}$. The output shares are computed as follows:

$$\begin{cases} Z_0 = (1 \oplus X_2 \oplus X_3)(1 \oplus Y_1 \oplus Y_2) \oplus Y_3 \oplus X_1, \\ Z_1 = (1 \oplus X_0 \oplus X_2)(1 \oplus Y_0 \oplus Y_3) \oplus Y_2 \oplus X_3, \\ Z_2 = (X_1 \oplus X_3)(Y_0 \oplus Y_3) \oplus Y_1 \oplus X_1, \\ Z_3 = (X_0 \oplus X_1)(Y_1 \oplus Y_2) \oplus Y_0 \oplus X_0. \end{cases}$$

This gadget satisfies the following key properties:

- **Correctness:** For all $2^4 \times 2^4$ possible inputs,

$$\bigoplus_{i=0}^{3} Z_i \;=\; \left(\bigoplus_{i=0}^{3} X_i\right) \cdot \left(\bigoplus_{i=0}^{3} Y_i\right).$$

- **Incompleteness:** Any (glitch-extended) probe on intermediate variables (e.g., on Z_i) depends on only an *incomplete* subset of the input shares.
- **Uniformity:** For inputs $(X = x, Y = y)$, the 4-sharing $\mathbf{Z}$ of xy is *uniform*; that is, each tuple $[Z_0, Z_1, Z_2, Z_3]$ summing to xy has the same number of preimages.

2.2 Simulation and Probe Propagation

We briefly review the *simulation* approach and its limitations for deterministic masking schemes (see [28] for more details). Note that the discussion in this subsection applies only to standard probes (i.e., not glitch-extended).

Simulation Technique. Directly verifying d-probing security becomes infeasible for large sharing orders n. In the non-extended probe setting, Ishai et al. [27] introduced the *simulation* approach: instead of checking all d-tuples of intermediate values, one shows that any d probes can be reproduced (simulated) using fewer than n input shares. The core idea is that randomness can *hide* intermediate values. Let V_1 and V_2 be intermediate variables, and let R be an online randomness variable:

– If $P = R \oplus V_1$ and R is not used by other probes, then P can be replaced by an independent random variable R'. In this case, we say that R *blinds* P.
– If $P = V_1 \oplus V_2$ or $P = V_1 V_2$, both V_1 and V_2 are required to simulate P.

Definition 1 (Probe Propagation). *If simulating a probe P requires an intermediate variable V, we say that P* propagates *to V, and V is thus a propagated probe.*

In general, probes propagate backward toward the inputs unless they are *blocked* by online randomness. Coron [13] formalized this in a *probe elimination* rule: if $P_i = R \oplus \mathsf{F}$ (here, F accepts intermediates as inputs) and R is not used elsewhere, then P_i can be removed from the set of probed values and replaced by fresh randomness R'.

Example 1. Consider $n = 2$ with input shares $\{A_1, A_2\}$. Suppose the probed intermediates are:

$$P_1 = R_1 \oplus (A_1 A_2) \oplus R_2, \quad P_2 = R_2 \oplus A_1 \oplus A_2, \quad P_3 = A_1.$$

– Remove P_1, since $P_1 = R_1 \oplus \mathsf{F}$ and R_1 does not appear in other probes.
– Next, remove P_2, which similarly depends on a randomness R_2 not used elsewhere in the updated list of probes.

Only $P_3 = A_1$ remains. Hence, simulating P_3 requires only one input share. □

Composition Rules. Barthe et al. [4] introduced *Strong Non-Interference* (SNI), a property ensuring that simulating up to a certain number of outputs does not require any input shares. This prevents probe propagation beyond the gadget itself. Cassiers and Standaert [12] refined this notion by introducing *Probe Isolating Non-Interference* (PINI), which guarantees secure composition without extra refresh steps.

Randomness Requirement. Both probe elimination and SNI/PINI rely on online randomness. Without it, outputs become deterministic functions of the inputs, making simulation without input shares impossible.

Limitations of Simulation-Based Security. Probing security requires that any set of d probes reveals no information about the secrets. This condition can be more general than simulation. For instance, consider

$$P = A_1 \oplus (B_1 B_2),$$

where $\{A_1, A_2\}$ are the shares of a secret A and $\{B_1, B_2\}$ are the shares of a secret B. The sets of shares of A and B are linearly independent. A direct analysis shows that P is independent of A and B. However, a standard simulation argument would require $\{A_1, B_1, B_2\}$ to simulate P, suggesting that P is dependent on B. Hence, while simulation greatly simplifies proofs, it can be more restrictive than general probing security.

Moreover, this example illustrates how probing security can be maintained by *hiding* secret-dependent intermediates (in this case, $B_1 B_2$) by shares of another input (in this case, A_1). A key prerequisite is that the n-sharings of different inputs be mutually independent [28].

Probe Elimination in Deterministic Masking. We extend the probe elimination approach by leveraging input shares. Specifically, if a probe P_i satisfies

$$P_i = A_i \oplus \mathsf{F},$$

and the following conditions hold:

1. The share A_i does not appear in any other probes;
2. The set of probes does not contain all shares of A;

then A_i effectively *hides* F, allowing P_i to be removed from the probe set.

2.3 Incompleteness Implies Glitch Resistance

We now shift our focus to glitch-extended probes. Recall that a glitch-extended probe reveals *all* intermediate values used in its computation. Consequently, if these recovered intermediates form an *incomplete* set of shares for each input, then the glitches do not compromise the underlying secret.

We say that a masked gadget is *d-order glitch resistant* if any collection of d glitch-extended probes can be computed using at most $n-1$ shares of each secret input.

Glitch Resistance Order vs. Probing Security Order. Glitch resistance and probing security are *not* equivalent. Probing security requires that d probed values be independent of the secrets, whereas d-order glitch resistance requires that the probed values be computable without the complete set of shares for any secret.

Example 2. Let $\mathbf{X} = [X_0, X_1, X_2]$ and $\mathbf{Y} = [Y_0, Y_1, Y_2]$ be two independent sharings (with $n = 3$). Define:

$$P_0 = (X_0 \oplus X_1)(Y_0 \oplus Y_1) \quad \text{and} \quad P_1 = (X_1 \oplus X_2)(Y_1 \oplus Y_2).$$

From a standard probing perspective, the set $\mathbf{P} = \{P_0, P_1\}$ is statistically independent of

$$X = X_0 \oplus X_1 \oplus X_2 \quad \text{and} \quad Y = Y_0 \oplus Y_1 \oplus Y_2.$$

If these probes P_0, P_1 are the only probes the adversary has access to, the probing security would be 2.

However, if these probes are glitch-extended, they are not *incomplete*: computing P_0 and P_1 together requires all shares of X and Y. Hence, the gadget's glitch resistance order is less than 2. On the other hand, each individual probe P_0 or P_1 depends on only two input shares (i.e., at most $n-1$). Thus, the glitch resistance order is 1.

In general, the glitch resistance order is at most the probing security order.

2.4 Composition in Deterministic Masking

Limited Probes and Indistinguishability of Reused Randomness. When an adversary has only a limited number of probes, recycled randomness may be indistinguishable from fresh randomness. For instance, with random bits R_1 and R_2, an adversary holding two probes cannot distinguish between

$$\{R_1, R_2, R_1 \oplus R_2\} \quad \text{and} \quad \{R_1, R_2, R_3\},$$

where R_3 is an independent random bit. Jahandideh et al. [28] observed a similar effect in round-based ciphers (e.g., ASCON) due to diffusion layers.

Bricklayer Design of Ciphers. Modern ciphers often run in multiple rounds, each comprising a non-linear *confusion* layer (implemented by parallel S-box functions) and a linear *diffusion* layer. Daemen and Rijmen [20] coined the term *bricklayer* for architectures that arrange these S-box-based components in parallel. Figure 2 shows a masked bricklayer design without using any refresh gadgets.

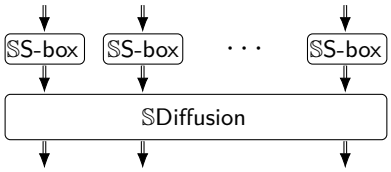

Fig. 2: Deterministic masking for the bricklayer architecture.

First-Order Probing Security. At sharing order n, if every gadget in a deterministic masked circuit is first-order probing secure, then the entire circuit remains first-order secure *provided* each gadget's inputs are n-shared with *jointly independent* sharings (Definition 2) [28]. If these gadgets are also bijective, the overall bricklayer structure is bijective, ensuring at least first-order security. Higher orders require additional safeguards from the diffusion layer, as we explain next.

Definition 2 (Independence of n-Sharings [28]). *A collection of n-sharings as $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ is called* independent *if, for any selection of $n-1$ shares from each $\mathbf{X}_i$, the resulting $(m \times (n-1))$ shares are jointly independent.*

Properties of the Diffusion Layer. A diffusion layer is an invertible linear transformation on a b-bit state S, often described by

$$S \leftarrow S\,\mathbf{M},$$

where $\mathbf{M}$ is a $b \times b$ matrix over $\mathbb{F}_2$. Its *linear branch number* $\mathcal{B}_l$ quantifies how bits mix via XOR:

Definition 3 (Linear Branch Number [20]). *Let* $\mathbf{M}^\top$ *be the transpose of a matrix* $\mathbf{M}$*. Then*

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

where $\mathsf{wt}(\cdot)$ *is the number of non-zero bits in its input.*

In Ascon, the diffusion layer has $\mathcal{B}_l = 4$ [22].

Lemma 1 ([28]). *If* $\mathcal{B}_l$ *is the linear branch number of the diffusion layer* Diffusion*, then placing* $\mathcal{B}_l - 1$ *probes on the input and output bits of* Diffusion *yields independently distributed observations. In the masking domain, any* $\mathcal{B}_l - 1$ *collections of* n*-sharings at the input and output of* $\mathbb{S}$Diffusion *also remain jointly independent.*

Higher-Order Probing Security. For a bricklayer cipher using d-secure $\mathbb{S}$S-box gadgets over multiple rounds (Figure 2), the *Composition Theorem* from [28] states:

Lemma 2 (Composition Theorem [28]). *Suppose that any set of* t $\mathbb{S}$S-box*es receive jointly independent* n*-shared inputs. Then the overall cipher's probing security order* D *satisfies*

$$\min\{d, t\} \;\leq\; D \;\leq\; d.$$

In Ascon, where each S-box input comes from distinct 64-bit rows and $\mathcal{B}_l = 4$, it follows that $t \leq \mathcal{B}_l - 1 = 3$. Consequently, two consecutive rounds limit t, and adding more rounds does not see to decrease t further. Thus, a deterministic masking of Ascon can be secure up to *third order*.

3 Share Clustering

Before going into the design of masking schemes, we introduce a fundamental concept for grouping the shares in an n-sharing. This new method lays the groundwork for ensuring both probing security and uniformity. We focus on the case where n is a perfect square (the smallest example being $n = 4$). Consequently, we can write $n = s^2$ and partition the n shares into s lists (or *multi-shares*), each containing s elements. We refer to each such partition as a *cluster*.

Definition 4 (Cluster). *Let* $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ *be an* n*-sharing. A* cluster *is a partition* $\mathcal{C} = [\mathbf{A}_0, \ldots, \mathbf{A}_{s-1}]$ *of* $\mathbf{X}$ *into* s multi-shares*, each containing exactly* s *elements. In other words,* $\mathcal{C}$ *satisfies:*

- *Each multi-share* $\mathbf{A}_i$ *has* s *elements:* $|\mathbf{A}_i| = s$*.*
- *The union of all multi-shares equals the original sharing:*

$$\bigcup_{i=0}^{s-1} \mathbf{A}_i = \mathbf{X}.$$

– *Any two distinct multi-shares* $\mathbf{A}_i$ *and* $\mathbf{A}_j$ *are disjoint:*

$$\mathbf{A}_i \cap \mathbf{A}_j = \emptyset \quad \textit{for } i \neq j.$$

Example 3 ($n = 4$*).* Let $\mathbf{X} = [X_0, X_1, X_2, X_3]$ be a 4-sharing (so $s = 2$). One possible cluster is

$$\mathcal{C}^0 \;=\; \left[\, \mathbf{A}_0^0 = [X_0, X_1],\ \mathbf{A}_1^0 = [X_2, X_3] \right]. \tag{1a}$$

Another distinct cluster for the same set of shares is

$$\mathcal{C}^1 \;=\; \left[\, \mathbf{A}_0^1 = [X_0, X_2],\ \mathbf{A}_1^1 = [X_1, X_3] \right]. \tag{1b}$$

We will see that these two clusters are *Maximally Non-Overlapping (MNO).*

Definition 5 (MNO Clusters). *Two clusters* $\mathcal{C}^1$ *and* $\mathcal{C}^2$ *are called* MNO *if, for every multi-share* $\mathbf{A}_i^1$ *in* $\mathcal{C}^1$ *and every multi-share* $\mathbf{A}_j^2$ *in* $\mathcal{C}^2$*, their intersection contains exactly one element:*

$$\left|\mathbf{A}_i^1 \cap \mathbf{A}_j^2\right| \;=\; 1 \quad \textit{for all } 0 \leq i, j < s.$$

In Example 3, observe how

$$\mathbf{A}_0^0 \cap \mathbf{A}_0^1 \;=\; [X_0] \quad \text{and} \quad \mathbf{A}_0^0 \cap \mathbf{A}_1^1 \;=\; [X_1],$$

demonstrating the MNO property for $\mathcal{C}^0$ and $\mathcal{C}^1$. Moreover, a third cluster,

$$\mathcal{C}^2 \;=\; \left[\, \mathbf{A}_0^2 = [X_0, X_3],\ \mathbf{A}_1^2 = [X_1, X_2] \right], \tag{1c}$$

can be formed such that any two clusters in the set

$$\mathfrak{C} \;=\; \{\mathcal{C}^0, \mathcal{C}^1, \mathcal{C}^2\}$$

are MNO.

Definition 6 (Simultaneous MNO Clusters). *A collection of clusters* $\mathfrak{C} = \{\mathcal{C}^0, \mathcal{C}^1, \ldots, \mathcal{C}^{|\mathfrak{C}|-1}\}$ *is called* simultaneous MNO *(SMNO) if every pair of clusters in* $\mathfrak{C}$ *is MNO.*

3.1 Maximum Number of SMNO Clusters

For the smallest non-trivial case $n = 4$, it is impossible to construct more than three clusters that remain SMNO. This observation motivates an investigation into the maximum number of SMNO clusters for larger square values of n.

Example 4 ($n = 9$*).* To visualize potential cluster arrangements for $n = 9$, we place the nine shares in a 3×3 square, as shown in Figure 3. This representation aids in identifying how shares can be grouped into clusters that satisfy the SMNO property.

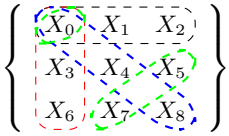

Fig. 3: First multi-share of each of the four SMNO clusters of $\mathbf{X}$ for $n = 9$.

In this 9-sharing scenario, one can construct four SMNO clusters by taking rows, columns, diagonals, and anti-diagonals of the share square. Specifically, these clusters are given by:

$$\mathcal{C}^0 = \big[\,[X_0, X_1, X_2],\ [X_3, X_4, X_5],\ [X_6, X_7, X_8]\big], \tag{2a}$$
$$\mathcal{C}^1 = \big[\,[X_0, X_3, X_6],\ [X_1, X_4, X_7],\ [X_2, X_5, X_8]\big], \tag{2b}$$
$$\mathcal{C}^2 = \big[\,[X_0, X_4, X_8],\ [X_1, X_5, X_6],\ [X_2, X_3, X_7]\big], \tag{2c}$$
$$\mathcal{C}^3 = \big[\,[X_0, X_5, X_7],\ [X_1, X_3, X_8],\ [X_2, X_4, X_6]\big]. \tag{2d}$$

Other Values of n. For $n = 4^2 = 16$, one can form five SMNO clusters (see Figure 4). In contrast, for $n = 6^2 = 36$, only three SMNO clusters are possible. The number of SMNO clusters is therefore not a straightforward (or even monotonic) function of n.

Remark 1. For any $n = s^2$ with $s \geq 2$, the clusters formed by rows, columns, and diagonals of an $s \times s$ arrangement always exist and are SMNO. Hence, for each square n, there are at least three SMNO clusters.

In the following sections, we will make use of *all* the SMNO clusters in our design. We, therefore, want to establish the maximum number of such clusters and to understand how they can be constructed.

A Systematic Clustering Structure. One convenient way to form clusters in an n-sharing is to lay out the shares in an $s \times s$ square, where $s = \sqrt{n}$. Let us denote the row-based cluster (indexed by 0) as follows:

$$\mathbf{A}_j^0 \ = \ \big\{\, X_l \,\big|\, l = i + js,\ 0 \leq i \leq s-1 \big\} \quad \text{for } 0 \leq j \leq s-1.$$

Subsequent clusters are generated using

$$\mathbf{A}_j^h \ = \ \big\{\, X_l \,\big|\, l = is + \mathsf{mod}(hi + j - i,\ s),\ 0 \leq i \leq s-1 \big\},$$

where $h \in \{1, \ldots, s\}$ identifies the cluster, $\mathsf{mod}(a, b)$ is the remainder of a divided by b, and $0 \leq j \leq s-1$. This construction yields $s+1$ potential clusters[1] for each s.

[1] Not necessarily all SMNO.

Lemma 3. *In an n-sharing with $s = \sqrt{n}$ prime, there are exactly $s + 1$ SMNO clusters. Furthermore, if s is composite with prime factorization $s = \prod_{i=1}^{r} p_i^{e_i}$ (where $p_1 < p_2 < \cdots < p_r$), then there exist at least $p_1 + 1$ clusters that are SMNO.*

Proof. See Appendix A for details. □

The SMNO clusters introduced here play a central role in our subsequent discussion. By organizing shares according to SMNO clusters, we achieve desirable properties related to both probing security and uniformity.

Interestingly, these constructions also connect to structures in *finite affine geometry*, further enriching the theoretical perspective on share clustering.

3.2 Finite Affine Geometry

In this section, we show how to view SMNO clusters through finite affine geometry. For an introduction to the topic see [5]. If the number of shares is $n = s^2$ with s prime, then a maximum set of clusters that is SMNO constitutes a finite affine plane of prime order s. In such a plane, there are s^2 *points* and $s^2 + s$ *lines* of s points each. Each point, moreover, lies on $s+1$ lines. These statements follow from the axioms for a finite affine plane. The three axioms for a finite affine plane are:

1. For every two distinct points, there is a unique line that contains both.
2. For each line L and point X not on L, there is precisely one line L' that contains X and is disjoint from L.
3. It is possible to choose a set of four points, such that no three of them lie on a single line.

The second axiom is often called Playfair's axiom [34] and can be seen as a replacement of Euclid's fifth postulate. Verifying the axioms for SMNO clusters is not hard (see below), and the number of SMNO clusters is then easily deduced by dividing the number of lines (clusters) s^2+s by the number of points (shares) on each line s (the number of multi-shares in a cluster), to find the number of SMNO clusters, $s + 1$.

Proof of the Properties of a Finite Affine Plane for SMNO Clusters. The first axiom that there is a unique line that contains both distinct points, is just a combination of having the union of all multi-shares being $\mathbf{X}$ and having all clusters be SMNO; $|\mathbf{A}_i^k \cap \mathbf{A}_j^l| \leq 1$ for all i, j, k, l, provided we do not have $i = j$ and $k = l$. The second axiom is already satisfied within a single cluster if the first line is $\mathbf{A}_i^k$, and $X \notin \mathbf{A}_i^k$, then there is some $j \neq i$ such that $X \in \mathbf{A}_j^k$. The third axiom works by taking the set of four points to be

$$\{\mathbf{A}_0^0[1], \mathbf{A}_0^0[2], \mathbf{A}_1^0[1], \mathbf{A}_2^0[1]\}.$$

By design, these are not all on any line $\mathbf{A}_j^0$, since $\mathbf{A}_j^0 \cap \mathbf{A}_0^0 = \emptyset$ for all $j \neq 0$. Furthermore, these four points are not all on any line $\mathbf{A}_j^k$ with $k \neq 0$, since then $|\mathbf{A}_0^0 \cap \mathbf{A}_j^k| \geq 2$. □

From this, we can also easily deduce that the number of shares should be a square.

Lemma 4. *The number of SMNO clusters is* 1 *if the number of shares is not a square.*

Proof. If we have $n \cdot m$ shares with $n > m$, then each multi-share in a cluster contains a divisor d_n of n shares; or a divisor d_m of m shares. Assume, without loss of generality, that each cluster has d_m multi-shares. Then each multi-share in a cluster has $e_m = \frac{m}{d_m} \cdot n$ shares. Suppose that we have two MNO clusters $C = [c_0, c_1, \ldots, c_{d_m-1}]$ and $B = [b_0, b_1, \ldots, b_{d_m-1}]$. Then we must have $|b_i \cap c_j| = 1$ for all i, j. Since all c_j are disjoint, and we have d_m of those, we have $|b_i| = d_m$, a contradiction. Hence there cannot be two MNO clusters, if the number of shares is not a square. □

Remark 2. By Lemma 4, we see that we need to have a square number of shares to obtain a maximal number of SMNO clusters. A heuristic way to consider this, is by seeing that for, say, 3rd order security, we need to have at least 3 rows and at least 3 columns. Therefore, the minimum number of shares is 9.

It is an open conjecture in the field of finite geometry that a finite plane of order s (such that each line has s points) exists if and only if s is a prime power. In Lemma 3, s is a prime number, so this follows the theory. As described in [29], we know that if s is any prime power, we still have $s+1$ SMNO clusters. The method to obtain them, however does not follow the descending diagonal construction in Lemma 3, as that only works for prime numbers and gives a lower bound for the number of SMNO clusters for composite numbers as described there. We present an example for $s = 4$ of the five SMNO clusters that we can have in Figure 4.

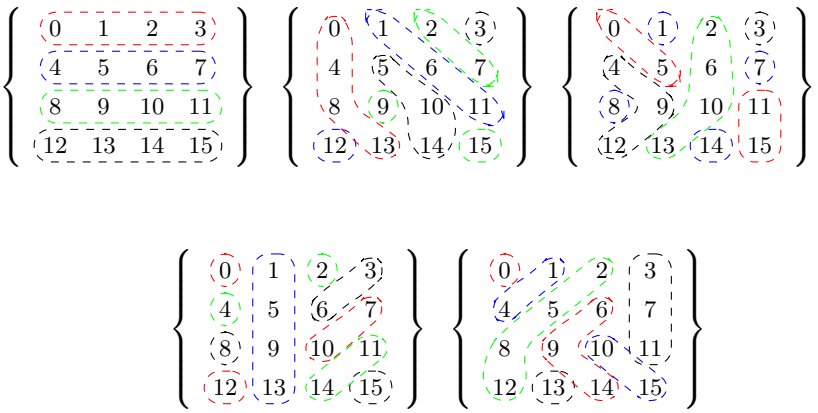

Fig. 4: Affine plane of order 16, or, multi-shares of each of the five SMNO clusters for $n = 16$.

In particular it shows that our clustering approach for designing multi-shares works as well in the case where $s = \sqrt{n}$ is any prime power, instead of s being a prime number.

3.3 Incompleteness of Multi-Shares

The clustering approach introduced so far facilitates reasoning about the probing security of masking schemes. In this subsection, we show how clusters remain robust against a specific probing attack in which each probe (similar to a glitch-extended probe) recovers one entire multi-share from an n-sharing $\mathbf{X}$. We begin with a simple case and then state the general result.

A Simple Case. Suppose each probe of the adversary reveals one multi-share of an n-sharing $\mathbf{X}$, where $n = s^2$. How many such probes are needed to recover *all* shares in $\mathbf{X}$? The answer is straightforward: if the s probed multi-shares all belong to the *same* cluster, then the adversary will learn every share in $\mathbf{X}$.

A More Involved Case. If the probed multi-shares do not all come from a single cluster, then more probes are needed. The following lemma establishes that at least $2s - 1$ such probes are required to recover all shares in $\mathbf{X}$ when no cluster is fully covered by those probes.

Lemma 5. *Let $\mathbf{X}$ be an n-sharing with $n = s^2$. Suppose the adversary probes a collection of multi-shares such that* no single cluster *is fully covered (i.e., each cluster is missing at least one multi-share). Then, if the number of probed multi-shares is less than $2s - 1$, the adversary cannot recover all n shares of $\mathbf{X}$.*

Proof. Due to the property that any two multi-shares intersect in exactly one share, a cluster with $s-1$ of its multi-shares probed (lacking exactly one) would require s additional multi-shares from other clusters to cover every element in that cluster's missing multi-share. Consequently, fewer than $2s-1$ probed multi-shares cannot reconstruct the complete set of n shares. □

For instance, when $n = 9$ and the SMNO clusters are given in (2), one can verify that, to cover *all* shares of $\mathbf{X}$ using at most two multi-shares per cluster, at least five multi-shares are required in total.

Additional Intuition via Rows and Columns. An intuitive illustration of Lemma 5 can be given by considering two clusters defined by rows and columns of an $s \times s$ square: Assume the adversary knows all row multi-shares except row i in the row-based cluster. Similarly, the adversary knows all column multi-shares except column j in the column-based cluster. In that scenario, the share X_{is+j} (the intersection of the missing i-th row and j-th column) remains unknown. Hence, knowing all but one row and all but one column (in total $2(s-1) = 2s-2$ multi-shares) still does *not* reveal all shares. This shows the need for $2s - 1$ probed multi-shares.

3.4 Linear Independence of Clusters

Clusters constructed for odd prime $s = \sqrt{n}$ exhibit an algebraic structure that is instrumental in proving the uniformity of the output shares in our proposed gadgets. By Lemma 3, there are $s + 1$ SMNO clusters when s is prime. Each cluster $\mathcal{C}^h$ consists of s multi-shares, and each multi-share $\mathbf{A}_j^h$ contains exactly s individual shares from the n-sharing $\mathbf{X}$. We use the notation

$$\oplus \mathbf{A}_j^h$$

to denote the bitwise sum (i.e., XOR) of all shares in the multi-share $\mathbf{A}_j^h$. For instance, when $n = 9$, the expression $\oplus \mathbf{A}_0^0$ equals $X_0 \oplus X_1 \oplus X_2$.

Key Observation: Maximal Linear Independence. Consider the set of sums

$$\left\{ \oplus \mathbf{A}_j^h \;\middle|\; 0 \le h \le s,\, 0 \le j \le s-1 \right\},$$

which forms $s(s + 1)$ parity equations. A crucial property is that these sums exhibit *maximal* linear independence, with only some dependencies introduced by the fact that the total sum of each cluster is the same. Concretely, for each cluster $\mathcal{C}^h$, we have

$$\bigoplus \mathcal{C}^h \;=\; \bigoplus_{j=0}^{s-1} \left(\oplus \mathbf{A}_j^h \right) \;=\; \bigoplus_{i=0}^{n-1} X_i \;=\; X,$$

where X is secret of $\mathbf{X}$. Consequently, for any two clusters h_1 and h_2, one finds

$$\bigoplus_{j=0}^{s-1} \left(\oplus \mathbf{A}_j^{h_1} \right) \;=\; \bigoplus_{j=0}^{s-1} \left(\oplus \mathbf{A}_j^{h_2} \right).$$

Thus, each additional cluster beyond the first introduces one dependent relation among the sums. As there are $s + 1$ clusters, the number of such dependencies is s. Therefore, the net rank of the system of $s(s + 1)$ parity equations is

$$(s+1)s - s \;=\; s^2 \;=\; n.$$

Example 5. (Case $n = 9$, $s = 3$) Recall the clusters from (2). Since $s = 3$ is an odd prime, we expect a total rank of 9. We construct the following set of 9 independent equations, omitting one relation from each new cluster (after the first) to avoid introducing linear dependence:

$$\begin{cases} X_0 \oplus X_1 \oplus X_2 \;=\; \oplus \mathbf{A}_0^0, & X_3 \oplus X_4 \oplus X_5 \;=\; \oplus \mathbf{A}_1^0, & X_6 \oplus X_7 \oplus X_8 \;=\; \oplus \mathbf{A}_2^0, \\ X_0 \oplus X_3 \oplus X_6 \;=\; \oplus \mathbf{A}_0^1, & X_1 \oplus X_4 \oplus X_7 \;=\; \oplus \mathbf{A}_1^1, & X_0 \oplus X_4 \oplus X_8 \;=\; \oplus \mathbf{A}_0^2, \\ X_1 \oplus X_5 \oplus X_6 \;=\; \oplus \mathbf{A}_1^2, & X_0 \oplus X_5 \oplus X_7 \;=\; \oplus \mathbf{A}_0^3, & X_1 \oplus X_3 \oplus X_8 \;=\; \oplus \mathbf{A}_1^3. \end{cases}$$

These nine equations are linearly independent, giving the system a rank of 9. □

General Case for Odd Prime s. This linear independence argument applies for all odd prime powers $s = p^k$. The formal statement appears below.

Lemma 6. *Let s be an odd prime power, and let $n = s^2$. Consider all sums*

$$\oplus \mathbf{A}_j^h \quad \textit{for} \quad 0 \le h \le s, \quad 0 \le j \le s-1,$$

obtained from the SMNO clusters of an n-sharing $\mathbf{X}$. Then the rank of the corresponding system of parity relations is n.

Proof. For each share X_i in $\mathbf{X}$, we identify a specific subset of these parity sums that reconstructs X_i. Exclude the *row* cluster ($\mathcal{C}^0$) and let $\mathcal{B}_i$ be the set of all remaining multi-shares that contain X_i:

$$\mathcal{B}_i \;=\; \big\{ \mathbf{A}_j^h \;\big|\; 1 \le h \le s,\; 0 \le j \le s-1,\; X_i \in \mathbf{A}_j^h \big\}.$$

Since X_i appears in exactly one multi-share of each of these s clusters, $\mathcal{B}_i$ has s elements. Any two multi-shares in $\mathcal{B}_i$ intersect exactly in $[X_i]$, so combining all elements of $\mathcal{B}_i$ yields s copies of X_i and one copy of each additional share in the relevant rows. Let j_0 be such that $X_i \in \mathbf{A}_{j_0}^0$ in the row cluster. Because s is odd, we obtain

$$X_i \;=\; \Big(\bigoplus_{\substack{j=0 \\ j \ne j_0}}^{s-1} \oplus \mathbf{A}_j^0 \Big) \;\oplus\; \Big(\bigoplus_{\mathbf{A}_j^h \in \mathcal{B}_i} \oplus \mathbf{A}_j^h \Big). \tag{3}$$

Since each X_i can be expressed in this manner, the system of all sums $\oplus \mathbf{A}_j^h$ must have rank at least n. On the other hand, there are only n variables total, so the rank is exactly n. □

Even Prime Case ($s = 2$). For example, when $s = 2$, the rank of the corresponding system of parity sums is 3, and for $s = 4$, it is 9. However, the statement of Lemma 6 for odd prime powers is sufficient for our constructions in the next section.

Example 6. (Detail for $n = 9$, X_5) Referring again to the clusters in (2), consider the share X_5. The set

$$\mathcal{B}_5 \;=\; \{\mathbf{A}_2^1,\; \mathbf{A}_1^2,\; \mathbf{A}_0^3\}$$

collects all multi-shares from clusters $1, 2, 3$ that contain X_5. Summing their parities,

$$\oplus \mathcal{B}_5 \;=\; (\oplus \mathbf{A}_2^1) \oplus (\oplus \mathbf{A}_1^2) \oplus (\oplus \mathbf{A}_0^3) \;=\; X_2 \oplus X_5 \oplus X_8 \oplus X_1 \oplus X_6 \oplus X_0 \oplus X_7.$$

Since X_5 lies in the row-based multi-share $\mathbf{A}_1^0$, equation (3) for $i = 5$ becomes

$$X_5 \;=\; (\oplus \mathbf{A}_0^0) \;\oplus\; (\oplus \mathbf{A}_2^0) \;\oplus\; (\oplus \mathcal{B}_5).$$

Hence, X_5 can be recovered from the row cluster (excluding its own row) and the parity sums in $\mathcal{B}_5$. □

4 Higher-Order Deterministic Multiplication

With our building blocks in place, we now turn to designing gadgets that perform higher-order deterministic AND operation. We start with a simpler gadget that is glitch resistant but is not uniform.

A $\mathbb{S}$AND gadget takes two n-sharings $\mathbf{X}$ and $\mathbf{Y}$ as inputs and computes an n-sharing $\mathbf{Z}$ such that the native value Z satisfies $Z = XY$.

Binomial Grouping. Given input n-sharings $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ and $\mathbf{Y} = [Y_0, \ldots, Y_{n-1}]$, for the native X and Y we can write:

$$Z \;=\; XY \;=\; \bigoplus_{i=0}^{n-1} X_i \bigoplus_{j=0}^{n-1} Y_j \;=\; \bigoplus_{i=0}^{n-1}\bigoplus_{j=0}^{n-1} X_i Y_j \;\overset{(\mathtt{I})}{=}\; \bigoplus_{k=0}^{n-1} Z_k. \tag{4}$$

Our goal is to partition the n^2 products $\{X_i Y_j\}$ into n disjoint *bins*, each assigned to a share Z_k. Intuitively, by carefully grouping these products into each Z_k, we aim to maximize the gadget's glitch-extended probing security order.

Maximizing the Probing Security Order. Let $\mathbf{L}_k$ be the set of all pairs (i, j) such that $X_i Y_j$ is included in Z_k. In other words,

$$Z_k \;=\; \bigoplus_{(i,j)\in\mathbf{L}_k} X_i\, Y_j,$$

and the partitioning condition implies

$$\bigcup_{k=0}^{n-1} \mathbf{L}_k \;=\; \{0, \ldots, n-1\}^2 \quad \text{and} \quad \mathbf{L}_k \cap \mathbf{L}_{k'} = \emptyset \quad (\text{for } k \neq k').$$

A glitch-extended probe on Z_k reveals *all* pairs $\{X_i, Y_j \mid (i, j) \in \mathbf{L}_k\}$. Denoting

$$\mathbf{I}_k \;=\; \{\, i \mid \exists\, j \text{ with } (i, j) \in \mathbf{L}_k\} \quad \text{and} \quad \mathbf{J}_k \;=\; \{\, j \mid \exists\, i \text{ with } (i, j) \in \mathbf{L}_k\},$$

we have $|\mathbf{L}_k| \leq |\mathbf{I}_k| \cdot |\mathbf{J}_k|$. Summing over all k, we get

$$n^2 \;=\; \sum_{k=0}^{n-1} |\mathbf{L}_k| \;\leq\; \sum_{k=0}^{n-1} |\mathbf{I}_k|\,|\mathbf{J}_k|. \tag{5}$$

To ensure high security order, we want each $|\mathbf{L}_k|$ to be as small as possible—equivalently, we want $|\mathbf{I}_k|$ and $|\mathbf{J}_k|$ to be bounded. Setting

$$I \;=\; \max_{0\leq k\leq n-1} |\mathbf{I}_k| \quad \text{and} \quad J \;=\; \max_{0\leq k\leq n-1} |\mathbf{J}_k|,$$

it follows from (5) that

$$n^2 \;\leq\; \sum_{k=0}^{n-1} |\mathbf{I}_k|\,|\mathbf{J}_k| \;\leq\; n\,I\,J \;\leq\; n\,(\max\{I, J\})^2 \;\Longrightarrow\; \sqrt{n} \;\leq\; \max\{I, J\}. \tag{6}$$

Hence, to maximize the order of security, we want $\max\{I, J\} = \sqrt{n}$.

A Clustering-Based Construction. To achieve $\max\{I, J\} = \sqrt{n}$, we use the SMNO clustering approach from Section 3. Recall that n is a perfect square, say $n = s^2$. We define, for each $k \in \{0, \ldots, n-1\}$,

$$\mathbf{L}_k \;=\; \mathbf{I}_k \times \mathbf{J}_k,$$

where

$$\begin{aligned} \alpha \;&=\; \lfloor k/s \rfloor, \quad \mathbf{X}\big|_{\mathbf{I}_k} = \; \mathbf{A}_{\alpha}^{h_1}(\mathbf{X}), \\ \beta \;&=\; \mathsf{mod}(k, s), \quad \mathbf{Y}\big|_{\mathbf{J}_k} = \; \mathbf{A}_{\beta}^{h_2}(\mathbf{Y}). \end{aligned} \tag{7}$$

Here, h_1 and h_2 index two distinct SMNO clusters; for example, one might use the "row" cluster for $\mathbf{X}$ and the "column" cluster for $\mathbf{Y}$. The parameters α and β range over all possible values in $\{0, \ldots, s-1\}$, ensuring a complete covering of the $s \times s$ square.

As a result, we obtain $|\mathbf{I}_k| = |\mathbf{J}_k| = s$. Hence,

$$I = J = s = \sqrt{n},$$

and we achieve equality in (6). Defining each share Z_k as

$$\begin{aligned} Z_k \;&=\; \mathbf{A}_{\alpha}^{h_1}(\mathbf{X}) \;\otimes\; \mathbf{A}_{\beta}^{h_2}(\mathbf{Y}) \\ &\overset{\mathsf{def}}{=} \bigoplus_{i=0}^{s-1} \bigoplus_{j=0}^{s-1} \Big(\mathbf{A}_{\alpha}^{h_1}(\mathbf{X})[i]\Big)\Big(\mathbf{A}_{\beta}^{h_2}(\mathbf{Y})[j]\Big) \\ &=\; \Big(\bigoplus_{i=0}^{s-1} \mathbf{A}_{\alpha}^{h_1}(\mathbf{X})[i]\Big) \;\cdot\; \Big(\bigoplus_{j=0}^{s-1} \mathbf{A}_{\beta}^{h_2}(\mathbf{Y})[j]\Big), \end{aligned} \tag{8}$$

we obtain an n-sharing $\mathbf{Z} = [Z_0, \ldots, Z_{n-1}]$ that satisfies correctness by construction, while also maximizing the probing security order under glitch-extended probes.

4.1 The $\mathbb{S}$AND-DN Gadget

The output $\mathbf{Z}$ of the multiplication described in this section is not uniformly distributed. For future reference, we denote this gadget by $\mathbb{S}$AND-DN, where "D" stands for "deterministic" and "N" indicates "non-uniform."

Algorithm 1 presents the $\mathbb{S}$AND-DN gadget for input clusters h_1 and h_2. As an example, for $s = 2$ and $(h_1, h_2) = (0, 1)$ following Example 3, the shares of Z are given by

$$\begin{cases} Z_0 = (X_0 \oplus X_1)\,(Y_0 \oplus Y_2), \\ Z_1 = (X_0 \oplus X_1)\,(Y_1 \oplus Y_3), \\ Z_2 = (X_2 \oplus X_3)\,(Y_0 \oplus Y_2), \\ Z_3 = (X_2 \oplus X_3)\,(Y_1 \oplus Y_3). \end{cases}$$

Further details for the case $s = 3$ are worked out in Appendix B.

Algorithm 1 The $\mathbb{S}$AND-DN gadget

Input: $\mathbf{X}, \mathbf{Y} \in (\mathbb{F}_{2^u})^n$; cluster indices h_1, h_2
Output: $\mathbf{Z} \in (\mathbb{F}_{2^u})^n$ such that $Z = XY$

1: **for** $k = 0$ to $n-1$ **do**
2: $\quad \alpha \leftarrow \lfloor k/s \rfloor$
3: $\quad \beta \leftarrow \mathsf{mod}(k, s)$
4: $\quad Z_k \leftarrow \mathbf{A}_\alpha^{h_1}(\mathbf{X}) \otimes \mathbf{A}_\beta^{h_2}(\mathbf{Y})$
5: **return** $\mathbf{Z} = [\, Z_0, Z_1, \ldots, Z_{n-1} \,]$

Correctness. To verify $Z = XY$, observe that the pair $(\alpha, \beta) = \big(\lfloor k/s \rfloor, \mathsf{mod}(k, s)\big)$ for $0 \le k \le n-1$ systematically spans all combinations of multi-shares from clusters h_1 and h_2 under the $\otimes$ operation. As a result, each product term $X_i\, Y_j$ (for $0 \le i, j < n$) appears exactly once in one of the Z_k. Hence,

$$\bigoplus_{k=0}^{n-1} Z_k \;=\; \bigoplus_{i=0}^{n-1} \bigoplus_{j=0}^{n-1} (X_i\, Y_j) \;=\; X\, Y,$$

so the gadget preserves correctness of the multiplication.

4.2 Properties of the Proposed $\mathbb{S}$AND-DN Gadget

Probing Security. In this gadget, any collection of up to $d = s-1$ (glitch-extended) probes only exposes an *incomplete* set of input shares, thanks to the clustering approach. Consequently, these probes remain independent of X and Y, and hence of Z. However, a set of $d+1$ glitch-extended probes can reveal the entire X and Y to the adversary.

Concretely, with $d+1$ probes, the adversary can learn

$$\mathbf{P} \;=\; \{Z_{ks+k} \;\mid\; 0 \le k < s\}.$$

From (8), each probed share is

$$P_k \;=\; Z_{ks+k} \;=\; \left(\bigoplus_{i=0}^{s-1} \mathbf{A}_k^{h_1}(\mathbf{X})[i] \right) \cdot \left(\bigoplus_{i=0}^{s-1} \mathbf{A}_k^{h_2}(\mathbf{Y})[i] \right). \tag{9}$$

Hence, the adversary learns the two sums

$$\bigoplus_{i=0}^{s-1} \mathbf{A}_k^{h_1}(\mathbf{X})[i] \quad \text{and} \quad \bigoplus_{i=0}^{s-1} \mathbf{A}_k^{h_2}(\mathbf{Y})[i],$$

for $0 \le k < s$. Summing over all k then reveals

$$\bigoplus_{k=0}^{s-1} \bigoplus_{i=0}^{s-1} \mathbf{A}_k^{h_1}(\mathbf{X})[i] \;=\; X \quad \text{and} \quad \bigoplus_{k=0}^{s-1} \bigoplus_{i=0}^{s-1} \mathbf{A}_k^{h_2}(\mathbf{Y})[i] \;=\; Y.$$

If $\mathbf{X}$ and $\mathbf{Y}$ are independent, then the gadget is secure up to $d = s-1$ glitch-extended probes. Each probe reveals only one multi-share of X (or Y), so fewer than s probes cannot fully recover X or Y.

Non-Uniformity of the Output Shares. For the binary field $\mathbb{F}_2$, each individual output share Z_k of the $\mathbb{S}$AND-DN gadget is biased. Specifically, for $0 \le k \le n-1$, we have

$$\Pr\big(Z_k = 0\big) \;=\; \tfrac{3}{4}.$$

Proof: From (8), we can write

$$Z_k \;=\; \Bigg(\bigoplus_{i=0}^{s-1} \mathbf{A}^{h_1}_{\lfloor k/s \rfloor}(\mathbf{X})[i]\Bigg) \;\cdot\; \Bigg(\bigoplus_{i=0}^{s-1} \mathbf{A}^{h_2}_{\mathsf{mod}(k,s)}(\mathbf{Y})[i]\Bigg).$$

In $\mathbb{F}_2$, each of these two sums is an independent, uniform random bit, so their product is zero except in the one case when both bits are 1. Therefore, the probability that $Z_k = 0$ is $\frac{3}{4}$.

Although the marginal distribution of each output share is biased, this gadget serves primarily as a stepping stone to our main construction, which will address uniformity. The primary motivation here is to demonstrate the fundamental clustering strategy and its impact on probing security, before tackling the uniformity issue in the next subsection.

4.3 A Uniform Deterministic Multiplication Gadget: $\mathbb{S}$AND-DU

Nikova et al. [31] introduced the first known example of a first-order, glitch-extended probing-secure, and *uniform* multiplication circuit (see also Section 2.1). Extending their method to higher orders remains an open problem. In this section, we contribute to this line of work by constructing a multiplication gadget denoted $\mathbb{S}$AND-DU, showing that for prime s, the gadget achieves both *uniformity* and $(t = s-1)$-order probing security (including glitch resistance).

Notion of r-Uniformity. Before detailing our construction, we clarify what level of uniformity a non-bijective map such as AND can achieve.

Definition 7 (r-Uniformity). *A list of shares $\mathbf{Z} \in (\mathbb{F}_{2^u})^n$ is called* r-uniform *if any sub-list $\mathbf{Z}|_{\mathbf{L}}$ of size $|\mathbf{L}| = r$ is distributed uniformly in $\big(\mathbb{F}_{2^u}\big)^r$.*

Since AND is not bijective over $\mathbb{F}_{2^u} \times \mathbb{F}_{2^u}$, its output Z may be biased, even when inputs X and Y are uniform. In particular, knowledge of the entire sharing $\mathbf{Z}$ reveals the underlying native value Z, which is itself non-uniform. Thus, no $\mathbb{S}$AND construction can achieve *n-uniformity.*

However, as we demonstrate below, it is possible to achieve $(n-1)$-uniformity. Building on the clustering approach from Section 3, we present an $(n-1)$-uniform multiplication gadget.

Achieving a Uniform $\mathbf{Z}$*.* The central idea is to construct an n-sharing $\mathbf{R}$ whose secret is 0 and add it to $\mathbf{Z}$ without degrading the gadget's probing security. Our proofs of probing security rely on *incompleteness*, so we must ensure that any multi-share appearing in Z_k also appears in R_k. In this way, adding $\mathbf{R}$ does not allow an adversary to "complete" any missing shares from X or Y.

To guarantee that $\mathbf{R}$ indeed shares the secret value 0, each share X_i (or Y_i) should appear an *even* number of times across the different R_k. Consequently, each X_i (or Y_i) cancels itself out when all shares of $\mathbf{R}$ are summed.

This procedure ensures that each individual $Z_k + R_k$ is *marginally* uniform. Yet, for $(n-1)$-uniformity, we also need the shares of $\mathbf{R}$ to be *jointly* uniform. In other words, these R_k random variables must be sufficiently (linearly) independent.[2] We achieve this by carefully designing $\mathbf{R}$ so that no unintended dependencies arise among its components, thus preserving joint uniformity across the entire n-sharing.

In the next subsection, we detail how to construct $\mathbf{R}$ and integrate it with the non-uniform gadget from Section 4.1 to obtain our final $\mathbb{S}$AND-DU gadget with $(n-1)$-uniformity and higher-order probing security.

Design of $\mathbb{S}$AND-DU Given the input n-sharings $\mathbf{X}$ and $\mathbf{Y}$, the gadget computes each output share Z_k as the XOR of two terms: a multiplication term W_k and an auxiliary share R_k. Unlike the simpler $\mathbb{S}$AND-DN construction, this gadget uses more than two clusters. Concretely, for $0 \leq k < n$:

$$\begin{aligned}
\alpha &= \lfloor k/s \rfloor, \\
\beta &= \mathsf{mod}(k, s), \\
R_k &= \bigoplus_{i=0}^{s-1} \Big(\mathbf{A}_\alpha^0(\mathbf{X})[i] \oplus \mathbf{A}_\beta^{\alpha+1}(\mathbf{X})[i] \oplus \mathbf{A}_\alpha^0(\mathbf{Y})[i] \oplus \mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})[i] \Big), \\
W_k &= \mathbf{A}_\alpha^0(\mathbf{X}) \otimes \mathbf{A}_\beta^{\alpha+1}(\mathbf{Y}), \\
Z_k &= R_k \oplus W_k.
\end{aligned} \tag{10}$$

Recall that $\mathbf{A}_j^h(\cdot)$ denotes the j-th multi-share from the SMNO cluster with index h. By Lemma 3, if s is prime, there exist $s+1$ mutually SMNO clusters.

Example 7 (Case $n = 4$). Using the four-share clusters from Example 3, the output shares of $\mathbb{S}$AND-DU are:

$$\begin{cases}
Z_0 = (X_0 \oplus X_1)(Y_0 \oplus Y_2) \oplus X_1 \oplus X_2 \oplus Y_1 \oplus Y_2, \\
Z_1 = (X_0 \oplus X_1)(Y_1 \oplus Y_3) \oplus X_0 \oplus X_3 \oplus Y_0 \oplus Y_3, \\
Z_2 = (X_2 \oplus X_3)(Y_0 \oplus Y_3) \oplus X_0 \oplus X_2 \oplus Y_0 \oplus Y_2, \\
Z_3 = (X_2 \oplus X_3)(Y_1 \oplus Y_2) \oplus X_1 \oplus X_3 \oplus Y_1 \oplus Y_3.
\end{cases}$$

In Appendix B, we detail an instance for $s = 3$.

Algorithm and Correctness. Algorithm 2 describes the $\mathbb{S}$AND-DU construction. Its correctness (i.e., that $Z = X\,Y$) follows from two main observations:

[2] If there exists a subset of $\mathbf{R}$ with r elements that sum to zero, then $\mathbf{R}$ can only be at most r-uniform.

– $\mathbf{W} = [W_0, \ldots, W_{n-1}]$ is an n-sharing of XY.
– $\mathbf{R} = [R_0, \ldots, R_{n-1}]$ is an n-sharing of 0.

We postpone the full correctness proof to Appendix C.

Algorithm 2 The $\mathbb{S}$AND-DU gadget

Input: $\mathbf{X}, \mathbf{Y} \in (\mathbb{F}_{2^u})^n$
Output: $\mathbf{Z} \in (\mathbb{F}_{2^u})^n$ such that $Z = XY$

for $k = 0$ to $n-1$ **do**
  $\alpha \leftarrow \lfloor k/s \rfloor$
  $\beta \leftarrow \mathsf{mod}(k, s)$
  $R_k \leftarrow 0$
  ▷ Compute the zero-sharing R_k.
  for $i = 0$ to $s-1$ **do**
    $R_k \leftarrow R_k \oplus \mathbf{A}^0_\alpha(\mathbf{X})[i] \oplus \mathbf{A}^{\alpha+1}_\beta(\mathbf{X})[i] \oplus \mathbf{A}^0_\alpha(\mathbf{Y})[i] \oplus \mathbf{A}^{\alpha+1}_\beta(\mathbf{Y})[i]$
  ▷ Compute the multiplication share W_k.
  $W_k \leftarrow \mathbf{A}^0_\alpha(\mathbf{X}) \otimes \mathbf{A}^{\alpha+1}_\beta(\mathbf{Y})$
for $k = 0$ to $n-1$ **do**
  $Z_k \leftarrow W_k \oplus R_k$
return $\mathbf{Z} = [Z_0, \ldots, Z_{n-1}]$

4.4 Properties of the Proposed $\mathbb{S}$AND-DU Gadget

Probing Security. We show that the $\mathbb{S}$AND-DU gadget is secure up to glitch-extended probing order $d = s - 1$, assuming $\mathbf{X}$ and $\mathbf{Y}$ are independent. Owing to the clustering approach, the proof is relatively straightforward.

The key observation is that computing up to d probed values requires only an *incomplete* set of shares from $\mathbf{X}$. Since this set is incomplete, the probed values remain independent of the native value X. The same argument applies to $\mathbf{Y}$.

From Algorithm 2, each probed value Z_k depends on two multi-shares indexed by α and β. Notably, these multi-shares lie in *different* clusters: multi-share α is from cluster 0, whereas multi-share β is from cluster $\alpha+1$, and we have $\alpha+1 > 0$.

By Lemma 5, a collection of fewer than $2s-1$ multi-shares across different clusters is *incomplete*. To ensure incompleteness, the following bound must hold for d:

$$2d \;<\; 2s-1 \;\implies\; d \;\leq\; s-1. \tag{11}$$

Thus, when probing $d = s - 1$ values, the adversary cannot fully probe all the n shares of $\mathbf{X}$ and $\mathbf{Y}$, preserving the incompleteness property and thereby the privacy of $\mathbf{X}$ and $\mathbf{Y}$.

Uniformity of $\mathbb{S}$AND-DU. For $s \in \{2, 3\}$, one can directly verify uniformity via exhaustive enumeration. Specifically, by constructing a truth table of all possible inputs and outputs and counting the frequency of each output $\mathbf{Z}$, one checks that:

– All $\mathbf{Z}$ with secret $Z = \oplus_{i=0}^{n-1} Z_i = 0$ appear with the same frequency.
– All $\mathbf{Z}$ with secret $Z = 1$ also appear with the same frequency.

This shows $(n-1)$-uniformity of the output shares. As discussed earlier, achieving n-uniformity is impossible because Z (the product of inputs) is inherently biased.

For larger s, enumerating all possible inputs grows infeasible. Instead, we use an *indirect counting* method. We prove that any subset of $\mathbf{Z}$ of size $n-1$ (denoted $\mathbf{Z}_1$) is uniformly distributed over $(\mathbb{F}_{2^u})^{n-1}$. Concretely, for each fixed realization $\mathbf{z}_1$, the number of input pairs $(\mathbf{X}, \mathbf{Y})$ that produce $\mathbf{Z}_1 = \mathbf{z}_1$ is the same, regardless of $\mathbf{z}_1$.

The main difficulty is that $\mathbf{Z}$ depends on $\mathbf{X}, \mathbf{Y}$ via a *non-linear* mapping. To overcome this, we introduce a technique to *linearize* the relevant parity relations, allowing us to apply linear algebra arguments to compute the number of preimages for any given $\mathbf{z}_1$. The full proof can be found in Appendix D.

5 Higher-Order Deterministic $\mathbb{S}$Toffoli Gadget

To demonstrate how the ideas developed thus far can be extended to deterministically mask a cipher, we introduce additional gadgets. Our target cipher is Ascon, which uses χ_5 as its core non-linearity. In the next section, we will focus on $\mathbb{S}\chi_5$. Here, we present a finer building block of χ_5 that is most naturally described by a Toffoli gate [38]. A Toffoli gate maps $(X, Y, Z) \in (\mathbb{F}_{2^u})^3$ to

$$(X,\, Y,\, W = Z \oplus X\,Y).$$

Correspondingly, an $\mathbb{S}$Toffoli gadget takes three n-sharings $[\mathbf{X}, \mathbf{Y}, \mathbf{Z}]$ and computes an n-sharing of $\mathbf{W}$, preserving

$$W \;=\; Z \oplus X\,Y.$$

Construction. Leveraging the clustering framework, we propose a $\mathbb{S}$Toffoli gadget as given in Algorithm 3. The scheme uses three (SMNO) clusters, assigning one to each input. By Lemma 3, for any s, it is possible to select such SMNO clusters. For brevity, we omit the correctness proof here, as it closely parallels that of $\mathbb{S}$AND-DN.

5.1 Properties of the $\mathbb{S}$Toffoli Gadget

We analyze probing security separately in the standard and glitch-extended settings.

Probing Security (Standard Probes). For any $0 \le k < n$, a probe P targeting the intermediate computations of W_k has the form $P = Z_k \oplus \mathsf{F}$. If no other probe targets this same W_k, the share Z_k *blinds* the information in F, allowing a simulator to replace P with an independent random variable (see Section 2.2).

Algorithm 3 The $\mathbb{S}$Toffoli gadget

Input: $\left[\mathbf{X}, \mathbf{Y}, \mathbf{Z}\right] \in \left(\mathbb{F}_{2^u}^n\right)^3$, cluster indices (h_X, h_Y, h_Z)
Output: $\mathbf{X}, \mathbf{Y}, \mathbf{W} \in \left(\mathbb{F}_{2^u}\right)^n$ such that $W = Z \oplus X\,Y$

for $k = 0$ to $n-1$ **do**
  $\alpha \leftarrow \lfloor k/s \rfloor, \quad \beta \leftarrow \mathsf{mod}(k, s)$
  $\mathbf{A}_\alpha^{h_Z}(\mathbf{W})[\beta] \leftarrow \mathbf{A}_\alpha^{h_Z}(\mathbf{Z})[\beta]$
  for $i = 0$ to $s-1$ **do**
    for $j = 0$ to $s-1$ **do**
      $\mathbf{A}_\alpha^{h_Z}(\mathbf{W})[\beta] \leftarrow \mathbf{A}_\alpha^{h_Z}(\mathbf{W})[\beta] \oplus \left(\mathbf{A}_\alpha^{h_X}(\mathbf{X})[i] \cdot \mathbf{A}_\beta^{h_Y}(\mathbf{Y})[j]\right)$
return $[\mathbf{X}, \mathbf{Y}, \mathbf{W}]$

However, if two or more probes target the same W_k, those probes can be simulated by revealing all variables in the computation of W_k. Consequently, two or more probes on a single W_k together yield:

$$\{\text{one share from } \mathbf{Z}\} \quad \text{and} \quad \{\text{one multi-share from } \mathbf{X} \text{ and } \mathbf{Y}\}.$$

Since $\mathbf{X}$ and $\mathbf{Y}$ each have s multi-shares and are assumed to be independent, up to

$$\left\lfloor \frac{d}{2} \right\rfloor < s \quad \Longrightarrow \quad d \leq 2s-1$$

probes can still be simulated using *incomplete* sets of input shares. Observe that each probe reveals at most one share from $\mathbf{Z}$, keeping the required $\mathbf{Z}$ shares incomplete as well.

For $n = 4$ (i.e., $s = 2$), we get $d \leq 3$, which is *optimal* in the sense that no deterministic (or even randomized) masking scheme can exceed a probing order $d = n - 1$.[3] For larger n, the achievable order becomes lower compared to $n-1$. For example, with $n = 9$, $s = 3$, we get $d \leq 5$.

Glitch-Extended Probing Security. The above simulator-based argument omits glitches, focusing only on standard probing. For glitch-extended probes, a probe on W_k automatically includes one share of $\mathbf{Z}$ *and* one multi-share from both $\mathbf{X}$ and $\mathbf{Y}$. Consequently, one obtains $d \leq s-1$ for glitch-extended security.

The $\mathbb{S}$AND-XOR Gadget. A Toffoli gate outputs three values (X, Y, $Z \oplus X\,Y$), copying the first two inputs to the output directly. In some applications (including those in the next section), only the combination $Z \oplus X\,Y$ is needed at the output. Thus, we define the $\mathbb{S}$AND-XOR gadget simply by omitting the X and Y outputs from the $\mathbb{S}$Toffoli gadget. Formally, it maps three n-sharings $[\mathbf{X}, \mathbf{Y}, \mathbf{Z}]$ into one n-sharing for $[\mathbf{Z} \oplus \mathbf{X}\,\mathbf{Y}]$, using the same cluster arrangement and inheriting the security properties just described.

[3] An adversary with n probes can directly learn all n input shares of an n-sharing.

6 Case Study on χ_m

Building on the previous sections, we now propose a deterministic higher-order probing secure masking scheme for χ_m. Our construction achieves a standard probing security order of $d = \sqrt{n}+1$. Due to computational constraints, we verify the uniformity of this scheme only for the specific cases $(n = 4, m = 5)$ and $(n = 4, m = 3)$. These choices correspond to the smallest square number of shares and the instances of χ_m used in Ascon [22], SHA-3 [1], and Xoodoo [19]. Notably, the combination $n = 4$ and $d = \sqrt{n} + 1 = 3$ is practically relevant because even with online randomness, no masking technique can surpass a probing security order of $d = 3$ for $n = 4$.

Uniformity Challenge for Masking χ_m. For odd m, χ_m is a permutation, implying that $\mathbb{S}\chi_m$ can, in principle, be a bijection of size mn. Thus, one might expect that if the inputs are mn-uniform, then the output could also be mn-uniform. However, a straightforward application of the Nikova et al. [31] multiplication (for $n = 4$) does *not* yield an mn-uniform construction of $\mathbb{S}\chi_m$. Early deterministic and uniform masking attempts for χ_m did not show promise [9].

To address this challenge, Daemen [17] proposed the "changing of the guards" technique, using randomness from adjacent $\mathbb{S}\chi_m$ instances to uniformize the current instance's output. In separate work, Shahmirzadi and Moradi [37] performed an exhaustive search to find a first- and second-order probing-secure χ_5 gadget that is also uniform. Here, we build on these ideas and present constructions for $\mathbb{S}\chi_5$ and $\mathbb{S}\chi_3$ that achieve *third-order* probing security while also ensuring uniformity for $n = 4$.

6.1 The $\mathbb{S}\chi_m$ Gadget

The map χ_m. Given m binary inputs $(X^0, \ldots, X^{m-1})$, χ_m produces m outputs $(Y^0, \ldots, Y^{m-1})$, where

$$Y^i \;=\; X^i \;\oplus\; \left(1 \oplus X^{i+1}\right) X^{i+2}, \tag{12}$$

with indices taken modulo m. Correspondingly, $\mathbb{S}\chi_m$ takes m input n-sharings $\left[\mathbf{X}^0, \ldots, \mathbf{X}^{m-1}\right]$ and computes m output n-sharings preserving the relation (12) among the native values.

When working over an n-sharing $\mathbf{X}$ of a secret X, it is straightforward to obtain an n-sharing for $(1 \oplus X)$ by simply adding 1 to exactly one share of $\mathbf{X}$, say X_0. With that, each output sharing $\mathbf{Y}^i$ can be computed using the $\mathbb{S}$AND-XOR gadget, as shown in Algorithm 4. An example for the case $(n = 4, m = 3)$ is provided in Appendix B.

This gadget requires three distinct SMNO clusters. Each n-sharing of the input appears in three separate $\mathbb{S}$AND-XOR gadgets, each associated with a different cluster. The shake function does not affect correctness or probing security order; it simply reverses the index order of the shares within each multi-share of the specified cluster. Without shake, the output of $\mathbb{S}\chi_m$ would not be uniform.

Algorithm 4 The $\mathbb{S}\chi_m$ gadget

Input: $[\mathbf{X}^0, \ldots, \mathbf{X}^{m-1}] \in \left(\mathbb{F}_2^n\right)^m$, cluster indices $\{h_0, h_1, h_2\}$
Output: $[\mathbf{Y}^0, \ldots, \mathbf{Y}^{m-1}] \in \left(\mathbb{F}_2^n\right)^m$

function $\mathsf{neg}(\mathbf{U})$
  $\mathbf{V} \leftarrow \mathbf{U}$
  $V_0 \leftarrow 1 \oplus U_0$
  return $\mathbf{V}$

function $\mathsf{shake}(\mathbf{U}, h)$
  $\mathbf{V} \leftarrow \mathbf{U}$
  for $j = 0$ **to** $s - 1$ **do**
    for $i = 0$ **to** $s - 1$ **do**
      $\mathbf{A}_j^h(\mathbf{V})[s - 1 - i] \leftarrow \mathbf{A}_j^h(\mathbf{U})[i]$
  return $\mathbf{V}$

▷ Compute outputs $\mathbf{Y}^i$.
for $i = 0$ **to** $m - 2$ **do**
  $\mathbf{Y}^i \leftarrow \mathbb{S}\mathsf{AND\text{-}XOR}\left(\mathbf{X}^{i+2}, \mathsf{neg}\left(\mathbf{X}^{i+1}\right), \mathbf{X}^i\right)$ ▷ Implements $Y^i = X^i \oplus (1 \oplus X^{i+1})\, X^{i+2}$.
$\mathbf{Y}^{m-1} \leftarrow \mathbb{S}\mathsf{AND\text{-}XOR}\left(\mathbf{X}^1, \mathsf{neg}\left(\mathbf{X}^0\right), \mathsf{shake}\left(\mathbf{X}^{m-1}, h_2\right)\right)$
return $[\mathbf{Y}^0, \ldots, \mathbf{Y}^{m-1}]$

While our motivation for shake is heuristic, we confirmed it produces mn-uniform results for $(n = 4, m = 5)$ and $(n = 4, m = 3)$. Determining its effectiveness for other cases remains an open problem for future work.

Probing Security (Standard Probes). Assume the input n-sharings are mutually independent and their clusters h_0, h_1, h_2 are distinct. From Algorithm 4, each $\mathbf{X}^i$ appears in exactly three consecutive $\mathbb{S}$AND-XOR gadgets, each time used under a *different* cluster index. The intermediates in these gadgets take the following form:

$$\begin{cases} D_1 = \mathbf{A}_{j_1}^{h_2}\left(\mathbf{X}^i\right)[j_2] \oplus \left(\bigoplus_{l_1=0}^{r_1} \bigoplus_{l_2=0}^{r_2} \mathbf{A}_{j_1}^{h_0}(\mathbf{X}^{i+1})[l_1]\, \mathbf{A}_{j_2}^{h_1}(\mathbf{X}^{i+2})[l_2]\right), \\ D_2 = \mathbf{A}_{j_3}^{h_2}\left(\mathbf{X}^{i-1}\right)[j_4] \oplus \left(\bigoplus_{l_1=0}^{r_3} \bigoplus_{l_2=0}^{r_4} \mathbf{A}_{j_3}^{h_0}\left(\mathbf{X}^i\right)[l_1]\, \mathbf{A}_{j_4}^{h_1}\left(\mathbf{X}^{i+1}\right)[l_2]\right), \\ D_3 = \mathbf{A}_{j_5}^{h_2}\left(\mathbf{X}^{i-2}\right)[j_6] \oplus \left(\bigoplus_{l_1=0}^{r_5} \bigoplus_{l_2=0}^{r_6} \mathbf{A}_{j_5}^{h_0}\left(\mathbf{X}^{i-1}\right)[l_1]\, \mathbf{A}_{j_6}^{h_1}\left(\mathbf{X}^i\right)[l_2]\right), \end{cases} \tag{13}$$

where $\{r_1, \ldots, r_6, j_1, \ldots, j_6\} \subseteq \{0, \ldots, s - 1\}$ label the relevant multi-share indices, consistent with Algorithm 3. For brevity, the effects of neg and shake are omitted in this illustration.

Single-Probe Case. A single probe targeting an intermediate such as D_3 becomes

$$D_3 = \underbrace{\mathbf{A}_{j_5}^{h_2}\left(\mathbf{X}^{i-2}\right)[j_6]}_{\text{blinding term}} \oplus \ldots$$

The additive share from cluster h_2 (here, $\mathbf{A}^{h_2}_{j_5}(\mathbf{X}^{i-2})[j_6]$) acts as a blinding term, ensuring that if no other probe discloses this same share, the simulator can replace D_3 with fresh randomness (Section 2.2).

Two or More Probes. With a second probe, simulation needs at most one multi-share from two *other* inputs. For example, by two probes on D_3 type intermediates, simulation requires a share of $\mathbf{X}^{i-2}$ along with one multi-share each of $\mathbf{X}^{i-1}$ and $\mathbf{X}^i$.

Upon learning one multi-share of a cluster $\mathbf{X}^{i-1}$, the adversary may choose the subsequent probes from type D_2. As such, the blinding term might have already been required for the simulation. Therefore, from the third probe on, we assume that any new probe will require one multi-share of $\mathbf{X}^i$ (the targeted n-sharing) for the simulation.

The first two probes only require one multi-share of $\mathbf{X}^i$ from cluster h_1. Subsequent probes require one multi-share from cluster h_0. Since h_0 and h_1 are pointing to two distinct (SMNO) clusters, fewer than

$$2 + (s - 1) \;=\; \sqrt{n} + 1$$

probes will not need all shares of $\mathbf{X}^i$ for the simulation, and this proves the claimed probing order $d = \sqrt{n} + 1$.

Glitch-Extended Probing Security. Using only the incompleteness argument (without relying on additive blinding), each intermediate in the $\mathbb{S}\chi_5$ gadget depends on at most one multi-share of each secret. Thus, the scheme resists *glitch-extended* probing up to $t = \sqrt{n} - 1$. Concretely, for $n = 4$, this provides first-order glitch resistance.

7 Application to Ascon

We now illustrate how our proposed $\mathbb{S}\chi_5$ can be used to implement a masked version of the Ascon cipher [22] without any online randomness.

Ascon Overview. Ascon is an authenticated encryption scheme based on a sponge construction, featuring a 128-bit key, tag, and nonce. As shown in Figure 5, its encryption algorithm runs in four stages:

1. *Initialization:* The 320-bit state is seeded with the key K and nonce N.
2. *Associated Data Processing:* The state is updated with blocks of associated data A_i.
3. *Plaintext Processing:* Plaintext blocks P_i are injected into the state to produce ciphertext blocks C_i.
4. *Finalization:* The key K is re-injected, and the tag T is extracted for authentication.

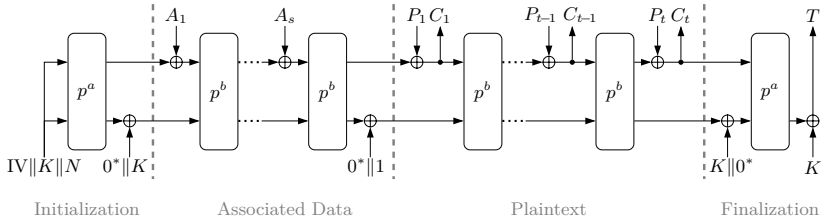

Fig. 5: Ascon. p^a and p^b are application of the permutation p in a and b rounds.

The Ascon Permutation. Ascon's permutation p repeatedly applies a round transformation to a 320-bit state, represented as a 5×64 array $\mathbf{s} = [\mathbf{s}_1, \ldots, \mathbf{s}_5]^\top$. Each round consists of:

1. *Add Round Constant:* A byte-sized constant is XORed into the state.
2. *Non-linear Layer:* A 5-bit S-box (see Figure 6) is applied in parallel across all 64 bit positions (columns).
3. *Diffusion Layer:* Each 64-bit word $\mathbf{s}_j$ is updated by XORing rotations of itself:

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

   where the rotations θ_j and θ'_j are specified in [22].

Ascon's S-box. Figure 6 shows the 5-bit S-box, which is χ_5 interleaved with small linear and affine layers. Each round contains 64 such S-box instances, one per bit column.

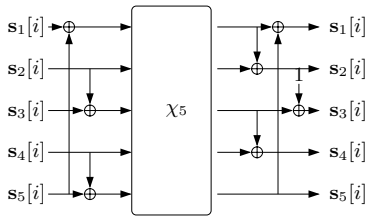

Fig. 6: Ascon's S-box.

7.1 A Deterministic $\mathbb{S}$Ascon

The only non-linear step in Ascon is the χ_5 function. In our deterministic masking approach, we replace χ_5 with the $\mathbb{S}\chi_5$ gadget from Section 6, while the diffusion layer and the linear/affine parts of the S-box are each masked directly without using any refresh gadgets or online randomness. This design follows the

work of Daemen et al. [18] and Giger et al. [26], which address first- and second-order probing security, respectively.[4] Our proposed $\mathbb{S}\chi_5$ extends these methods to third-order probing security.

Probing Security of Composition. For Ascon, the diffusion layer has linear branch number $\mathcal{B}_l = 4$ (Definition 3). Lemma 1 implies that any $\mathcal{B}_l - 1$ n-sharings in a single row (i.e., the $\mathbf{s}_i$ row lists) are jointly independent (Definition 2). In each application of the diffusion layer, rows partition the state so that the independence property is preserved for the entire state. By Lemma 2, if any three $\mathbb{S}$S-boxes receive jointly independent n-sharings, then the composition remains secure up to third-order probing, provided only two consecutive rounds are probed. As discussed in [28], the main independence bottleneck arises in two consecutive rounds; subsequent rounds introduce more S-box combinations, making it much more likely that the n-sharings remain independent.

Probing Security of $\mathbb{S}$S-box. Let the n-sharings entering $\mathbb{S}\chi_5$ be $[\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}]$. In Ascon, each S-box input is

$$[\ \mathbf{A} \oplus \mathbf{D} \oplus \mathbf{E},\ \mathbf{B},\ \mathbf{C} \oplus \mathbf{B},\ \mathbf{D},\ \mathbf{E} \oplus \mathbf{D}\ ],$$

which is just the inverse of the preceding linear layer (see Figure 6). Consequently, every intermediate can be written in the form of (13), combined with output affine terms. The key observation is that each output incorporates distinct blinding terms, allowing arguments similar to those used for $\mathbb{S}\chi_5$ in Section 6.

Probing Environment. In typical software implementations, glitches are less of a concern, so we focus here on standard probing security. To mitigate potential glitch-related vulnerabilities, one could introduce *synchronization layers* that ensure the blinding terms remain valid in each intermediate, without requiring additional randomness. We omit these details, as our primary goal is to illustrate the feasibility of higher-order deterministic masking through the clustering framework developed in this work.

8 Conclusion

We introduced a systematic *clustering* approach for shares and demonstrated its effectiveness in designing gadgets under the extreme constraints of *deterministic masking*, where randomness is limited to the initial input sharing. Using this method, we constructed a higher-order secure AND gadget, addressing a long-standing open problem. To further illustrate the use of clustering, we applied it to the deterministic protection of Ascon. Our results show that higher-order deterministic masking is feasible for both individual gadgets and larger cryptographic circuits. However, ensuring probing security in arbitrary compositions may be impossible without incorporating online randomness.

[4] The widely cited deterministic software masking of Ascon at `https://github.com/ascon/simpleserial-ascon` also follows these approaches for first and second order.

References

1. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015-08-04 2015). 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., 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)
4. 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)
5. Batten, L.M.: Combinatorics of Finite Geometries. Cambridge University Press (1997)
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. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Private Multiplication over Finite Fields. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 397–426. Springer International Publishing (2017)
8. Belaïd, S., Goudarzi, D., Rivain, M.: Tight Private Circuits: Achieving Probing Security with the Least Refreshing. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 343–372. Springer International Publishing (2018)
9. Bilgin, B., Daemen, J., Nikov, V., Nikova, S., Rijmen, V., Van Assche, G.: Efficient and First-Order DPA Resistant Implementations of Keccak. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications. pp. 187–199. Springer International Publishing (2014)
10. Cassiers, G., Grégoire, B., Levi, I., Standaert, F.X.: Hardware Private Circuits: From Trivial Composition to Full Verification. IEEE Transactions on Computers **70**(10), 1677–1690 (2021). https://doi.org/10.1109/TC.2020.3022979
11. 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
12. 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
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.: Changing of the Guards: A Simple and Efficient Method for Achieving Uniformity in Threshold Sharing. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 137–153. Springer (2017)
18. 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
19. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Xoodyak, a lightweight cryptographic scheme. IACR Trans. Symmetric Cryptol. **2020**(S1), 60–87 (2020)
20. 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
21. De Cnudde, T., Bilgin, B., Reparaz, O., Nikov, V., Nikova, S.: Higher-Order Threshold Implementation of the AES S-Box. In: Homma, N., Medwed, M. (eds.) Smart Card Research and Advanced Applications. Springer International Publishing (2016)
22. 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
23. 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
24. 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)
25. Gigerl, B., Hadzic, V., Primas, R., Mangard, S., Bloem, R.: Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs. In: Bailey, M.D., Greenstadt, R. (eds.) 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021. pp. 1469–1468. USENIX Association (2021), https://www.usenix.org/conference/usenixsecurity21/presentation/gigerl
26. Gigerl, B., Mendel, F., Schläffer, M., Primas, R.: Efficient Second-Order Masked Software Implementations of Ascon in Theory and Practice, NIST LWC Workshop (2023)
27. 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)

28. Jahandideh, V., Mennink, B., Batina, L.: Higher-Order Deterministic Masking with Application to Ascon. Cryptology ePrint Archive, Paper 2025/179 (2025), https://eprint.iacr.org/2025/179
29. Laywine, C.F., Mullen, G.L.: Discrete Mathematics Using Latin Squares. John Wiley & Sons (1998)
30. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag, Berlin, Heidelberg (2007)
31. 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)
32. Petrides, G.: On Non-Completeness in Threshold Implementations. In: Proceedings of ACM Workshop on Theory of Implementation Security Workshop. p. 24–28. TIS'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338467.3358951
33. 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)
34. Playfair, J.: The element of geometry. W.E. Dean (1836)
35. 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
36. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.X. (eds.) Cryptographic Hardware and Embedded Systems, CHES 2010. pp. 413–427. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
37. 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)
38. Toffoli, T.: Reversible computing. In: de Bakker, J., van Leeuwen, J. (eds.) Automata, Languages and Programming. pp. 632–644. Springer Berlin Heidelberg, Berlin, Heidelberg (1980)
39. Wang, W., Ji, F., Zhang, J., Yu, Y.: Efficient Private Circuits with Precomputation. IACR Transactions on Cryptographic Hardware and Embedded Systems (2), 286–309 (Mar 2023). https://doi.org/10.46586/tches.v2023.i2.286-309

A Proof of Lemma 3

In this section, we give a proof of Lemma 3, that at the same time gives a construction on how to create SMNO clusters in an easy way. The construction of a finite affine plane does not necessarily give the same SMNO clusters.

Proof of Lemma 3: We arrange shares in a square of side length s (as illustrated in Figure 3). The rows form one cluster. For this cluster, the shares $0 \leq i \leq s-1$ in the j-th multi-share are denoted by:

$$\mathbf{A}_j^0[i] = X_{i+js}. \tag{14}$$

Subsequent clusters are formed as follows:

$$\mathbf{A}_j^h[i] = X_{is+\mathsf{mod}(hi+j-i,s)}. \tag{15}$$

Here, each $h \in \{1, \ldots, s\}$ defines a cluster.

Clusters Covering $\mathbf{X}$. We show that each cluster covers all shares in $\mathbf{X}$. For a given h, any share X_l with $0 \leq l < s^2$ must be in one multi-share of this cluster. Setting $l = is+\mathsf{mod}(hi+j-i,s)$ yields $i = \lfloor l/s \rfloor$ and $\mathsf{mod}(l,s) = \mathsf{mod}(hi+j-i,s)$. This leads to a unique $j = \mathsf{mod}(l - hi + i, s)$ for each h.

SMNO Property. We prove that each multi-share from cluster h_1 intersects exactly once with each multi-share from cluster h_2: $|\mathbf{A}_{j_1}^{h_1} \cap \mathbf{A}_{j_2}^{h_2}| = 1$. For any two multi-shares j_1 and j_2, there exists a pair of indices (i_1, i_2) satisfying:

$$i_1 s + \mathsf{mod}(h_1 i_1 + j_1 - i_1, s) = i_2 s + \mathsf{mod}(h_2 i_2 + j_2 - i_2, s). \tag{16}$$

Applying $\lfloor ./s \rfloor$ to both sides gives $i_1 = i_2$. Replacing i_1 and i_2 with i, we get:

$$is + \mathsf{mod}(h_1 i + j_1 - i, s) = is + \mathsf{mod}(h_2 i + j_2 - i, s). \tag{17}$$

Solving for i yields

$$i = (h_1 - h_2)^{-1}(j_2 - j_1) \mod s.$$

With s being prime, $(h_1 - h_2)^{-1}$ is always unique, leading to:

$$|\mathbf{A}_{j_1}^{h_1} \cap \mathbf{A}_{j_2}^{h_2}| = 1.$$

Furthermore, the rows cluster and each cluster defined by (15) are MNO. The r-th multi-share in the rows cluster, which is the r-th row, contains shares from indices rs to $rs+s-1$. Each multi-share from other clusters includes exactly one share in the range rs to $rs+s-1$, located at index $i = r$. Thus, each multi-share from the rows cluster intersects exactly once with each multi-share from other clusters.

There Are No Other SMNO Clusters. For the sake of contradiction, assume there exists an unaccounted cluster $\mathcal{C}'$, and $\mathbf{B}'$ is one of its multi-shares. Imagine shares in a square of side length s. $\mathbf{B}'$ must have one intersection with the first row and one with the second row. Let c_1 and c_2 be the respective column indices of these intersections. Using the structure defined in (15), we can show that in the cluster with $h = \mathsf{mod}(c_2 - c_1 + 1, s)$, the first two members of c_1-th multi-share coincide with $\mathbf{B}_0$. Consequently,

$$|\mathbf{A}_{c_1}^{\mathsf{mod}(c_2-c_1+1,s)} \cap \mathbf{B}_0| > 1.$$

This contradiction indicates that no additional cluster $\mathcal{C}'$ exists outside the defined structure.

Composite s. For non-prime s, we seek an as large as possible set $\mathbf{T}$ where $\mathsf{gcd}(h_1 - h_2, s) = 1$ for any distinct $h_1, h_2 \in \mathbf{T}$, ensuring that $h_1 - h_2$ is invertible modulo s. We always have $0, 1 \in \mathbf{T}$. A third value h is in $\mathbf{T}$ if $\mathsf{gcd}(h, s) = 1$ and $\mathsf{gcd}(h - 1, s) = 1$. Hence, if 2 does not divide s, then $2 \in \mathbf{T}$. Following up on this, let p be the smallest prime factor of a composite s. Then all values up to the smallest p are included in $\mathbf{T}$, incrementally. Let $k < p$ be such that $0, 1, \ldots, k-1 \in \mathbf{T}$. Then we need $\gcd(k-0, s) = \gcd(k-1, s) = \gcd(k-2, s) = \ldots = \gcd(k-(k-1), s) = 1$. $\gcd(k, s) = 1$ is obvious, since $k < p$. All other gcds in the above line are because $k-1, k-2, \ldots, 1 \in \mathbb{T}$. Any value h in $\mathbf{T}$ defines an MNO cluster per (15). That $h_1 - h_2$ is invertible modulo s for distinct $h_1, h_2 \in \mathbf{T}$ ensures that these clusters are SMNO. □

Example 8. For instance, with $n = 16$, we have $s = 2^2$ and $p_1 = 2$. Hence, the number of SMNO clusters, that we can create with the method from Lemma 3 is 3. In general, since 2 is the smallest prime, we have at least three clusters that are SMNO for any value s. This gives a different argument to the one in Remark 1 for having at least 3 SMNO clusters. (We saw that with finite affine geometry, we can create five SMNO clusters for $n = 16$.)

B Instances of $\mathbb{S}$AND-DN, $\mathbb{S}$AND-DU, and $\mathbb{S}\chi_3$

B.1 $\mathbb{S}$AND-DN

In the $n = 9$ configuration, there are four possible clusters that are SMNO. These clusters are identified in (2). To work out an instance of $\mathbb{S}$AND-DN, we choose clusters indexed by $h_1 = 0$ and $h_2 = 1$. The shares in $\mathbf{Z}$ are computed using Algorithm 1 as follows:

$$\begin{aligned}
Z_0 &= (X_0 \oplus X_1 \oplus X_2) \cdot (Y_0 \oplus Y_3 \oplus Y_6) & Z_5 &= (X_3 \oplus X_4 \oplus X_5) \cdot (Y_2 \oplus Y_5 \oplus Y_8) \\
Z_1 &= (X_0 \oplus X_1 \oplus X_2) \cdot (Y_1 \oplus Y_4 \oplus Y_7) & Z_6 &= (X_6 \oplus X_7 \oplus X_8) \cdot (Y_0 \oplus Y_3 \oplus Y_6) \\
Z_2 &= (X_0 \oplus X_1 \oplus X_2) \cdot (Y_2 \oplus Y_5 \oplus Y_8) & Z_7 &= (X_6 \oplus X_7 \oplus X_8) \cdot (Y_1 \oplus Y_4 \oplus Y_7) \\
Z_3 &= (X_3 \oplus X_4 \oplus X_5) \cdot (Y_0 \oplus Y_3 \oplus Y_6) & Z_8 &= (X_6 \oplus X_7 \oplus X_8) \cdot (Y_2 \oplus Y_5 \oplus Y_8) \\
Z_4 &= (X_3 \oplus X_4 \oplus X_5) \cdot (Y_1 \oplus Y_4 \oplus Y_7)
\end{aligned}$$

B.2 $\mathbb{S}$AND-DU

For $\mathbb{S}$AND-DU, we need all four clusters. For these clusters. we use the same indexes as in (2). The output shares computed with Algorithm 2 are as follows:

$$
\begin{aligned}
Z_0 &= (X_0 \oplus X_1 \oplus X_2) \cdot (Y_0 \oplus Y_3 \oplus Y_6) \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_6 \oplus Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_6 \\
Z_1 &= (X_0 \oplus X_1 \oplus X_2) \cdot (Y_1 \oplus Y_4 \oplus Y_7) \oplus X_0 \oplus X_2 \oplus X_4 \oplus X_7 \oplus Y_0 \oplus Y_2 \oplus Y_4 \oplus Y_7 \\
Z_2 &= (X_0 \oplus X_1 \oplus X_2) \cdot (Y_2 \oplus Y_5 \oplus Y_8) \oplus X_0 \oplus X_1 \oplus X_5 \oplus X_8 \oplus Y_0 \oplus Y_1 \oplus Y_5 \oplus Y_8 \\
Z_3 &= (X_3 \oplus X_4 \oplus X_5) \cdot (Y_0 \oplus Y_4 \oplus Y_8) \oplus X_0 \oplus X_3 \oplus X_5 \oplus X_8 \oplus Y_0 \oplus Y_3 \oplus Y_5 \oplus Y_8 \\
Z_4 &= (X_3 \oplus X_4 \oplus X_5) \cdot (Y_1 \oplus Y_5 \oplus Y_6) \oplus X_1 \oplus X_3 \oplus X_4 \oplus X_6 \oplus Y_1 \oplus Y_3 \oplus Y_4 \oplus Y_6 \\
Z_5 &= (X_3 \oplus X_4 \oplus X_5) \cdot (Y_2 \oplus Y_3 \oplus Y_7) \oplus X_2 \oplus X_4 \oplus X_5 \oplus X_7 \oplus Y_2 \oplus Y_4 \oplus Y_5 \oplus Y_7 \\
Z_6 &= (X_6 \oplus X_7 \oplus X_8) \cdot (Y_2 \oplus Y_4 \oplus Y_6) \oplus X_2 \oplus X_4 \oplus X_7 \oplus X_8 \oplus Y_2 \oplus Y_4 \oplus Y_7 \oplus Y_8 \\
Z_7 &= (X_6 \oplus X_7 \oplus X_8) \cdot (Y_1 \oplus Y_3 \oplus Y_8) \oplus X_1 \oplus X_3 \oplus X_6 \oplus X_7 \oplus Y_1 \oplus Y_3 \oplus Y_6 \oplus Y_7 \\
Z_8 &= (X_6 \oplus X_7 \oplus X_8) \cdot (Y_0 \oplus Y_5 \oplus Y_7) \oplus X_0 \oplus X_5 \oplus X_6 \oplus X_8 \oplus Y_0 \oplus Y_5 \oplus Y_6 \oplus Y_8
\end{aligned}
$$

B.3 $\mathbb{S}_{\boldsymbol{\chi_3}}$

For an instance of $\mathbb{S}_{\chi_3}$ with $n = 4$, we choose clusters indexed 0 to 2 in accordance with Example 3. By application of Algorithm 4 and based on input n-sharings $[\mathbf{X}^0, \mathbf{X}^1, \mathbf{X}^2]$, we obtain the following descriptions for the output n-sharings $[\mathbf{Y}^0, \mathbf{Y}^1, \mathbf{Y}^2]$ of $\mathbb{S}_{\chi_3}$:

$$
\begin{aligned}
Y_0^0 &= X_0^0 \oplus \overline{X_0^1} X_0^2 \oplus \overline{X_0^1} X_2^2 \oplus X_1^1 X_0^2 \oplus X_1^1 X_2^2 & Y_1^0 &= X_3^0 \oplus \overline{X_0^1} X_1^2 \oplus \overline{X_0^1} X_3^2 \oplus X_1^1 X_1^2 \oplus X_1^1 X_3^2 \\
Y_2^0 &= X_1^0 \oplus X_2^1 X_0^2 \oplus X_2^1 X_2^2 \oplus X_3^1 X_0^2 \oplus X_3^1 X_2^2 & Y_3^0 &= X_2^0 \oplus X_2^1 X_1^2 \oplus X_2^1 X_3^2 \oplus X_3^1 X_1^2 \oplus X_3^1 X_3^2 \\
Y_0^1 &= X_0^1 \oplus \overline{X_0^2} X_0^0 \oplus \overline{X_0^2} X_2^0 \oplus X_1^2 X_0^0 \oplus X_1^2 X_2^0 & Y_1^1 &= X_3^1 \oplus \overline{X_0^2} X_1^0 \oplus \overline{X_0^2} X_3^0 \oplus X_1^2 X_1^0 \oplus X_1^2 X_3^0 \\
Y_2^1 &= X_1^1 \oplus X_2^2 X_0^0 \oplus X_2^2 X_2^0 \oplus X_3^2 X_0^0 \oplus X_3^2 X_2^0 & Y_3^1 &= X_2^1 \oplus X_2^2 X_1^0 \oplus X_2^2 X_3^0 \oplus X_3^2 X_1^0 \oplus X_3^2 X_3^0 \\
Y_0^2 &= X_3^2 \oplus \overline{X_0^0} X_0^1 \oplus \overline{X_0^0} X_2^1 \oplus X_1^0 X_0^1 \oplus X_1^0 X_2^1 & Y_1^2 &= X_0^2 \oplus \overline{X_0^0} X_1^1 \oplus \overline{X_0^0} X_3^1 \oplus X_1^0 X_1^1 \oplus X_1^0 X_3^1 \\
Y_2^2 &= X_2^2 \oplus X_2^0 X_0^1 \oplus X_2^0 X_2^1 \oplus X_3^0 X_0^1 \oplus X_3^0 X_2^1 & Y_3^2 &= X_1^2 \oplus X_2^0 X_1^1 \oplus X_2^0 X_3^1 \oplus X_3^0 X_1^1 \oplus X_3^0 X_3^1
\end{aligned}
$$

Here, $\overline{X_j^i}$ denotes $1 \oplus X_j^i$. Order of XOR (that is from left to right) is relevant to the probing security.

C Correctness of $\mathbb{S}$AND-DU

We first show that R_k for $0 \leq k \leq n-1$ is an n-sharing of zero. That is $\oplus_{k=0}^{n-1} R_k = 0$. In the following, we demonstrate this property for $\mathbf{X}$. The steps for $\mathbf{Y}$ are similar. We have:

$$
\bigoplus_{k=0}^{n-1} R_k = \bigoplus_{k=0}^{n-1} \bigoplus_{i=0}^{s-1} \left(\mathbf{A}_\alpha^0(\mathbf{X})[i] \oplus \mathbf{A}_\beta^{\alpha+1}(\mathbf{X})[i] \right) \tag{18}
$$

Since there is a one to one map between k and (α, β), we can decompose a summation over k into summations over α and β. That is $\oplus_{k=0}^{n-1}(.) = \oplus_{\alpha=0}^{s-1} \oplus_{\beta=0}^{s-1}$

$(.) = \oplus_{\beta=0}^{s-1} \oplus_{\alpha=0}^{s-1} (.)$. So, we can rewrite (18) as:

$$\begin{aligned}\bigoplus_{k=0}^{n-1} R_k &= \left(\bigoplus_{\beta=0}^{s-1}\bigoplus_{\alpha=0}^{s-1}\bigoplus_{i=0}^{s-1} \mathbf{A}_\alpha^0(\mathbf{X})[i]\right) \oplus \left(\bigoplus_{\alpha=0}^{s-1}\bigoplus_{\beta=0}^{s-1}\bigoplus_{i=0}^{s-1} \mathbf{A}_\beta^{\alpha+1}(\mathbf{X})[i]\right) \\ &\overset{(\mathtt{I})}{=} \left(\bigoplus_{\beta=0}^{s-1} X\right) \oplus \left(\bigoplus_{\alpha=0}^{s-1} X\right) = 0\end{aligned} \tag{19}$$

The right-hand-side of (I) is obtained by the identity $\oplus_{j=0}^{s-1} \oplus_{i=0}^{s-1} \mathbf{A}_j^h(\mathbf{X})[j] = X$ that is valid for any cluster h.

For the summation of the W_k terms, we have:

$$\begin{aligned}\bigoplus_{k=0}^{n-1} W_k &= \bigoplus_{k=0}^{n-1} \left(\mathbf{A}_\alpha^0(\mathbf{X}) \otimes \mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})\right) = \bigoplus_{k=0}^{n-1} \left[\left(\bigoplus_{i=0}^{s-1} \mathbf{A}_\alpha^0(\mathbf{X})[i]\right)\left(\bigoplus_{i=0}^{s-1} \mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})[i]\right)\right] \\ &= \bigoplus_{\alpha=0}^{s-1}\bigoplus_{\beta=0}^{s-1} \left[\left(\bigoplus_{i=0}^{s-1} \mathbf{A}_\alpha^0(\mathbf{X})[i]\right)\left(\bigoplus_{i=0}^{s-1} \mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})[i]\right)\right] \\ &= \bigoplus_{\alpha=0}^{s-1} \left[\left(\bigoplus_{i=0}^{s-1} \mathbf{A}_\alpha^0(\mathbf{X})[i]\right)\left(\bigoplus_{\beta=0}^{s-1}\bigoplus_{i=0}^{s-1} \mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})[i]\right)\right] \\ &= \bigoplus_{\alpha=0}^{s-1} \left[\left(\bigoplus_{i=0}^{s-1} \mathbf{A}_\alpha^0(\mathbf{X})[i]\right) Y\right] = \left(\bigoplus_{\alpha=0}^{s-1}\bigoplus_{i=0}^{s-1} \mathbf{A}_\alpha^0(\mathbf{X})[i]\right) Y = XY\end{aligned} \tag{20}$$

Note that $Z = \oplus_{k=0}^{n-1} Z_k = (\oplus_{k=0}^{n-1} W_k) \oplus (\oplus_{k=0}^{n-1} R_k) = XY$, and this completes the proof of correctness. □

D Proof of the Uniformity of 𝕊AND-DU.

Shares in $\mathbf{Z}$ are non-linearly related to $\mathbf{X}$ and $\mathbf{Y}$. To work around this, let us fix the summations of the shares in the multi-shares of cluster 0 of $\mathbf{X}$. These summations, i.e., values $\oplus\mathbf{A}_i^0(\mathbf{X})$, are the only terms creating non-linear dependencies. With this trick, $\mathbf{Z}$ will be linearly related to $\mathbf{X}$ and $\mathbf{Y}$, and consequently, we can determine the shape of its distribution.

Let, for $0 \le i < s$, the sum of shares in the i-th multi-share of cluster 0 of $\mathbf{X}$ be fixed as:

$$\theta_i = \oplus_{j=0}^{s-1} \mathbf{A}_i^0(\mathbf{X})[j] \overset{\text{def}}{=} \oplus\mathbf{A}_i^0(\mathbf{X}).$$

By substituting the values θ_i into (10), for Z_k, $0 \le k \le n-1$, we have:

$$\begin{aligned}&\alpha = \lfloor k/s \rfloor, \quad \beta = \mathsf{mod}(k, s), \\ &Z_k = \theta_\alpha \left(\oplus\mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})\right) \oplus \theta_\alpha \oplus \left(\oplus\mathbf{A}_\beta^{\alpha+1}(\mathbf{X})\right) \oplus \left(\oplus\mathbf{A}_\alpha^0(\mathbf{Y})\right) \oplus \left(\oplus\mathbf{A}_\beta^{\alpha+1}(\mathbf{Y})\right).\end{aligned} \tag{21}$$

The resulting system of relations is *linear* in both $\mathbf{X}$ and $\mathbf{Y}$. We now show that for each $(n-1)$-element set $\mathbf{Z}_1 \subset \mathbf{Z}$ with realization $\mathbf{z}_1$ and *any* values

$\{\theta_0, \ldots, \theta_{s-1}\}$, there are exactly $2^{u(n-s+1)}$ tuples $(\mathbf{x}, \mathbf{y})$ that map to $\mathbf{z}_1$. Recall that u is the bit-width of the underlying field.
We demonstrate that the collection of all equations of the form $\oplus\mathbf{A}_i^0(\mathbf{X}) = \theta_i$ and at most $n-1$ of the equations Z_k are linearly independent.
In Equation (21), if we just care about $\mathbf{X}$, Z_k has only one parity sum: $\oplus\mathbf{A}_\beta^{\alpha+1}(\mathbf{X})$. By Lemma 6, we know that among s^2+s parity relations in the system of equations: $\{\oplus\mathbf{A}_\beta^{\alpha+1}(\mathbf{X}), \oplus\mathbf{A}_i^0(\mathbf{X})\}$ there are s^2 independent relations. Specifically, the following set of parity equations are linearly independent.

$$\mathcal{B}_X = \{\oplus\mathbf{A}_\beta^{\alpha+1}(\mathbf{X}), \oplus\mathbf{A}_i^0(\mathbf{X}) \mid 0 \leq \alpha, i < s,\ 0 \leq \beta < s-1\}$$

$\mathcal{B}_X$ is defined by omitting the last multi-share of clusters > 0 that are clusters other than the rows cluster. Therefore, we have $|\mathcal{B}_X| = s(s-1)+s = s^2$.

Summing properly chosen Z_k equations, we build other $s-1$ linear equations that annihilate the $\mathbf{X}$-shares and, consequently, are independent of $\mathcal{B}_X$. We call the set of these equations $\mathcal{B}_Y$. We will show that $s-1$ of these equations are linearly independent of each other. $\mathcal{B}_Y$ is as follows.

$$\mathcal{B}_Y = \left\{ \bigoplus_{k=0}^{s-1} Z_k, \bigoplus_{k=s}^{2s-1} Z_k, \ldots, \bigoplus_{k=s^2-s}^{s^2-1} Z_k \right\}.$$

For $0 \leq i < s$, the i-th parity relation in $\mathcal{B}_Y$ is

$$\mathcal{B}_Y[i] = \bigoplus_{k=is}^{(i+1)s-1} Z_k = \theta_i Y \oplus \theta_i \oplus X \oplus \left(\oplus\mathbf{A}_i^0(\mathbf{Y})\right) \oplus Y.$$

Note that $X = \oplus_{i=0}^{s-1}\theta_i$. Hence, $\mathcal{B}_Y[i]$ only has variables of $\mathbf{Y}$ as its unknowns. This makes $\mathcal{B}_X$ and $\mathcal{B}_Y$ linearly independent.

What remains is to show that there are $s-1$ independent relations in $\mathcal{B}_Y$ for any set of θ_i values. If there was no term such as $\theta_i Y \oplus Y = (1 \oplus \theta_i)Y$ in the $\mathcal{B}_Y[i]$ relations, then the required independence relation was obvious, since each $\mathcal{B}_Y[i]$ is using one multi-share of $\mathbf{Y}$ and these multi-shares are disjoint (they are all from the same cluster). Addition of $(1 \oplus \theta_i)Y$ term can at most decrease the rank of $\mathcal{B}_Y$ by one. Therefore, we identified $s^2+s-1 = n+s-1$ linearly independent equations for given $\{\theta_0, \ldots, \theta_{s-1}\}$ and $n-1$ values of $\mathbf{z}_1$ based on $2n$ variables in $\mathbf{X}$ and $\mathbf{Y}$. So, there $2n-(n+s-1) = n-s+1$ *free* variables in $\mathbf{X}$ and $\mathbf{Y}$. That implies there are $2^{u(n-s+1)}$ preimages for any set $\{\theta_0, \ldots, \theta_{s-1}\}$ and values of $\mathbf{z}_1$. Considering only preimages of $\mathbf{z}_1$, we can conclude that for any realization $\mathbf{z}_1$, there are $2^{us}2^{u(n-s+1)} = 2^{u(n+1)}$ preimages, a number which is independent of actual values of $\mathbf{z}_1$, and this proves the aimed uniformity. □