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@proton.me,lejla@cs.ru.nl

Abstract. Masking typically relies on fresh *online* randomness to prevent higher-order side-channel leakage, which is costly to generate on constrained devices. We study the extreme setting of *deterministic masking*, where a circuit can only recycle the entropy already present in the input sharing, and we address the key obstacle of building nonlinear gadgets that are both higher-order secure and *uniform*. Our main tool is a *share-clustering* framework inspired by finite affine geometry, which for $n = s^2$ shares (with s a prime power) yields structured families of clusters with tightly controlled overlaps. Using these clusters, we construct deterministic multiplication gadgets over $\mathbb{F}_{2^u}$: a first gadget, $\mathbb{S}$Mul-DN, achieves $(d = s-1)$-order security in the glitch-extended probing model, and a second gadget, $\mathbb{S}$Mul-DU, restores output uniformity by adding a deterministically generated zero-sharing—without consuming any online randomness. Finally, leveraging a diffusion-based composition rule for bricklayer ciphers, we implement a fully deterministic third-order masking of Ascon in software under the standard probing model. Experiments on a Cortex-M4 target (ChipWhisperer) show no third-order leakage detected by Welch t-tests in our measurement setting over up to 2×10^6 traces.

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

1 Introduction

Side-channel threat. Implementations of cryptographic algorithms often leak information through physical side channels (power, electromagnetism, timing), enabling *side-channel attacks* (SCAs) even when the underlying primitive is secure in a black-box model. A standard countermeasure is *masking*: each sensitive variable is split into n *shares* using *offline* randomness, and computations are carried out on the shares. To avoid unwanted recombination of shares, most masking schemes additionally consume *fresh online randomness* during execution [2, 6]. Generating high-quality online randomness under side-channel constraints [9,13] is costly, so reducing (or eliminating) this requirement has become a major research direction [4, 23, 28, 34].

Deterministic masking and composition. We study the extreme point of this spectrum: *deterministic masking*, where the circuit uses *no online randomness* and can only recycle the entropy already present in the input sharings. This raises two intertwined challenges. First, designing deterministic gadgets for nonlinear operations that achieve meaningful higher-order probing security and remain *uniform*. Second, ensuring that such gadgets can be safely *composed* into larger circuits, despite probe correlations that may build up across gadget boundaries [14,32]. For round-based ciphers with a bricklayer structure, a diffusion layer can mitigate such build-up by mixing the outputs of many gadgets. This idea was formalized for deterministic masking in [28] and is particularly relevant for Ascon [21,35]; see Figure 1.

Probing models. We use the *standard probing model* of Ishai et al. [27], where an adversary may inspect up to d intermediate values. To also capture hardware transient effects, we consider the *glitch-extended* probing model [22], where probing a wire in a combinational cone reveals (within a clock cycle) all its inputs. A common mitigation is to insert synchronization layers limiting simultaneous interactions [8, 22], which does not require online randomness. Our main gadget results are stated in the glitch-extended model, while our full-cipher application to Ascon is analyzed in the (software-oriented) standard probing model.

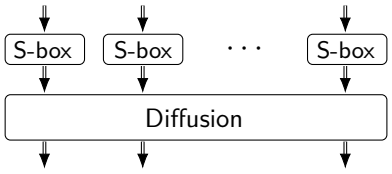

Fig. 1: Round transformation in a cipher.

1.1 An Open Problem: Higher-Order Deterministic Multiplication

Most online randomness in masked implementations is spent on nonlinear operations. The simplest nonlinear operation is multiplication over $\mathbb{F}_2$ (i.e., AND). Classic schemes such as ISW use $n(n-1)/2$ random values to obtain a probing security of $d = \lfloor (n-1)/2 \rfloor$ [27], and later work reduced randomness for higher security levels [4,33]. Belaïd et al. [4] showed that achieving *maximal* $d = n-1$ probing security *without* online randomness is impossible. This leaves open whether deterministic gadgets can achieve *nontrivial* higher orders $d < n-1$. Nikova et al. [30] gave a deterministic ($n = 4, d = 1$) multiplication gadget, but higher-order deterministic constructions remained open in general.

1.2 Our Approach: Share Clustering

Our main technical tool is a systematic way to organize an n-sharing into structured *clusters* of *multi-shares*. A cluster partitions the n shares into s disjoint blocks (multi-shares) of size s (so $n = s^2$). We focus on families of clusters satisfying a strong low-overlap property (SMNO), which enables clean *non-completeness* arguments: a bounded number of probed multi-shares cannot cover

all shares of an input. For s a prime power, such families arise canonically from finite affine planes, yielding $s+1$ pairwise MNO clusters. This clustering viewpoint becomes the backbone of both our gadget design and our security proofs.

1.3 Our Contributions

We make three main contributions.

(1) A systematic clustering framework. We introduce the cluster/multi-share viewpoint and formalize the SMNO/MNO overlap conditions. We give a modular construction of $s+1$ pairwise MNO clusters when s is prime, and connect the general prime-power case to finite affine geometry. We also prove two structural lemmas used throughout the paper: a non-completeness lemma for probed multi-shares, and a linear-independence statement for parity relations induced by clusters.

(2) Higher-order deterministic multiplication with uniform outputs. Using clusters, we first construct a deterministic multiplication gadget $\mathbb{S}$Mul-DN that maximizes glitch-extended probing security for $n = s^2$ shares (order $d = s-1$), but is not uniform. We then introduce our main gadget $\mathbb{S}$Mul-DU, which adds a deterministically generated zero-sharing to restore *uniformity* while preserving $d = s-1$ glitch-extended probing security over $\mathbb{F}_{2^u}$. This provides (to our knowledge) the first deterministic multiplication gadget beyond the $(n = 4, d = 1)$ setting of [30].

Table 1 positions our $\mathbb{S}$Mul-DN and $\mathbb{S}$Mul-DU against prior work.

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 [33]	n	$\frac{n(n-1)}{2}$	$n-1$	✓	$2(n^2-n)$	n^2	×	✓
Belaïd et al. [4]	n	$\approx \frac{n^2+2n}{4}$	$n-1$	✓	$\approx \frac{7n^2}{4}$	n^2	×	✓
Nikova et al. [30]	4	0	1	✓	24	4	✓	×
De Cnudde et al. [19]	6	7	2	✓	29	36	✓	×
$\mathbb{S}$Mul-DN [This work]	n	0	$\sqrt{n}-1$	×	$2(n-\sqrt{n})$	n	✓	×
$\mathbb{S}$Mul-DU [This work]	n	0	$\sqrt{n}-1$	✓	$2\sqrt{n}(n-1)$	n	✓	×

Table 1: Comparison of masking algorithms for secure computation of the AND operation. The columns indicate: (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 resistance; (H) whether the security proof avoids assuming input share independence.

(3) Application to Ascon without online randomness. We instantiate the framework for Ascon [21,35]. We design a deterministic $n = 4$ masking of the χ_5 layer (hence the Ascon S-box) achieving $d = 3$ *standard probing* security and preserving bijectivity/uniformity for the masked S-box. Combined with the diffusion-based composition rule from [28], this yields a deterministic masked Ascon permutation. We further provide a software implementation and empirical side-channel evaluation.[1]

Table 2 positions our deterministic mask implementation of Ascon against prior work.

Candidate	(A)	(B)	(C)	(D)
Daemen et al. [17]	2	✓	1	Verified for one round (using maskVerif [1])
Gigerl et al. [26]	3	✓	2	Verified for one round (using CoCo [25])
Jahandideh et al. [28]	2,3	✓	1,2	Proved for multiple rounds
This work	4	✓	3	Proved using the composition theorem of [28]

Table 2: Comparison of deterministic masking methods for Ascon. The columns indicate: (A) sharing order; (B) no requirement for online randomness; (C) probing security order; (D) type of security proof.

Organization. Section 2 recalls probing models and notation. Section 3 introduces clusters, SMNO families, and their finite affine geometry interpretation. Section 4 presents $\mathbb{S}$Mul-DN and $\mathbb{S}$Mul-DU and analyzes their properties, followed by a discussion on composability (Section 4.5). Section 5 applies the constructions to Ascon, including our masked χ_5 design. Section 5.4 reports implementation and evaluation results, and we conclude in Section 6.

2 Preliminaries

Notation. Random variables are denoted by uppercase letters (e.g., X), and their values/realizations by lowercase letters (e.g., x). Boldface symbols (e.g., $\mathbf{X}$) denote tuples/lists of random variables, such as a vector of shares of a secret. For an event E, we write $\Pr[E]$, and in particular $\Pr[X = x]$ for the probability that X takes value x.

We denote circuits, gadgets, and functions using a sans-serif font (e.g., X). To refer to the masked counterpart of an algorithm, we prefix $\mathbb{S}$ to its name (e.g., $\mathbb{S}\mathsf{X}$), pronounced "shared X". For a list $\mathbf{X}$, its length is denoted by $|\mathbf{X}|$. Given an index set $\mathbf{L} \subseteq \{0, \ldots, |\mathbf{X}| - 1\}$, we write $\mathbf{X}[\mathbf{L}]$ for the sublist of entries with indices in $\mathbf{L}$, and $\mathbf{X}[i]$ for the i-th entry of $\mathbf{X}$ (with i starting at 0).

[1] Available through https://github.com/vahid-jahandideh/deterministic-masking-ascon

2.1 Boolean Masking

Let X be a random variable over the finite field $\mathbb{F}_{2^u}$. Boolean masking represents X by a list of n *shares*

$$\mathbf{X} = [X_0, X_1, \ldots, X_{n-1}] \in (\mathbb{F}_{2^u})^n,$$

such that X is recovered by field addition (which we denote by $\oplus$):

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

Concretely, to generate an n-sharing of X, we sample $X_0, \ldots, X_{n-2} \xleftarrow{\$} \mathbb{F}_{2^u}$ independently and uniformly, and set $X_{n-1} \leftarrow X \oplus \bigoplus_{i=0}^{n-2} X_i$. We call $\mathbf{X}$ an *n-sharing* of X. The parameter u indicates the bit-width of the field, where $u = 1$ corresponds to $\mathbb{F}_2$.

Given a circuit C, a masked implementation $\mathbb{S}\mathsf{C}$ is obtained by n-sharing each input and replacing every gate in C by its corresponding masked sub-circuit, commonly called a *gadget*. We refer to the variables of C as *native* variables. Some of them may be secret (e.g., a key K), while others may be public (e.g., nonces or IVs). Throughout this work, we assume that all values manipulated inside $\mathbb{S}\mathsf{C}$ are represented in shared form; in particular, a public value can be encoded by choosing random shares whose $\oplus$-sum is equal to that value.

Gadgets. Let $Y = \mathsf{G}(X)$ be an affine function over $\mathbb{F}_{2^u}$. We write $\mathsf{G}(x) = \mathsf{L}(x) \oplus c$, where L is $\mathbb{F}_{2^u}$-linear and $c = \mathsf{G}(0)$. Given an n-sharing $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ of X, a natural gadget $\mathbb{S}\mathsf{G}$ outputs an n-sharing $\mathbf{Y} = [Y_0, \ldots, Y_{n-1}]$ of Y by applying the linear part share-wise and injecting the constant into a single share:

$$Y_0 \leftarrow \mathsf{L}(X_0) \oplus c, \qquad Y_i \leftarrow \mathsf{L}(X_i) \quad (1 \le i \le n-1).$$

For nonlinear operations, most notably field multiplication (which we denote by Mul), securely computing output shares is substantially more challenging. Several arbitrary-order $\mathbb{S}\mathsf{Mul}$ gadgets are known [5,10,27,36], but they typically rely on fresh online randomness. Prior to this work, deterministic constructions for $\mathbb{S}\mathsf{Mul}$ gadgets were essentially limited to first-order TI-style settings (e.g., the scheme of Nikova et al. [30] for $u = 1$ and $d = 1$).

TI $\mathbb{S}\mathsf{AND}$ [30]. Here $u = 1$, so Mul coincides with AND. The $\mathbb{S}\mathsf{AND}$ gadget takes 4-sharings $\mathbf{X} = [X_0, X_1, X_2, X_3]$ and $\mathbf{Y} = [Y_0, Y_1, Y_2, Y_3]$ of secrets 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:

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

This gadget satisfies the following key properties:

– **Correctness:** For all input share tuples $(\mathbf{X}, \mathbf{Y}) \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$,

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

– **Non-completeness:** Each output share Z_i is independent of at least one share of $\mathbf{X}$ and at least one share of $\mathbf{Y}$ (i.e., it does not depend on all shares of any input variable). Consequently, any intermediate signal inside the combinational logic computing a fixed Z_i is also non-complete.
– **Uniformity:** For any fixed input values $(X = x, Y = y)$ and uniformly random Boolean sharings of X and Y, the resulting 4-sharing $\mathbf{Z}$ of xy is uniform, conditioned on $Z = xy$; that is, for all $\mathbf{z} \in \mathbb{F}_2^4$ with $\bigoplus_{i=0}^{3} z_i = xy$, we have $\Pr[\mathbf{Z} = \mathbf{z} \mid Z = xy] = 2^{-3}$.

2.2 Probe Elimination and Composition

We now recall the probe-elimination methodology [12, 28] we use throughout to argue (standard) probing security of deterministic circuits.

Probe Elimination via Input Shares. Let B be an n-shared secret with shares $B_0, \ldots, B_{n-1}$, and let $P = B_i \oplus \mathsf{F}$ be a probed value in a probe set $\mathcal{P}$, where F is a (deterministic) function of intermediate variables. Assume that:

1. the share B_i does not appear in the computation of F nor in any other probed value in $\mathcal{P}$;
2. the probe set $\mathcal{P}$ does not contain all n shares of B.

Then B_i is uniform and independent of $(\mathsf{F}, \mathcal{P} \setminus \{P\})$, and hence $P = B_i \oplus \mathsf{F}$ is uniform and independent of $(\mathsf{F}, \mathcal{P} \setminus \{P\})$. So, P can be *eliminated*, i.e., replaced by an *independent uniform* variable, without affecting the adversary's view.

Glitch Resistance via Non-completeness. Recall that a glitch-extended probe on a wire reveals *all* intermediate values in its (combinational) computation cone. Therefore, when reasoning in the glitch-extended probing model, a probe set effectively reveals the union of cones of the probed wires.

We call a set of revealed values *non-complete* (w.r.t. a secret sharing) if, for every n-shared secret B with shares $\mathbf{B} = [B_0, \ldots, B_{n-1}]$, the revealed set does not contain all shares $B_0, \ldots, B_{n-1}$. Intuitively, if at least one share of every secret never appears in the revealed cones, then glitches do not enable reconstruction of the secrets.

We say that a deterministic gadget is *d-order glitch-resistant* if, for any set of at most d glitch-extended probes, the union of the revealed computation cones is non-complete; i.e., it does not collectively include all n shares of any secret.

Example 1. Let $\mathbf{X} = [X_0, X_1, X_2]$ and $\mathbf{Y} = [Y_0, Y_1, Y_2]$ be 3-sharings and define $P_0 = (X_0 \oplus X_1)(Y_0 \oplus Y_1)$ and $P_1 = (X_1 \oplus X_2)(Y_1 \oplus Y_2)$. While $\{P_0, P_1\}$ is independent of $\{X, Y\}$ in the standard probing model, in the glitch-extended model

a probe on P_0 reveals X_0, X_1, Y_0, Y_1 and a probe on P_1 reveals X_1, X_2, Y_1, Y_2, so together they expose all shares of X and Y.

Since glitch-extended probes reveal a superset of the information available in the standard probing model, the achievable security order in the glitch-extended model is at most the standard probing security order.

Composition in Deterministic Masking. In the absence of online randomness, multi-round security relies on the fact that reused randomness can remain *indistinguishable* from fresh randomness to a bounded adversary. For example, the triple $(R_1, R_2, R_1 \oplus R_2)$ is *2-wise independent*: any two of its components are jointly uniform, and hence indistinguishable (to an adversary limited to two probes) from any two components of (R_1, R_2, R_3) where R_3 is an independent random bit. This principle extends to input sharings in round-based ciphers.

Modern ciphers often use a *bricklayer* architecture [18], comprising parallel S-box layers followed by a linear diffusion layer. In the masked setting, an appropriate diffusion layer can act as an *implicit re-encoding* step, so that reusing input shares across rounds remains indistinguishable to an adversary limited to d probes (Figure 1).

Definition 1 (Independent n-Sharings [28]). *A collection of n-sharings $\{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$ is said to be* independent *if, for any choice of $n-1$ shares from each $\mathbf{X}_i$, the resulting $m(n-1)$ variables are jointly independent.*

Higher-Order Security via Diffusion. Let b denote the number of S-boxes per nonlinear layer, and represent the layer input as $S = (S[0], \ldots, S[b-1])$, where $S[j]$ is the input word of the j-th S-box. Let $\mathbf{M}$ denote the (linear) diffusion layer acting on these b words, and write $S\mathbf{M}^\top$ for the output vector.

Definition 2 (Box Linear Branch Number [18]). *The* box linear branch number $\mathcal{B}_{l\text{-}box}$ *is defined as*

$$\mathcal{B}_{l\text{-}box} = \min_{S \neq 0} \Big(\mathrm{wt}_{box}(S) + \mathrm{wt}_{box}(S\mathbf{M}^\top) \Big),$$

where $\mathrm{wt}_{box}(S)$ *denotes the number of indices j such that $S[j] \neq 0$ (i.e., the number of active* S-boxes*).*

The following result from [28] ensures secure composition of two rounds in deterministic masking. Moreover, leveraging the diffusion properties, [28] argues that analyzing two consecutive rounds captures the bottleneck, and that adding further rounds does not reduce the overall probing security order.

Lemma 1 (Composition Theorem [28]). *Let $d_{\mathbb{S}\text{S-box}}$ denote the local probing security order of each $\mathbb{S}$S-box gadget, and let $\mathcal{B}_{l\text{-}box}$ be the box linear branch number of the diffusion layer. Then the global probing security order d satisfies*

$$\min\{d_{\mathbb{S}\text{S-box}},\, \mathcal{B}_{l\text{-}box} - 1\} \;\leq\; d \;\leq\; d_{\mathbb{S}\text{S-box}}.$$

In Ascon, $\mathcal{B}_{l\text{-box}} = 4$ [21]. Thus, achieving third-order global probing security reduces to obtaining a deterministic S-box with local order $d_{\mathbb{S}\text{S-box}} = 3$, which is the main gap addressed in this work.

3 Share Clustering

Deterministic masking faces a recurring tension: without online randomness, one must enforce (i) *probing security* (via non-completeness / probe elimination) and (ii) *uniformity* of the resulting sharings, purely through structural properties of how shares are combined. To address this, we introduce the notion of *share clustering*, which organizes an n-sharing into structured groups. Clustering will be our main tool to construct deterministic multiplication gadgets and analyze their security.

Clustering of an n-sharing. Throughout this section we focus on the case where n is a perfect square. Write $n = s^2$ (the smallest non-trivial instance being $n = 4$), and consider an n-sharing $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ of a value X. A *cluster* partitions the n shares into s disjoint groups, each of size s.

Definition 3 (Cluster). *Let $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ be an n-sharing with $n = s^2$. A* cluster *of $\mathbf{X}$ is a partition*

$$\mathcal{C}^{(j)}(\mathbf{X}) \;=\; [\,\mathbf{X}_0^{(j)}, \ldots, \mathbf{X}_{s-1}^{(j)}\,]$$

of the shares into s disjoint multi-shares*, where each multi-share $\mathbf{X}_i^{(j)}$ is a list of length s. Equivalently, the following hold:*

- *(size) $|\mathbf{X}_i^{(j)}| = s$ for all $0 \le i < s$;*
- *(cover) $\bigcup_{i=0}^{s-1} \mathbf{X}_i^{(j)}$ equals the underlying set of shares of $\mathbf{X}$;*
- *(disjointness) $\mathbf{X}_i^{(j)} \cap \mathbf{X}_{i'}^{(j)} = \emptyset$ for all $i \neq i'$.*

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

$$\mathcal{C}^{(0)}(\mathbf{X}) = \Big[\, \mathbf{X}_0^{(0)} = [X_0, X_1],\ \mathbf{X}_1^{(0)} = [X_2, X_3] \,\Big]. \tag{1a}$$

Another distinct cluster of the same shares is

$$\mathcal{C}^{(1)}(\mathbf{X}) = \Big[\, \mathbf{X}_0^{(1)} = [X_0, X_2],\ \mathbf{X}_1^{(1)} = [X_1, X_3] \,\Big]. \tag{1b}$$

We will see that these two clusters are *maximally non-overlapping (MNO).*

Definition 4 (MNO clusters). *Two clusters $\mathcal{C}^{(j)}(\mathbf{X})$ and $\mathcal{C}^{(j')}(\mathbf{X})$ are called* maximally non-overlapping (MNO) *if every multi-share from the first intersects every multi-share from the second in exactly one share, i.e.,*

$$\big|\, \mathbf{X}_i^{(j)} \cap \mathbf{X}_{i'}^{(j')} \,\big| = 1 \qquad \textit{for all } 0 \le i, i' < s.$$

In Example 2, observe for instance that

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

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

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

can be formed such that any two clusters in the set

$$\mathfrak{C} = \{\mathcal{C}^{(0)}(\mathbf{X}), \mathcal{C}^{(1)}(\mathbf{X}), \mathcal{C}^{(2)}(\mathbf{X})\}$$

are MNO.

Definition 5 (Simultaneous MNO clusters). *Let $\mathfrak{C}$ be a collection of clusters of $\mathbf{X}$, $\mathfrak{C} = \{\mathcal{C}^{(0)}(\mathbf{X}), \ldots, \mathcal{C}^{(|\mathfrak{C}|-1)}(\mathbf{X})\}$. We call the clusters in $\mathfrak{C}$* simultaneous MNO (SMNO) *if every pair of distinct clusters in $\mathfrak{C}$ is MNO.*

3.1 Maximum Number of SMNO Clusters

For the smallest non-trivial case $n = 4$ (i.e., $s = 2$), the maximum number of pairwise MNO clusters is 3, hence at most 3 SMNO clusters exist. This motivates investigating the maximum number of SMNO clusters for larger square values $n = s^2$.

Example 3 ($n = 9$). To visualize potential clusters for $n = 9$ (so $s = 3$), we arrange the nine shares in a 3×3 array, as shown in Figure 2. This representation helps identify clusters in which each multi-share consists of three shares.

In this 9-sharing setting, one can construct four pairwise MNO clusters by taking the row partition, the column partition, and two additional "diagonal" directions (with wrap-around indexing modulo 3). Concretely, for $\mathbf{X} = [X_0, \ldots, X_8]$, these clusters are:

$$\mathcal{C}^{(0)}(\mathbf{X}) = \big[[X_0, X_1, X_2],\ [X_3, X_4, X_5],\ [X_6, X_7, X_8] \big], \tag{2a}$$

$$\mathcal{C}^{(1)}(\mathbf{X}) = \big[[X_0, X_3, X_6],\ [X_1, X_4, X_7],\ [X_2, X_5, X_8] \big], \tag{2b}$$

$$\mathcal{C}^{(2)}(\mathbf{X}) = \big[[X_0, X_4, X_8],\ [X_1, X_5, X_6],\ [X_2, X_3, X_7] \big], \tag{2c}$$

$$\mathcal{C}^{(3)}(\mathbf{X}) = \big[[X_0, X_5, X_7],\ [X_1, X_3, X_8],\ [X_2, X_4, X_6] \big]. \tag{2d}$$

Other values of n. For $n = 4^2 = 16$, one can form five pairwise MNO clusters (see Figure 3). In contrast, for $n = 6^2 = 36$, only three such clusters are possible. Thus, the maximum number of SMNO clusters is not a straightforward (and in particular not monotonic) function of n.

Remark 1. For any $n = s^2$ with $s \geq 2$, there always exist at least three pairwise MNO clusters of an n-sharing: the row cluster, the column cluster, and the "diagonal" cluster of slope $+1$ (with wrap-around indexing modulo s). Hence, for every square n, there are at least three SMNO clusters.

The "anti-diagonal" direction (slope -1) is not always compatible with the diagonal direction: when s is even (e.g., $s = 2$ or $s = 4$), diagonal and anti-diagonal clusters can intersect in two shares, so they cannot both be included in a set of pairwise MNO clusters.

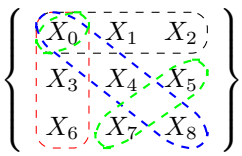

Fig. 2: First multi-share of each of the four SMNO clusters of $\mathbf{X}$ for $n = 9$.

In the following sections, we will make use of *all* available SMNO clusters in our design. We therefore want to understand the maximum number of such clusters and how to construct them systematically.

A Systematic Clustering Structure. A convenient way to define clusters is to lay out the shares of an n-sharing $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ in an $s \times s$ array (with $s = \sqrt{n}$), using the linear index mapping X_{rs+c} for row r and column c.

We define the *row cluster* as $\mathcal{C}^{(0)}(\mathbf{X}) = [\mathbf{X}_0^{(0)}, \ldots, \mathbf{X}_{s-1}^{(0)}]$, where

$$\mathbf{X}_j^{(0)} \;=\; [\, X_{js},\, X_{js+1},\, \ldots,\, X_{js+(s-1)}\,] \qquad \text{for } 0 \le j \le s-1.$$

For each value of $h \in \{1, \ldots, s\}$, we define an additional cluster $\mathcal{C}^{(h)}(\mathbf{X}) = [\mathbf{X}_0^{(h)}, \ldots, \mathbf{X}_{s-1}^{(h)}]$ by

$$\mathbf{X}_j^{(h)} \;=\; \big[\, X_{is+\mathsf{mod}((h-1)i+j,\, s)}\,\big]_{i=0}^{s-1}, \tag{3}$$

where $\mathsf{mod}(a, s)$ denotes the remainder of a modulo s. Note that $h = 1$ corresponds to columns, $h = 2$ to the wrap-around diagonals of slope $+1$, and $h = s$ to slope -1. Altogether, this construction yields $s + 1$ candidate clusters[2]

Lemma 2. *Let $n = s^2$ and let $\mathcal{C}^{(0)}(\mathbf{X}), \mathcal{C}^{(1)}(\mathbf{X}), \ldots, \mathcal{C}^{(s)}(\mathbf{X})$ be the $s+1$ clusters defined by the row cluster and by Equation (3). If s is prime, then these $s + 1$ clusters are pairwise MNO (hence form an SMNO family of size $s + 1$).*

Proof. For $h \in \{1, \ldots, s\}$, consider $\mathcal{C}^{(h)}(\mathbf{X}) = [\mathbf{X}_0^{(h)}, \ldots, \mathbf{X}_{s-1}^{(h)}]$ as defined in (3).

Each $\mathcal{C}^{(h)}(\mathbf{X})$ is a cluster. Write any index $l \in \{0, \ldots, s^2 - 1\}$ uniquely as $l = is + c$ with $i \in \{0, \ldots, s-1\}$ and $c = \mathsf{mod}(l, s)$. For the unique $j = \mathsf{mod}(c - (h-1)i, s)$ we have $c = \mathsf{mod}((h-1)i + j, s)$, hence $X_l = X_{is+c}$ appears in $\mathbf{X}_j^{(h)}$ at position i. Moreover, for fixed i the map $j \mapsto \mathsf{mod}((h-1)i + j, s)$ is a bijection, so different j's give disjoint multi-shares. Thus $\mathcal{C}^{(h)}(\mathbf{X})$ partitions all shares into s disjoint multi-shares of size s.

MNO between two modular clusters. Let $h_1 \neq h_2$ in $\{1, \ldots, s\}$ and pick any multi-share indices j_1, j_2. A share lies in $\mathbf{X}_{j_1}^{(h_1)} \cap \mathbf{X}_{j_2}^{(h_2)}$ iff there exists $i \in \{0, \ldots, s-1\}$ such that

$$\mathsf{mod}((h_1 - 1)i + j_1, s) = \mathsf{mod}((h_2 - 1)i + j_2, s),$$

equivalently

$$(h_1 - h_2)i \equiv (j_2 - j_1) \pmod{s}.$$

Since s is prime, $h_1 - h_2 \not\equiv 0 \pmod{s}$ is invertible, so this congruence has a unique solution i modulo s. Hence $|\mathbf{X}_{j_1}^{(h_1)} \cap \mathbf{X}_{j_2}^{(h_2)}| = 1$.

[2] Not necessarily all pairwise MNO when s is not prime.

MNO with the row cluster. The row cluster $\mathcal{C}^{(0)}(\mathbf{X})$ groups shares by fixed row index i. For any $h \in \{1, \ldots, s\}$ and any j, the multi-share $\mathbf{X}_j^{(h)}$ contains exactly one share from each row (one per i), so it intersects each row multi-share in exactly one share. Thus $\mathcal{C}^{(0)}(\mathbf{X})$ is MNO with every $\mathcal{C}^{(h)}(\mathbf{X})$.

Combining the above, all $s+1$ clusters are pairwise MNO. □

Remark 2 (Prime powers and affine geometry). When s is a prime power, one can construct $s+1$ pairwise MNO clusters using the affine-plane viewpoint over $\mathbb{F}_s$ (parallel classes of lines). We discuss this construction in Section 3.2. For s prime, this affine-plane construction coincides with the modular construction above.

Remark 3 (Composite s under integer modular slopes). If one insists on the integer-modulo construction (3) for composite s, then two slopes $h_1 \neq h_2$ are guaranteed to be MNO only when $\gcd(h_1 - h_2, s) = 1$. In particular, letting p be the smallest prime divisor of s, the set of slopes $\{0, 1, \ldots, p\}$ yields at least $p+1$ pairwise MNO clusters. This lower bound is generally not tight, and for prime powers $s = p^e$ the affine-plane construction recovers $s+1$ clusters.

3.2 Finite Affine Geometry

In this section, we explain how SMNO *clusters* naturally arise from finite affine geometry. For background on finite affine planes we refer to [3]. The key takeaway is that, when $n = s^2$ and s is a prime power, the (finite) affine plane over $\mathbb{F}_s$ provides a canonical family of $s+1$ pairwise MNO *clusters* (parallel classes).

Affine planes and a dictionary to clusters. An affine plane of order s has s^2 *points* and $s^2 + s$ *lines*, each line containing exactly s points, and each point lying on exactly $s+1$ lines. Moreover, lines fall into $s+1$ *parallel classes*, each consisting of s pairwise disjoint lines; and any two non-parallel lines intersect in exactly one point. These facts follow from the axioms of affine planes (see below).

We use the following correspondence:

- points $\longleftrightarrow$ shares of an n-sharing (so $n = s^2$);
- a line $\longleftrightarrow$ a *multi-share* (a block of s shares);
- a parallel class of lines $\longleftrightarrow$ a *cluster* $\mathcal{C}^{(j)}(\mathbf{X})$ (a partition of all shares into s disjoint multi-shares);
- two distinct parallel classes $\longleftrightarrow$ two *MNO clusters* (every line from one meets every line from the other in exactly one point).

In a set of SMNO clusters, the parallel classes correspond to clusters, showing that if we have an affine plane of order s, we have $s+1$ SMNO clusters.

Axioms of affine geometry. The axioms for a finite affine plane are:

$\mathcal{A}1$ For any two distinct points, there is a unique line that contains both;
$\mathcal{A}2$ For each line L and point P not on L, there is a unique line L' that contains P and is disjoint from L;
$\mathcal{A}3$ There exists a set of four points such that no three of them lie on a single line.

The second axiom ($\mathcal{A}2$) is often called Playfair's axiom [31]. Axiom $\mathcal{A}1$ follows from the fact that the clusters give a full SMNO family (uniqueness). Axiom $\mathcal{A}2$ works in any single cluster (set of parallel lines). For Axiom $\mathcal{A}3$ we can take $\{\mathbf{X}_0^{(0)}[0], \mathbf{X}_0^{(0)}[1], \mathbf{X}_1^{(0)}[0], \mathbf{X}_2^{(0)}[1]\}$. By design, these are not all in a single line in cluster $\mathcal{C}^{(0)}$, since $\mathbf{X}_j^{(0)} \cap \mathbf{X}_0^{(0)} = \emptyset$ for all $j \neq 0$. Furthermore, these points are not all on any line in $\mathcal{C}^{(j)}$ with $j \neq 0$, since then $|\mathbf{X}_0^{(0)} \cap \mathbf{X}_k^{(j)}| \geq 2$.

Connection to our modular construction. When s is prime, the $s+1$ clusters produced by Lemma 2 coincide with the $s+1$ parallel classes of the affine plane. For prime powers $s = p^e$, a similar construction works by working over the field $\mathbb{F}_s$ (we do *not* rely on integer modular slopes when $e > 1$).

Proposition 1 (Adapted from [3] §4.6). *If s is a prime power, then an affine plane with $n = s^2$ points exists and yields an SMNO family of size $s+1$, i.e., $s+1$ pairwise MNO clusters of an n-sharing with $n = s^2$.*

Remark 4. It is a long-standing open problem in finite geometry whether an affine plane of order s exists only when s is a prime power. In this work, assuming s is a prime power suffices and matches all instances we use.

Example: $s = 4$ and $n = 16$. Figure 3 illustrates the case $s = 4$ (hence $n = s^2 = 16$ shares) where $s + 1 = 5$ SMNO clusters exist. This is an affine plane of order 4: it has $4^2 = 16$ points and $4 + 1 = 5$ parallel classes.

We use the finite field $\mathbb{F}_4 := \mathbb{F}_2(\alpha) = \mathbb{F}_2[X]/(X^2 + X + 1)$. The points are elements of $\mathbb{F}_4 \times \mathbb{F}_4$, and each parallel class corresponds to one of:

$$\ell_{\lambda,b} = \{(x, y) \in \mathbb{F}_4^2 \mid y = \lambda x + b\} \quad \text{for } \lambda, b \in \mathbb{F}_4,$$
$$\ell_a = \{(a, y) \mid y \in \mathbb{F}_4\} \quad \text{for } a \in \mathbb{F}_4.$$

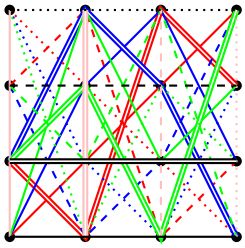

Fig. 3: Affine plane of order 4: five parallel classes, hence 5 SMNO clusters for $n = 16$.

Lines of the same color in Figure 3 form one parallel class (one cluster), i.e., they are pairwise disjoint, while lines of different colors intersect in exactly one point (the MNO property between the corresponding clusters). For instance, the red double line corresponds to $\ell_{1,1}$, and the blue double line corresponds to $\ell_{\alpha+1,1}$.

Remark 5. We use affine planes as a systematic way to obtain large SMNO families when s is a prime power. We do not claim anything for when s is not a prime power; there may be other approaches to tackle those cases.

3.3 Non-completeness of Multi-Shares

The cluster-based view introduced so far simplifies reasoning about probing security. In this subsection, we adopt the following convenient abstraction: a glitch-extended probe is assumed to reveal all elementary shares belonging to one *multi-share*. (Equivalently, probing a signal inside a multi-share "covers" that entire multi-share.)

Lemma 3. *Let* $\mathbf{X}$ *be an* n*-sharing with* $n = s^2$*, and fix a cluster* $\mathcal{C}^{(j)}(\mathbf{X}) = [\mathbf{X}_0^{(j)}, \ldots, \mathbf{X}_{s-1}^{(j)}]$*. Assume the adversary probes* $s-1$ *multi-shares of this cluster, i.e., all but one* $\mathbf{X}_{i^\star}^{(j)}$*. Also assume that every other probed multi-share belongs to a cluster that is MNO with* $\mathcal{C}^{(j)}(\mathbf{X})$*. If the total number of probed multi-shares is at most* $2s-2$*, then the adversary does not recover all* n *shares of* $\mathbf{X}$*.*

Proof. After probing $s-1$ multi-shares of $\mathcal{C}^{(j)}(\mathbf{X})$, the only shares of $\mathbf{X}$ that may remain unrevealed are the s shares contained in $\mathbf{X}_{i^\star}^{(j)}$.

By the MNO property, any multi-share from a cluster that is MNO with $\mathcal{C}^{(j)}(\mathbf{X})$ intersects $\mathbf{X}_{i^\star}^{(j)}$ in at most one share. Therefore, each additionally probed multi-share can reveal at most one share from $\mathbf{X}_{i^\star}^{(j)}$. With at most $(2s-2) - (s-1) = s-1$ additional probes, the adversary can cover at most $s-1$ shares of $\mathbf{X}_{i^\star}^{(j)}$, so at least one share remains unrecovered. □

3.4 Linear Independence of Clusters

For odd prime powers $s = \sqrt{n}$, SMNO clusters exhibit an algebraic structure that we will exploit to prove uniformity of the output sharings in our gadgets. Let $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ be an n-sharing with $n = s^2$. Assume we have an SMNO family of $s+1$ clusters

$$\mathfrak{C} = \{\mathcal{C}^{(0)}(\mathbf{X}), \ldots, \mathcal{C}^{(s)}(\mathbf{X})\}, \qquad \mathcal{C}^{(h)}(\mathbf{X}) = [\mathbf{X}_0^{(h)}, \ldots, \mathbf{X}_{s-1}^{(h)}],$$

where each multi-share $\mathbf{X}_j^{(h)}$ contains exactly s elementary shares.

For a multi-share $\mathbf{X}_j^{(h)}$, we write $\oplus\mathbf{X}_j^{(h)}$ for the XOR-sum (i.e., $\mathbb{F}_{2^u}$-sum) of all shares in that multi-share. For example, when $n = 9$, $\oplus\mathbf{X}_0^{(0)} = X_0 \oplus X_1 \oplus X_2$.

Key observation (rank intuition). Consider the collection of parity sums

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

which yields $s(s+1)$ linear relations in the $n = s^2$ variables $X_0, \ldots, X_{n-1}$. For every cluster $\mathcal{C}^{(h)}(\mathbf{X})$,

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

where X is the secret shared by $\mathbf{X}$. Hence, the total XOR of the s multi-share parities is the same for all clusters, inducing s evident dependencies across the $s+1$ clusters. Importantly, we prove below that *no further dependencies occur.*

Example 4. (Case $n = 9$, $s = 3$) Recall the clusters from Equations (2). Since $s = 3$ is an odd prime, we expect a total rank of 9. One may select the following 9 equations (dropping one parity from each new cluster after the first) to obtain an independent set:

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

These nine equations are linearly independent, giving rank 9. □

General case for odd prime powers. The following lemma shows that the full collection of multi-share parities has rank n for all odd prime powers s.

Lemma 4. *Let $s = \sqrt{n}$ be an odd prime power, and consider an SMNO family $\mathfrak{C} = \{\mathcal{C}^{(0)}(\mathbf{X}), \ldots, \mathcal{C}^{(s)}(\mathbf{X})\}$ of $s+1$ clusters for an n-sharing $\mathbf{X}$. Then the $s(s+1)$ parity sums $\oplus \mathbf{X}_j^{(h)}$ (for $0 \le h \le s$ and $0 \le j \le s-1$) have rank n as linear relations in the variables $X_0, \ldots, X_{n-1}$.*

Proof. Fix a share X_i and let j_0 be such that $X_i \in \mathbf{X}_{j_0}^{(0)}$. Exclude the cluster $\mathcal{C}^{(0)}(\mathbf{X})$ and define

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

Since $\mathfrak{C}$ is SMNO, the share X_i lies in exactly one multi-share of each of the s clusters $\mathcal{C}^{(1)}(\mathbf{X}), \ldots, \mathcal{C}^{(s)}(\mathbf{X})$, hence $|\mathcal{B}_i| = s$. Moreover, any two distinct multi-shares in $\mathcal{B}_i$ intersect exactly in X_i.

Consider the XOR of the parities of all multi-shares in $\mathcal{B}_i$. Every share other than X_i appears either 0 or 2 times in this XOR, while X_i appears exactly s times. Because addition is XOR (characteristic 2) and s is odd, s copies of X_i reduce to a single X_i. Combining this with the row parities excluding $\mathbf{X}_{j_0}^{(0)}$ yields

$$X_i \;=\; \Big(\bigoplus_{\substack{j=0 \\ j \ne j_0}}^{s-1} \oplus \mathbf{X}_j^{(0)} \Big) \;\oplus\; \Big(\bigoplus_{\mathbf{X}_j^{(h)} \in \mathcal{B}_i} \oplus \mathbf{X}_j^{(h)} \Big). \tag{4}$$

Thus each variable X_i can be expressed as a linear combination of the parity sums, implying rank at least n. Since there are only n variables, the rank is exactly n. □

Even characteristic. When s is even (e.g., $s = 2$), the above reconstruction argument fails because s copies of X_i cancel to 0 under XOR, and the rank of the parity system can drop below n. For our gadget constructions in the next section, the odd prime power case suffices.

Example 5. (Detail for $n = 9$, X_5) Referring again to the clusters in Equations (2), consider the share X_5. The set

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

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

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

Since X_5 lies in the row-based multi-share $\mathbf{X}_1^{(0)}$, Equation (4) becomes

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

□

4 Higher-Order Deterministic Multiplication

With our clustering framework in place, we now design gadgets that perform higher-order deterministic multiplication. We start with a simpler construction that achieves glitch resistance but does not yield uniformity.

A $\mathbb{S}\mathsf{Mul}$ gadget takes two n-sharings $\mathbf{X}$ and $\mathbf{Y}$ as inputs and outputs an n-sharing $\mathbf{Z} = [Z_0, \ldots, Z_{n-1}]$ such that the native value Z satisfies $Z = XY$.

Binomial grouping. Given n-sharings $\mathbf{X} = [X_0, \ldots, X_{n-1}]$ and $\mathbf{Y} = [Y_0, \ldots, Y_{n-1}]$, we expand

$$Z \;=\; XY \;=\; \left(\bigoplus_{i=0}^{n-1} X_i\right)\cdot\left(\bigoplus_{j=0}^{n-1} Y_j\right) \;=\; \bigoplus_{i=0}^{n-1}\bigoplus_{j=0}^{n-1} X_iY_j \;=\; \bigoplus_{k=0}^{n-1} Z_k. \qquad (5)$$

Our goal is to partition the n^2 products $\{X_iY_j\}$ into n disjoint *bins*, each assigned to one output share Z_k. A glitch-extended probe on Z_k then reveals all intermediate products contributing to that bin.

Maximizing glitch-extended probing security. Let $\mathbf{L}_k \subseteq \{0, \ldots, n-1\}^2$ be the set of index pairs (i, j) such that X_iY_j is included in Z_k, i.e.,

$$Z_k \;=\; \bigoplus_{(i,j)\in\mathbf{L}_k} X_iY_j, \qquad \bigcup_{k=0}^{n-1} \mathbf{L}_k = \{0, \ldots, n-1\}^2, \quad \mathbf{L}_k \cap \mathbf{L}_{k'} = \emptyset \;(k \neq k').$$

Define the sets of input-share indices involved in bin k by

$$\mathbf{I}_k = \{\, i \mid \exists j \;(i, j) \in \mathbf{L}_k\}, \qquad \mathbf{J}_k = \{\, j \mid \exists i \;(i, j) \in \mathbf{L}_k\}.$$

A glitch-extended probe of Z_k necessarily depends on the shares $\{X_i\}_{i\in\mathbf{I}_k}$ and $\{Y_j\}_{j\in\mathbf{J}_k}$. To maximize the glitch-extended security order, we want these sets to be as small as possible (uniformly over k). A counting argument implies $\max_k |\mathbf{I}_k| \geq s$ or $\max_k |\mathbf{J}_k| \geq s$ when $n = s^2$, and the optimum is attained when $|\mathbf{I}_k| = |\mathbf{J}_k| = s$ for all k.

A cluster-based construction. Assume $n = s^2$ and fix two *distinct* clusters (i.e., two distinct parallel classes)

$$\mathcal{C}^{(h_1)}(\mathbf{X}) = [\mathbf{X}_0^{(h_1)}, \ldots, \mathbf{X}_{s-1}^{(h_1)}], \qquad \mathcal{C}^{(h_2)}(\mathbf{Y}) = [\mathbf{Y}_0^{(h_2)}, \ldots, \mathbf{Y}_{s-1}^{(h_2)}],$$

where each multi-share $\mathbf{X}_\alpha^{(h_1)}$ and $\mathbf{Y}_\beta^{(h_2)}$ contains exactly s shares.

For each $k \in \{0, \ldots, n-1\}$, write

$$\alpha = \lfloor k/s \rfloor, \qquad \beta = \mathsf{mod}(k, s),$$

and define

$$\begin{aligned}
\mathbf{I}_k &:= \{\text{indices of shares in the multi-share } \mathbf{X}_\alpha^{(h_1)}\},\\
\mathbf{J}_k &:= \{\text{indices of shares in the multi-share } \mathbf{Y}_\beta^{(h_2)}\},\\
\mathbf{L}_k &:= \mathbf{I}_k \times \mathbf{J}_k.
\end{aligned}$$

Equivalently, Z_k is the bilinear sum of products of all shares in of one multi-share from $\mathbf{X}$ with those in one multi-share from $\mathbf{Y}$:

$$\begin{aligned}
Z_k &:= \mathbf{X}_\alpha^{(h_1)} \otimes \mathbf{Y}_\beta^{(h_2)}\\
&\overset{\mathsf{def}}{=} \bigoplus_{i=0}^{s-1} \bigoplus_{j=0}^{s-1} \left(\mathbf{X}_\alpha^{(h_1)}[i]\right)\left(\mathbf{Y}_\beta^{(h_2)}[j]\right)\\
&= \left(\bigoplus_{i=0}^{s-1} \mathbf{X}_\alpha^{(h_1)}[i]\right) \cdot \left(\bigoplus_{j=0}^{s-1} \mathbf{Y}_\beta^{(h_2)}[j]\right).
\end{aligned} \tag{6}$$

By construction, each bin involves exactly $|\mathbf{I}_k| = |\mathbf{J}_k| = s$ input shares from $\mathbf{X}$ and $\mathbf{Y}$, respectively, meeting the optimal bound for glitch-extended probing security under this binning strategy. Moreover, the n bins $\mathbf{L}_k$ form a partition of $\{0, \ldots, n-1\}^2$, hence correctness follows from (5). In the notation above, $\mathbf{X}_\alpha^{(h_1)}[i]$ denotes the i-th share inside the multi-share α of the cluster $\mathcal{C}^{(h_1)}(\mathbf{X})$.

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

The output sharing $\mathbf{Z}$ of the multiplication construction from the previous section is, in general, *not* uniformly distributed. We denote this gadget by $\mathbb{S}$Mul-DN, where "D" stands for "deterministic" and "N" for "non-uniform."

Algorithm 1 formally describes $\mathbb{S}$Mul-DN for a fixed choice of two (distinct) cluster indices h_1 and h_2. As an illustration, consider $s = 2$ (hence $n = 4$) and choose $h_1 = 0$ (rows) for $\mathbf{X}$ and $h_2 = 1$ (columns) for $\mathbf{Y}$. Then the gadget outputs:

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

An example for the case $s = 3$ is provided in Appendix B.

Algorithm 1 The $\mathbb{S}$Mul-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: $\quad Z_k \leftarrow \mathbf{X}_\alpha^{(h_1)} \otimes \mathbf{Y}_\beta^{(h_2)}$
5: **return** $\mathbf{Z} = [\, Z_0, Z_1, \ldots, Z_{n-1} \,]$

Correctness. Let $n = s^2$ and recall that for each $k \in \{0, \ldots, n-1\}$ we set $\alpha = \lfloor k/s \rfloor$ and $\beta = \mathsf{mod}(k, s)$, so that (α, β) ranges over all pairs in $\{0, \ldots, s-1\}^2$ exactly once. In $\mathbb{S}$Mul-DN we define $Z_k := \mathbf{X}_\alpha^{(h_1)} \otimes \mathbf{Y}_\beta^{(h_2)}$, i.e.,

$$Z_k = \bigoplus_{i=0}^{s-1} \bigoplus_{j=0}^{s-1} \mathbf{X}_\alpha^{(h_1)}[i]\ \mathbf{Y}_\beta^{(h_2)}[j].$$

Since each cluster partitions the n shares, the family $\{\mathbf{X}_\alpha^{(h_1)} \times \mathbf{Y}_\beta^{(h_2)}\}_{\alpha,\beta}$ forms a partition of all index pairs $(i, j) \in \{0, \ldots, n-1\}^2$. Hence every product term $X_i Y_j$ appears in exactly one output share Z_k, and therefore

$$\bigoplus_{k=0}^{n-1} Z_k \;=\; \bigoplus_{i=0}^{n-1} \bigoplus_{j=0}^{n-1} X_i Y_j \;=\; \left(\bigoplus_{i=0}^{n-1} X_i\right) \cdot \left(\bigoplus_{j=0}^{n-1} Y_j\right) \;=\; XY.$$

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

Glitch-extended probing security. Assume $n = s^2$ and that $\mathbf{X}$ and $\mathbf{Y}$ are independently n-shared. Under the abstraction that a glitch-extended probe covers one output share Z_k (hence the entire bilinear bin defining it), any set of at most $d = s - 1$ probes involves at most $s - 1$ multi-shares from each input sharing and therefore remains *non-complete*; in particular it is statistically independent of (X, Y) and of $Z = XY$.

This bound is tight: with s probes one can recover the full sums defining X and Y. Indeed, consider the s outputs

$$\mathbf{P} = \{\, Z_{\gamma s+\gamma} \mid 0 \le \gamma < s \,\}.$$

By Equation (6), each probe reveals the two multi-share parities

$$S_\gamma^X := \bigoplus_{i=0}^{s-1} \mathbf{X}_\gamma^{(h_1)}[i], \qquad S_\gamma^Y := \bigoplus_{i=0}^{s-1} \mathbf{Y}_\gamma^{(h_2)}[i],$$

since $Z_{\gamma s+\gamma} = S_\gamma^X \cdot S_\gamma^Y$. Summing over γ yields $\bigoplus_\gamma S_\gamma^X = X$ and $\bigoplus_\gamma S_\gamma^Y = Y$, because each cluster partitions all shares. Hence s probes may reveal the secrets.

Non-uniformity. Over $\mathbb{F}_2$, each output share Z_k is biased:

$$\Pr[Z_k = 0] = \tfrac{3}{4} \qquad \text{for all } 0 \le k \le n-1.$$

Indeed, by Equation (6), Z_k is the product of two (independent) parity sums of s input bits. For uniformly random Boolean sharings, each parity sum is a uniform bit, so their product is 1 only when both are 1.

The gadget $\mathbb{S}$Mul-DN is mainly a stepping stone: it showcases the cluster-based binning that achieves higher-order glitch resistance, but it does not preserve uniform output sharings. In the next subsection we refine the construction to restore uniformity while keeping the same security intuition.

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

Nikova et al. [30] gave the first deterministic (binary) multiplication gadget that is first-order probing-secure, glitch-resistant, and *uniform*. We extend this direction by introducing a higher-order deterministic gadget $\mathbb{S}$Mul-DU which, for $n = s^2$ with s a prime power, achieves both *uniformity* and $(d = s-1)$-order glitch resistance over $\mathbb{F}_{2^u}$.

Uniformizing by adding a structured 0*-sharing.* Starting from the non-uniform gadget $\mathbb{S}$Mul-DN, we restore uniformity by adding an auxiliary sharing $\mathbf{R} = [R_0, \ldots, R_{n-1}]$ of the constant 0 to the outputs. The key constraints are: (i) each R_k must be computable using only the same multi-shares already involved in the computation of Z_k (to preserve non-completeness under glitch-extended probes), and (ii) the family (R_k) must be sufficiently *jointly* random so that $\mathbf{Z} \oplus \mathbf{R}$ forms a uniform n-sharing. Condition (i) preserves probing security; condition (ii) is ensured by the linear-independence property of cluster parities from Section 3.4.

Design of $\mathbb{S}$Mul-DU Let $n = s^2$. Fix an SMNO family of $s+1$ clusters $\{\mathcal{C}^{(0)}(\cdot), \ldots, \mathcal{C}^{(s)}(\cdot)\}$, where $\mathcal{C}^{(h)}(\mathbf{X}) = [\mathbf{X}_0^{(h)}, \ldots, \mathbf{X}_{s-1}^{(h)}]$ (and similarly for $\mathbf{Y}$). For each $k \in \{0, \ldots, n-1\}$, write

$$k = \alpha s + \beta \quad \Rightarrow \quad \alpha = \lfloor k/s \rfloor, \beta = \mathsf{mod}(k, s).$$

We define Z_k as the XOR of a multiplication term W_k and an auxiliary term R_k:

$$\begin{aligned} R_k &:= \bigoplus_{i=0}^{s-1} \Big(\mathbf{X}_\alpha^{(0)}[i] \oplus \mathbf{X}_\beta^{(\alpha+1)}[i] \oplus \mathbf{Y}_\alpha^{(0)}[i] \oplus \mathbf{Y}_\beta^{(\alpha+1)}[i] \Big), \\ W_k &:= \mathbf{X}_\alpha^{(0)} \otimes \mathbf{Y}_\beta^{(\alpha+1)}, \\ Z_k &:= R_k \oplus W_k. \end{aligned} \tag{7}$$

Here, $h = 0$ denotes the row cluster, while $h = \alpha + 1 \in \{1, \ldots, s\}$ selects one of the remaining s SMNO clusters. (depending on the row index α), which is possible when s is a prime power (see Proposition 1).

Example ($n = 4$, $s = 2$). For $s = 2$ we have three clusters $h \in \{0, 1, 2\}$. Taking the standard SMNO family for $n = 4$ (rows as $h = 0$ and columns/diagonals as $h \in \{1, 2\}$), Equation (7) yields the four output shares:

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

An explicit instance for $n = 9$ is given in Appendix B.

Algorithm and correctness. Algorithm 2 summarizes the construction of $\mathbb{S}$Mul-DU. A proof of correctness (i.e., $\bigoplus_{k=0}^{n-1} Z_k = XY$) is given in Appendix A.

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

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

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

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

Glitch-extended probing security. Assume $n = s^2$ and that the input sharings $\mathbf{X}$ and $\mathbf{Y}$ are independent. We claim that $\mathbb{S}$Mul-DU is secure up to glitch-extended probing order $d = s - 1$.

The argument follows directly from non-completeness at the multi-share level. Fix any probed output share Z_k with $k = \alpha s + \beta$. By Equation (7) (or Algorithm 2), the computation of Z_k involves only the two multi-shares

$$\mathbf{X}_\alpha^{(0)},\ \mathbf{X}_\beta^{(\alpha+1)} \qquad \text{and} \qquad \mathbf{Y}_\alpha^{(0)},\ \mathbf{Y}_\beta^{(\alpha+1)},$$

i.e., one multi-share from the row cluster $\mathcal{C}^{(0)}(\cdot)$ and one multi-share from a non-row cluster $\mathcal{C}^{(\alpha+1)}(\cdot)$ (with $\alpha + 1 \in \{1, \ldots, s\}$).

Consider any set of at most $d = s-1$ glitch-extended probes. Then, for each secret sharing (say $\mathbf{X}$), the probes can cover at most $s-1$ multi-shares from the row cluster and at most $s-1$ multi-shares from the other involved clusters. By Lemma 3, such a collection of probed multi-shares is *non-complete*: it cannot cover all n shares of $\mathbf{X}$ (and similarly of $\mathbf{Y}$). Consequently, the resulting probed values are statistically independent of the native secrets X and Y (and hence also of $Z = XY$).

Uniformity of $\mathbb{S}$Mul-DU. For $s \in \{2, 3\}$ (i.e., $n \in \{4, 9\}$), uniformity can be verified by exhaustive enumeration. For larger s, this is infeasible, and we instead use a counting argument. It suffices to show that for any fixed $z \in \mathbb{F}_{2^u}$, every $(n-1)$-subtuple of the output shares is uniform conditioned on $Z = z$; equivalently, for any fixed realization $\mathbf{z} \in (\mathbb{F}_{2^u})^{n-1}$, the number of input pairs $(\mathbf{x}, \mathbf{y})$ mapping to $\mathbf{Z}_1 = \mathbf{z}_1$ depends only on z (and not on $\mathbf{z}_1$).

Linearization via row parities of $\mathbf{X}$. Write $n = s^2$ and index $k = \alpha s + \beta$ with $0 \le \alpha, \beta < s$. Define the *row parities* of $\mathbf{X}$ by

$$\theta_\alpha \;:=\; \bigoplus_{i=0}^{s-1} \mathbf{X}_\alpha^{(0)}[i] \qquad (0 \le \alpha < s).$$

These θ_α are the only source of nonlinearity in Equation (7) through the products in W_k. Condition on fixed values $(\theta_0, \ldots, \theta_{s-1})$. Then each output share Z_k becomes an *affine-linear* expression in the remaining parity variables of $\mathbf{X}$ and $\mathbf{Y}$. Concretely, substituting θ_α into Equation (7) gives, for $k = \alpha s + \beta$,

$$Z_k = \theta_\alpha \cdot \Big(\bigoplus \mathbf{Y}_\beta^{(\alpha+1)}\Big) \;\oplus\; \theta_\alpha \;\oplus\; \Big(\bigoplus \mathbf{X}_\beta^{(\alpha+1)}\Big) \;\oplus\; \Big(\bigoplus \mathbf{Y}_\alpha^{(0)}\Big) \;\oplus\; \Big(\bigoplus \mathbf{Y}_\beta^{(\alpha+1)}\Big), \tag{8}$$

where we use the shorthand $\bigoplus \mathbf{X}_j^{(h)} := \bigoplus_{i=0}^{s-1} \mathbf{X}_j^{(h)}[i]$ (and similarly for $\mathbf{Y}$). Thus, conditioned on θ_α values, the system consisting of Z_k-equations is linear.

Counting preimages for $(n-1)$ *observed shares.* Fix any $(n-1)$-subtuple $\mathbf{Z}_1$ of the output shares and a realization $\mathbf{z}_1$. Let the missing share lie in row $\alpha^\star$ (i.e., its index is $\alpha^\star s + \beta^\star$ for some $\beta^\star$). From the observed shares, we can form the $s-1$ row-sums

$$S_\alpha \;:=\; \bigoplus_{\beta=0}^{s-1} Z_{\alpha s+\beta} \qquad (\alpha \in \{0, \ldots, s-1\} \setminus \{\alpha^\star\}),$$

which are known values determined by $\mathbf{z}_1$.

Summing Z_k as in Equation (8) over β and using that each cluster partitions all shares yields, for each such α,

$$S_\alpha = \theta_\alpha\, Y \;\oplus\; \theta_\alpha \;\oplus\; X \;\oplus\; \Big(\bigoplus \mathbf{Y}_\alpha^{(0)}\Big) \;\oplus\; Y, \tag{9}$$

(where we use $X = \bigoplus_\alpha \theta_\alpha$ and $\bigoplus_\beta \bigoplus \mathbf{Y}_\beta^{(\alpha+1)} = Y$). Crucially, Equation (9) contains no $\mathbf{X}$-shares beyond $X = \bigoplus_\alpha \theta_\alpha$ (which is fixed under the conditioning).

On the $\mathbf{X}$ side, consider the parity family

$$\mathcal{B}_X := \Big\{ \bigoplus \mathbf{X}_i^{(0)} = \theta_i \;\Big|\; 0 \le i < s \Big\} \cup \Big\{ \bigoplus \mathbf{X}_\beta^{(\alpha+1)} \;\Big|\; 0 \le \alpha < s,\ 0 \le \beta < s-1 \Big\}.$$

This set has size $s + s(s-1) = s^2 = n$, and by Lemma 4 (rank-n independence of cluster parities for odd prime powers), these n relations are linearly independent and determine the $\mathbf{X}$-shares with full rank.

Finally, the $s-1$ row-sum equations from Equation (9) provide $s-1$ additional independent linear constraints on $\mathbf{Y}$ (they may lose rank by at most one due to the common Y term, hence $s-1$ is the guaranteed rank contribution here). Altogether, for any fixed $(\theta_0, \ldots, \theta_{s-1})$ and any fixed $\mathbf{z}_1$, we obtain at least

$$n + (s-1) = n + s - 1$$

linearly independent equations in the $2n$ share variables of $(\mathbf{X}, \mathbf{Y})$. Therefore, the solution space has dimension

$$2n - (n + s - 1) = n - s + 1,$$

so there are exactly $2^{u(n-s+1)}$ input pairs $(\mathbf{x}, \mathbf{y})$ consistent with $\mathbf{Z}_1 = \mathbf{z}_1$ for each fixed choice of $(\theta_0, \ldots, \theta_{s-1})$.

Since $(\theta_0, \ldots, \theta_{s-1}) \in (\mathbb{F}_{2^u})^s$ ranges over 2us possibilities, the total number of preimages of $\mathbf{z}_1$ equals

$$2^{us} \cdot 2^{u(n-s+1)} \;=\; 2^{u(n+1)},$$

which is independent of $\mathbf{z}_1$. This proves that $\mathbf{Z}_1$ is uniform, and hence $\mathbf{Z}$ is a uniform n-sharing.

4.5 Composability of $\mathbb{S}$Mul-DU

Having a gadget that is both d-probe secure and *uniform* does not, by itself, imply that an arbitrary composition of such gadgets yields a secure masked cipher. The missing ingredient is a *composition theorem*: when gadgets are connected, probes on internal wires may correlate across gadget boundaries, and (in deterministic masking) there is no fresh online randomness that one can use to explicitly refresh intermediate sharings.

Classical compositional frameworks therefore rely on dedicated *refresh* gadgets inserted between nonlinear layers. This approach is incompatible with our goal of eliminating online randomness.

Why $\mathbb{S}$Mul-DU *is "refresh-like".* Recall that $\mathbb{S}$Mul-DU outputs $\mathbf{Z} = \mathbf{W} \oplus \mathbf{R}$, where $\mathbf{W}$ is an n-sharing of XY and $\mathbf{R}$ is a deterministically generated n-sharing of 0. Although $\mathbf{R}$ uses no fresh randomness, its addition is similar to a lightweight re-randomization at the gadget boundary.

To reason about internal leakage, we follow the *probe propagation* viewpoint of Cassiers–Standaert [10]: an adversarial probe inside a gadget can be replaced by a set of *propagated probes* on its input shares that suffice to simulate it. Since each output component Z_k depends only on a local set of input shares (one multi-share from the row cluster and one multi-share from a second cluster, for each input), any internal (glitch-extended) probe can be conservatively charged to a small number of such input multi-shares. These propagated probes then act as effective probes on the previous-stage outputs.

At a high level, the additive structure induced by $\mathbf{R}$ suggests that such effective output probes can often be handled with the probe-elimination principle of Section 2.2, as long as the global probe set remains non-complete. We leave a formal propagation/simulation argument (and a general composability theorem) for future work.

Scope of our claims. We do *not* prove a general composability statement for $\mathbb{S}$Mul-DU in the sense of PINI/O-PINI [10,11]. Formalizing such a result (e.g., in a simulation-based framework) is an interesting direction for future work. In the remainder of the paper, we instead establish compositional security in *specific* settings: notably, for round-based ciphers with a bricklayer structure, where a diffusion layer can provably prevent completeness from building up across rounds (Section 2.2), and where we can give a dedicated standard probing analysis for Ascon.

5 Application to Ascon

In this section, we give a concrete application of the clustering framework to deterministic masking at the scale of a full cipher. Our target is Ascon [21], standardized by NIST as a lightweight primitive [35]. Ascon follows a bricklayer structure: a parallel S-box layer followed by a linear diffusion layer (see Figure 1). The nonlinear layer has algebraic degree 2 and is built from AND operations (one quadratic term per output bit). As discussed in Section 2.2, the diffusion layer of Ascon supports deterministic multi-round composition up to 3 *standard* probes.

Therefore, our goal is to realize a $d = 3$ masking of the Ascon S-box with the minimum number of shares, namely $n = 4$. In this section, we focus on the *standard probing* model and on software-oriented constructions.

Ascon is summarized in Appendix C. The main obstacle for deterministic masking is the masked S-box. As depicted in Figure 7, the S-box is a 5-bit permutation obtained by two lightweight linear layers around the core nonlinearity χ_5.

The map χ_m. Given m bits $(X^0, \ldots, X^{m-1})$, the map χ_m outputs $(Y^0, \ldots, Y^{m-1})$ defined by

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

where indices are taken modulo m. A corresponding gadget $\mathbb{S}\chi_m$ takes m input n-sharings $(\mathbf{X}^0, \ldots, \mathbf{X}^{m-1})$ and returns m output n-sharings $(\mathbf{Y}^0, \ldots, \mathbf{Y}^{m-1})$ that realize χ_m on the native values.

To build $\mathbb{S}\chi_5$, it is convenient to combine XOR and one multiplication into a single primitive. This gives additional freedom in how we assign clusters, and it simplifies the probe-elimination argument in the standard probing model.

5.1 $\mathbb{S}$Mul-XOR Gadget

A Mul-XOR gate maps $(X, Y, Z) \in (\mathbb{F}_{2^u})^3$ to $W = Z \oplus X \cdot Y$. Accordingly, a gadget $\mathbb{S}$Mul-XOR takes three n-sharings $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ and outputs an n-sharing $\mathbf{W}$ such that $W = Z \oplus X \cdot Y$.

Construction. Assume $n = s^2$ and fix three pairwise MNO clusters (e.g., rows, columns, and wrap-around diagonals), with indices (h_X, h_Y, h_Z). We compute the output in the coordinates of cluster h_Z, i.e., we fill the shares $\mathbf{W}_\alpha^{(h_Z)}[\beta]$ for $0 \le \alpha, \beta < s$. Using the multi-share tensor notation $\mathbf{U} \otimes \mathbf{V} = \bigoplus_{i,j} U[i] \cdot V[j]$ introduced in Section 4, we set

$$\mathbf{W}_\alpha^{(h_Z)}[\beta] \;=\; \mathbf{Z}_\alpha^{(h_Z)}[\beta] \;\oplus\; \big(\mathbf{X}_\alpha^{(h_X)} \otimes \mathbf{Y}_\beta^{(h_Y)}\big).$$

Algorithm 3 gives an explicit implementation.

Algorithm 3 The $\mathbb{S}$Mul-XOR gadget (clustered form)

Input: $(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \in \big((\mathbb{F}_{2^u})^n\big)^3$, cluster indices (h_X, h_Y, h_Z)
Output: $\mathbf{W} \in (\mathbb{F}_{2^u})^n$ with $W = Z \oplus X \cdot Y$

for $\alpha = 0$ to $s-1$ **do**
  for $\beta = 0$ to $s-1$ **do**
    $W \leftarrow \mathbf{Z}_\alpha^{(h_Z)}[\beta]$
    for $i = 0$ to $s-1$ **do**
      for $j = 0$ to $s-1$ **do**
        $W \leftarrow W \oplus \big(\mathbf{X}_\alpha^{(h_X)}[i] \cdot \mathbf{Y}_\beta^{(h_Y)}[j]\big)$
    $\mathbf{W}_\alpha^{(h_Z)}[\beta] \leftarrow W$
return W

Standard probing security (probe elimination viewpoint). Each share $\mathbf{W}_\alpha^{(h_Z)}[\beta]$ is computed as an accumulator initialized with the input share $\mathbf{Z}_\alpha^{(h_Z)}[\beta]$ and then updated by terms depending only on the multi-shares $\mathbf{X}_\alpha^{(h_X)}$ and $\mathbf{Y}_\beta^{(h_Y)}$. Thus, any *single* probe on this accumulator is one-time padded by $\mathbf{Z}_\alpha^{(h_Z)}[\beta]$ and can be eliminated as long as this Z-share is not otherwise exposed (Section 2.2).

The only way to "remove" this pad is to either (i) also probe $\mathbf{Z}_\alpha^{(h_Z)}[\beta]$, or (ii) probe the accumulator twice at two different times and cancel the common

$\mathbf{Z}_\alpha^{(h_Z)}[\beta]$. In a conservative accounting, each such non-eliminable event can be charged to revealing one multi-share of $\mathbf{X}$ (namely $\mathbf{X}_\alpha^{(h_X)}$) and one multi-share of $\mathbf{Y}$ (namely $\mathbf{Y}_\beta^{(h_Y)}$), plus one share of $\mathbf{Z}$. Hence, before completeness can occur in either $\mathbf{X}$ or $\mathbf{Y}$, we must have collected at least s distinct charged events. This yields the bound $d \leq 2s - 1$ under standard probes. In particular, for $n = 4$ ($s = 2$) we obtain $d \leq 3$, which matches the information-theoretic upper bound $d \leq n - 1$.

5.2 Gadget $\mathbb{S}\chi_m$

When m is odd, the map χ_m is a permutation on $\mathbb{F}_2^m$. In a deterministic masking setting, it is therefore desirable that a gadget $\mathbb{S}\chi_m$ not only be probing-secure and uniform, but also behave *bijectively* on the mn input shares (so that it does not "collapse" the set of valid sharings). However, a direct instantiation based on the multiplication gadget of Nikova et al. [30] (for $n = 4$) does *not* yield a bijective realization of $\mathbb{S}\chi_m$, and earlier attempts at uniform deterministic gadgets for χ_m faced similar issues [7].

Daemen [16] proposed the *Changing of the Guards* technique, which borrows entropy from neighboring $\mathbb{S}\chi_m$ instances to restore uniformity, and Shahmirzadi–Moradi [34] found (by exhaustive search) uniform and probing-secure realizations of $\mathbb{S}\chi_5$ (up to second order) that are bijective. For a comparison of our χ_m-masking with existing approaches, see Table 3.

Candidate	(A)	(B)	(C)	(D)	(E)	(F)	(G)
Unprotected [15]	1	0	0	odd m	✓	$2m$	m
Bilgin et al. [7]	4	0	1	5	✓	80	30
Daemen et al. [17]	2	1 bit	1	5	×	22	20
Shahmirzadi and Moradi [34]	3	0	2	5	✓	80	45
$\mathbb{S}\chi_m$ [This work]	4	0	3	5 & 3	✓	85 ($m = 5$)	80 ($m = 5$)

Table 3: Comparison of dedicated masked algorithms for the maps χ_m. The columns indicate: (A) sharing order n; (B) required online randomness; (C) probing security order; (D) supported values of m; (E) uniform output (i.e., mn-bit permutation); (F) number of XORs; (G) number of ANDs.

Here we give a simple *template* for $\mathbb{S}\chi_m$ built from our $\mathbb{S}$Mul-XOR gadget, and we instantiate it for $m \in \{3, 5\}$ with $n = 4$ to obtain *third-order* standard probing security while preserving bijectivity (hence uniformity). An explicit instantiation for $\mathbb{S}\chi_5$ is given in Appendix B. All indices on the superscripts $(\cdot)^i$ are taken modulo m.

This construction requires three *pairwise MNO* clusters, indexed by (h_0, h_1, h_2). Each input n-sharing $\mathbf{X}^i$ is used in exactly three different $\mathbb{S}$Mul-XOR calls—once in each role (left multiplicand, right multiplicand, and additive input)—and therefore is accessed through three distinct clusters. The map $\mathsf{shake}(\cdot, h_2)$ is

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

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

function $\mathsf{neg}(\mathbf{U})$
  $\mathbf{V} \leftarrow \mathbf{U}$
  $V_0 \leftarrow 1 \oplus U_0$ ▷ flip the native bit
  return $\mathbf{V}$

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

for $i = 0$ **to** $m-2$ **do**
  $\mathbf{Y}^i \leftarrow \mathbb{S}\mathsf{Mul\text{-}XOR}\big(\mathbf{X}^{i+2}, \mathsf{neg}(\mathbf{X}^{i+1}), \mathbf{X}^i; h_0, h_1, h_2\big)$
$\mathbf{Y}^{m-1} \leftarrow \mathbb{S}\mathsf{Mul\text{-}XOR}\big(\mathbf{X}^1, \mathsf{neg}(\mathbf{X}^0), \mathsf{shake}(\mathbf{X}^{m-1}, h_2); h_0, h_1, h_2\big)$
return $(\mathbf{Y}^0, \ldots, \mathbf{Y}^{m-1})$

a fixed share permutation: it preserves correctness and probing security, but it breaks a symmetry that otherwise prevents $\mathbb{S}\chi_m$ from being bijective. We verified by exhaustive enumeration that for $(n, u) = (4, 1)$ and $m \in \{3, 5\}$, the resulting $\mathbb{S}\chi_m$ is bijective (hence uniform). Bijectivity for other parameter choices (n, m, u) remains open.

Probing Security. Assume that the input n-sharings $\mathbf{X}^0, \ldots, \mathbf{X}^{m-1}$ are mutually independent (Definition 1) and that h_0, h_1, h_2 are three *distinct* clusters. In Algorithm 4, each sharing $\mathbf{X}^i$ is involved in exactly three consecutive $\mathbb{S}\mathsf{Mul\text{-}XOR}$ calls: (i) as the additive input (hence under cluster h_2), (ii) as a multiplicand under cluster h_1 (via neg), and (iii) as a multiplicand under cluster h_0. Accordingly, any intermediate value probed inside these calls has the generic form *(additive share)* $\oplus$ *(partial sum of products of two multi-shares)*. Concretely, ignoring the (linear) effects of neg and shake for readability, we can write:

$$
\begin{aligned}
D_1 &= (\mathbf{X}^i)_\alpha^{(h_2)}[\beta] \;\oplus\; \Big(\bigoplus_{p \in I_1} \bigoplus_{q \in J_1} (\mathbf{X}^{i+2})_\alpha^{(h_0)}[p] \cdot (\mathbf{X}^{i+1})_\beta^{(h_1)}[q] \Big), \\
D_2 &= (\mathbf{X}^{i-1})_\alpha^{(h_2)}[\beta] \;\oplus\; \Big(\bigoplus_{p \in I_2} \bigoplus_{q \in J_2} (\mathbf{X}^{i+1})_\alpha^{(h_0)}[p] \cdot (\mathbf{X}^i)_\beta^{(h_1)}[q] \Big), \\
D_3 &= (\mathbf{X}^{i-2})_\alpha^{(h_2)}[\beta] \;\oplus\; \Big(\bigoplus_{p \in I_3} \bigoplus_{q \in J_3} (\mathbf{X}^i)_\alpha^{(h_0)}[p] \cdot (\mathbf{X}^{i-1})_\beta^{(h_1)}[q] \Big),
\end{aligned}
\tag{10}
$$

where $\alpha, \beta \in \{0, \ldots, s-1\}$ identify the output position ($k = \alpha s + \beta$), $I_t, J_t \subseteq \{0, \ldots, s-1\}$ depend on the exact intermediate in the accumulation, and indices

$i\pm1, i\pm2$ are modulo m. (The blue term highlights the occurrence of the targeted input $\mathbf{X}^i$.)

Single-probe case. If the adversary places a *single* probe on any intermediate of the form (10), then this probe contains a unique additive share from the third input of the corresponding $\mathbb{S}$Mul-XOR call (e.g., $(\mathbf{X}^{i-2})_{\alpha}^{(h_2)}[\beta]$ in D_3). Provided this additive share does not appear in any other probe, the probe can be eliminated by the rule of Section 2.2.

Two or more probes. Elimination can fail only if the adversary probes *twice* in a way that reveals the same additive share (e.g., two probes targeting intermediates in the computation of the same output component). In this case, we conservatively charge such a pair of probes as leaking: (i) that additive share, and (ii) the two involved multiplicand multi-shares (one from each of clusters h_0 and h_1), since the product-sum part depends only on these two multi-shares. Applied to $\mathbf{X}^i$, this means that once elimination fails, a probe can effectively reveal (at most) one new multi-share of $\mathbf{X}^i$ under cluster h_0 or h_1.

In the worst case, the first non-eliminable event reveals one multi-share of $\mathbf{X}^i$ under one of $\{h_0, h_1\}$, and subsequent probes can reveal at most one additional multi-share under the other cluster. Hence, with at most $2 + (s-1) = s+1 = \sqrt{n}+1$ probes, the adversary learns at most $(s-1)$ multi-shares from one cluster and one multi-share from a second cluster, i.e., at most $2s-2$ multi-shares. By Lemma 3, this set is non-complete, so the probes remain independent of the native secrets. Therefore, Algorithm 4 achieves standard probing security of order $d = s+1$ (in particular, $d = 3$ for our target instantiation $n = 4$).

5.3 A Deterministic $\mathbb{S}$Ascon

To obtain a deterministic masking of Ascon, we replace the nonlinear χ_5 by our $\mathbb{S}\chi_5$ gadget (Section 5.2), while masking the surrounding affine/linear operations in a share-wise manner. In particular, we do *not* insert refresh gadgets and we use no online randomness: all computations reuse the initial input sharings. Our software implementation[3] builds on the reference code base maintained by the Ascon team [20].

A pitfall in the $d = 3$ masked S*-box of [20].* The repository simpleserial-ascon [20] provides masked S-box implementations targeting $d \in \{1, 2, 3\}$. While the $d \in \{1, 2\}$ cases are linked to earlier work, we could not find a formal analysis for the $d = 3$ instance and, in the *standard probing* model, we observed a problematic dependency pattern.[4]

[3] Available through https://github.com/vahid-jahandideh/deterministic-masking-ascon

[4] This observation concerns only the $d = 3$ instance in that code base.

Concretely, one update computes an output share Z'_j from input sharings $\mathbf{Z} = [Z_0, \ldots, Z_3]$, $\mathbf{X}$, and $\mathbf{Y}$ as

$$Z'_j \;=\; Z_j \oplus X_jY_0 \oplus X_jY_1 \oplus X_jY_2 \oplus X_jY_3 \;=\; Z_j \oplus X_j\Big(\bigoplus_{i=0}^{3} Y_i\Big).$$

An adversary probing $P_1 = Z_j$ and $P_2 = Z'_j$ learns

$$P_1 \oplus P_2 \;=\; X_j\Big(\bigoplus_{i=0}^{3} Y_i\Big) \;=\; X_j\, Y,$$

where $Y = \bigoplus_i Y_i$ is the native value. The distribution of X_jY depends on Y, hence these two probes already leak information about the secret and violate third-order probing security. Our implementation avoids this issue by using a third-order secure $\mathbb{S}\chi_5$ construction.

Multi-round composition in Ascon. For Ascon, the diffusion layer has box linear branch number $\mathcal{B}_{l\text{-box}} = 4$ (Definition 2). By Lemma 1, the global probing security order d of the composed cipher satisfies

$$\min\{d_{\mathbb{S}\text{-box}}, \mathcal{B}_{l\text{-box}} - 1\} \;\le\; d \;\le\; d_{\mathbb{S}\text{-box}},$$

where $d_{\mathbb{S}\text{-box}}$ is the local probing security of the masked S-box. The cases $d_{\mathbb{S}\text{-box}} \in \{1, 2\}$ were analyzed in [28] (building on [34]), and in this work we provide a construction achieving $d_{\mathbb{S}\text{-box}} = 3$, hence enabling deterministic composition up to $d = 3$ standard probes.

Local probing security and uniformity of the masked S-box. Let the n-sharings entering $\mathbb{S}\chi_5$ be $[\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}]$. In Ascon, these are obtained from the state sharings by the (share-wise) affine pre-layer:

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

which corresponds to the inverse of the preceding linear layer (Figure 7). Since this step is affine and applied share-wise, it preserves uniformity and does not create cross-share dependencies beyond those already present in the inputs. The nonlinear part is then handled by $\mathbb{S}\chi_5$, whose intermediate values match the $\mathbb{S}$Mul-XOR pattern (Equation (10), up to affine terms), so the standard probing argument of Section 5.2 applies. Finally, the post-linear layer is again share-wise and thus preserves the guarantees.

We additionally translated our masked S-box into the input language of the verification tool SILVER [29] and mechanically checked third-order *standard* probing security and uniformity.

Randomness Savings. Each round of the Ascon permutation applies 64 parallel χ_5 instances to the 320-bit state, and each χ_5 uses 5 quadratic terms. Hence, a masked round involves $64 \cdot 5 = 320$ binary multiplications.

Without deterministic masking, standard d-probe-secure binary multiplication gadgets (e.g., ISW/PINI-style [10, 27]) require around $\frac{d(d+1)}{2}$ online randomness bit per multiplication. Restricting randomness to the nonlinear part, this yields an online randomness budget of

$$320 \cdot \tfrac{d(d+1)}{2} \;=\; 160\, d(d+1) \quad \text{random bits per round,}$$

which accumulates linearly with the number of rounds. Our deterministic masking removes this cost: the masked permutation uses no online randomness.

5.4 Implementation Results

Implementation and platform. We implemented our deterministic $d = 3$ masking of Ascon in software. Beyond masking, the implementation incorporates standard hardening techniques including share rotation [24] and word-level interleaving. To prevent compiler-induced reordering in the critical nonlinear computations, the core masked S-box is implemented in ARM assembly.

All measurements were performed on a ChipWhisperer CW308 board with an STM32F303 (ARM Cortex-M4) target,[5] using a ChipWhisperer-Lite for power acquisition. We use synchronous sampling, with the sampling frequency set to either $f_s = f_{\text{clk}}$ or $f_s = 4f_{\text{clk}}$ (depending on the experiment), which yields highly aligned traces and, consequently, an unusually high signal-to-noise ratio (SNR) in this educational setup.

Performance. Table 4 reports cycle counts for encryption under different protection orders. We report both the cost of encrypting a 1-byte message (dominated by initialization/overhead) and the steady-state cost per byte for a long message.

Table 4: Cycle count per byte under different protection orders (software on STM32F303).

Protection Order	Source	1 Byte	Long Message
$d = 0$	Ascon Team [20]	4,274	99
	Jahandideh et al. [28]	5,109	124
$d = 1$	Ascon Team [20]	11,085	310
	Jahandideh et al. [28]	13,339	388
$d = 2$	Ascon Team [20]	20,883	585
	Jahandideh et al. [28]	25,414	745
$d = 3$	This work	73,522	2,229

[5] `https://github.com/newaetech/chipwhisperer`

Overall, the proposed $d = 3$ protection is about 3 times slower than $d = 2$ implementation [28] in cycle count. We stress that these numbers exclude any cost for random-number generation, since our construction uses no online randomness.

Leakage assessment methodology. We evaluate leakage using two complementary tools: (i) an SNR measurement to characterize the measurement setup and locate informative time regions, and (ii) fixed-vs-random Welch t-tests (TVLA-style) to detect higher-order leakage.

Signal-to-noise ratio (SNR). Given traces $T(t)$ (sample index t) and a discrete *class* variable C (here: different fixed input instances), we compute the per-sample SNR as

$$\mathrm{SNR}(t) \;=\; \frac{\mathrm{Var}\big(\mathbb{E}[T(t) \mid C]\big)}{\mathbb{E}\big[\mathrm{Var}(T(t) \mid C)\big]}.$$

Intuitively, the numerator captures how much the *mean* trace changes across classes, while the denominator captures within-class noise. Figure 4 shows the SNR for the first $\approx$ 9K samples, covering initialization and multiple rounds. The relatively high SNR confirms that our ChipWhisperer setup is strongly favorable to attacks and therefore provides a conservative testbed.

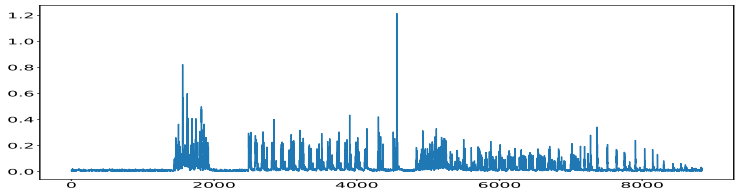

Fig. 4: SNR for the first $\approx$ 9K samples (initialization + multiple rounds) on CW308/STM32F303.

Higher-order t-tests. For order r, we apply Welch's t-test to centered products of r samples (chosen from SNR peaks around the nonlinear computations). We use the conventional detection threshold ± 4.5. Due to the combinatorial explosion in the number of r-tuples, we report the maximum $|t|$ over a representative set of tuples drawn from those high-SNR regions.

Leakage results. In [28], the $d = 1$ software masking is detectably vulnerable to second-order t-tests with about 30,000 traces. In our experiments, the $d = 2$ implementation becomes detectably vulnerable to third-order t-tests at around 4×10^5 traces (Figure 5, exceeding ± 4.5), which motivates $d = 3$ protection in this measurement setting.

For our $d = 3$ implementation, the third-order t-test remains below threshold on the tested tuples for up to 2 million traces. This is an empirical sanity check

(not a proof of absence of leakage): higher-order leakage may still exist and could become detectable with more traces or a different tuple selection strategy. In addition to these measurements, we translated our masked S-box into the verification tool SILVER [29] and confirmed its third-order *standard probing* security and uniformity.

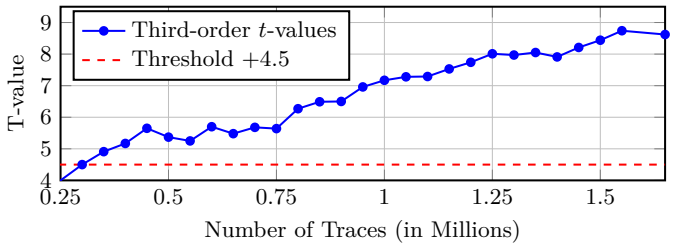

Fig. 5: Third-order Welch t-test on selected sample triples around nonlinear computations.

6 Conclusion

We introduced a systematic *cluster* approach for organizing shares and showed how it enables gadget design under the strict constraints of *deterministic masking*, where randomness is confined to the initial input sharing.

Using clustering, we constructed a higher-order secure multiplication gadget, addressing a longstanding open problem. We further demonstrated the approach on a full cipher by deterministically protecting Ascon, illustrating that higher-order deterministic masking can be realized beyond standalone gadgets.

Future work. Our clustering framework currently targets square numbers of shares ($n = s^2$). Extending deterministic masking to non-square n (or finding alternative structures) is a natural direction. Another open avenue is to better understand whether the additional algebraic/geometric structure can be exploited for attacks; we did not observe such attacks in our investigations, but a deeper analysis remains worthwhile.

Acknowledgements.

We thank Joan Daemen for useful suggestions for this research. Jan Schoone is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. Lejla Batina is supported by Dutch Research Council (NWO) through the PROACT project (NWA.1215.18.014), TTW PREDATOR project 19782 and the CiCS project of the research programme Gravitation under the grant 024.006.037.

References

1. 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)
2. 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)
3. Batten, L.M.: Combinatorics of Finite Geometries. Cambridge University Press (1997)
4. 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)
5. 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)
6. 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)
7. 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)
8. 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
9. 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
10. 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
11. Cassiers, G., Standaert, F.X.: Provably Secure Hardware Masking in the Transition- and Glitch-Robust Probing Model: Better Safe than Sorry. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(2), 136–158 (Feb 2021). https://doi.org/10.46586/tches.v2021.i2.136-158, https://tches.iacr.org/index.php/TCHES/article/view/8790
12. 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)

13. 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)
14. 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)
15. Daemen, J.: Cipher and Hash Function Design Strategies based on linear and differential cryptanalysis. Ph.D. thesis, Katholieke Universiteit Leuven (1995)
16. 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)
17. Daemen, J., Dobraunig, C., Eichlseder, M., Groß, H., Mendel, F., Primas, R.: Protecting against Statistical Ineffective Fault Attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 508–543 (2020). https://doi.org/10.13154/TCHES.V2020.I3.508-543
18. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020). https://doi.org/10.1007/978-3-662-60769-5
19. 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)
20. Dietrich, F., Dobraunig, C., Mendel, F., Primas, R., Schläffer, M.: simpleserial-ascon: Protected Ascon Implementations. https://github.com/ascon/simpleserial-ascon (2022), accessed: January 2026
21. 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
22. 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
23. 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)
24. Gaspoz, J., Dhooghe, S.: Threshold Implementations in Software: Microarchitectural Leakages in Algorithms. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(2), 155–179 (Mar 2023). https://doi.org/10.46586/tches.v2023.i2.155-179
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. 1469–1486. 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.: Probing Secure Composability Without Fresh Randomness: Theory and Application to Ascon. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2025**(4), 972–1009 (2025). https://doi.org/10.46586/TCHES.V2025.I4.972-1009
29. Knichel, D., Sasdrich, P., Moradi, A.: SILVER - Statistical Independence and Leakage Verification. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 787–816. Springer (2020). https://doi.org/10.1007/978-3-030-64837-4_26
30. 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)
31. Playfair, J.: The element of geometry. W.E. Dean (1836)
32. 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
33. 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)
34. 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)
35. Turan, M.S., McKay, K.A., Chang, D., Kang, J., Kelsey, J.: Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions. NIST Special Publication 800-232, National Institute of Standards and Technology, Gaithersburg, MD (Aug 2025). https://doi.org/10.6028/NIST.SP.800-232, https://csrc.nist.gov/pubs/sp/800/232/final
36. Wang, W., Ji, F., Zhang, J., Yu, Y.: Efficient Private Circuits with Precomputation. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(2), 286–309 (Mar 2023). https://doi.org/10.46586/tches.v2023.i2.286-309

A Proof of Correctness of $\mathbb{S}$Mul-DU

Recall Algorithm 2. Correctness follows from two facts:

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

Since $Z_k = W_k \oplus R_k$, this implies $\bigoplus_{k=0}^{n-1} Z_k = XY$.

The auxiliary sharing $\mathbf{R}$ *encodes* 0. Write $n = s^2$ and index $k = \alpha s + \beta$ with $0 \leq \alpha, \beta < s$. From (7), the $\mathbf{X}$-dependent part of R_k equals $\bigoplus_{i=0}^{s-1} \big(\mathbf{X}_\alpha^{(0)}[i] \oplus \mathbf{X}_\beta^{(\alpha+1)}[i]\big)$.

Hence

$$\bigoplus_{k=0}^{n-1} R_k = \bigoplus_{\alpha=0}^{s-1}\bigoplus_{\beta=0}^{s-1}\bigoplus_{i=0}^{s-1}\Big(\mathbf{X}_\alpha^{(0)}[i] \oplus \mathbf{X}_\beta^{(\alpha+1)}[i]\Big) \;\oplus\; \text{(analogous } \mathbf{Y}\text{-terms)}. \tag{11}$$

We show that the total contribution of the $\mathbf{X}$-terms in (11) is zero; the $\mathbf{Y}$-terms follow identically.

Split the $\mathbf{X}$-part into two sums:

$$S_1 := \bigoplus_{\alpha=0}^{s-1}\bigoplus_{\beta=0}^{s-1}\bigoplus_{i=0}^{s-1}\mathbf{X}_\alpha^{(0)}[i], \qquad S_2 := \bigoplus_{\alpha=0}^{s-1}\bigoplus_{\beta=0}^{s-1}\bigoplus_{i=0}^{s-1}\mathbf{X}_\beta^{(\alpha+1)}[i].$$

For S_1, the inner expression does not depend on β, so

$$S_1 = \bigoplus_{\beta=0}^{s-1}\left(\bigoplus_{\alpha=0}^{s-1}\bigoplus_{i=0}^{s-1}\mathbf{X}_\alpha^{(0)}[i]\right).$$

Because $\mathcal{C}^{(0)}(\mathbf{X}) = [\mathbf{X}_0^{(0)}, \ldots, \mathbf{X}_{s-1}^{(0)}]$ is a partition of the shares,

$$\bigoplus_{\alpha=0}^{s-1}\bigoplus_{i=0}^{s-1}\mathbf{X}_\alpha^{(0)}[i] = \bigoplus_{j=0}^{n-1} X_j = X.$$

Thus $S_1 = \bigoplus_{\beta=0}^{s-1} X = s \cdot X$.

For S_2, fix α and note that $\mathcal{C}^{(\alpha+1)}(\mathbf{X})$ is also a partition, hence

$$\bigoplus_{\beta=0}^{s-1}\bigoplus_{i=0}^{s-1}\mathbf{X}_\beta^{(\alpha+1)}[i] = \bigoplus_{j=0}^{n-1} X_j = X,$$

so $S_2 = \bigoplus_{\alpha=0}^{s-1} X = s \cdot X$.

Over characteristic 2, $s \cdot X \oplus s \cdot X = 0$, hence the total $\mathbf{X}$-contribution is $S_1 \oplus S_2 = 0$. The same argument applies to the $\mathbf{Y}$-terms in (7), so $\bigoplus_{k=0}^{n-1} R_k = 0$.

The sharing $\mathbf{W}$ *encodes* XY. Again with $k = \alpha s + \beta$, from (7) we have

$$W_k = \mathbf{X}_\alpha^{(0)} \otimes \mathbf{Y}_\beta^{(\alpha+1)} = \left(\bigoplus_{i=0}^{s-1}\mathbf{X}_\alpha^{(0)}[i]\right) \cdot \left(\bigoplus_{i=0}^{s-1}\mathbf{Y}_\beta^{(\alpha+1)}[i]\right).$$

Therefore

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

Conclusion. Combining both parts,

$$\bigoplus_{k=0}^{n-1} Z_k = \left(\bigoplus_{k=0}^{n-1} W_k\right) \oplus \left(\bigoplus_{k=0}^{n-1} R_k\right) = XY \oplus 0 = XY,$$

which proves correctness. □

B Instances of $\mathbb{S}$Mul-DN, $\mathbb{S}$Mul-DU, and $\mathbb{S}_{\boldsymbol{\chi_5}}$

B.1 $\mathbb{S}$Mul-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 $\mathbb{S}$Mul-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 $\mathbb{S}$Mul-DU

For $\mathbb{S}$Mul-DU with $n = 9$, 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 $\mathbb{S}_{\boldsymbol{\chi_5}}$

For an instance of $\mathbb{S}\chi_5$ with $n = 4$, we choose clusters indexed by 0, 1, and 2 as defined in Example 2. Applying Algorithm 4 to the input n-sharings $\left[\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}\right]$, we obtain the corresponding output n-sharings $\left[\mathbf{A}', \mathbf{B}', \mathbf{C}', \mathbf{D}', \mathbf{E}'\right]$ of the masked $\mathbb{S}\chi_5$ transformation. For each

$$(\mathbf{Z}, \mathbf{X}, \mathbf{Y}) \in \{(\mathbf{A}, \mathbf{B}, \mathbf{C}),\ (\mathbf{B}, \mathbf{C}, \mathbf{D}),\ (\mathbf{C}, \mathbf{D}, \mathbf{E}),\ (\mathbf{D}, \mathbf{E}, \mathbf{A})\},$$

the output shares $\mathbf{Z}' = [Z'_0, Z'_1, Z'_2, Z'_3]$ are computed as:

$$\begin{aligned} Z'_0 &= Z_0 \oplus \overline{X_0}Y_0 \oplus \overline{X_0}Y_3 \oplus X_1Y_0 \oplus X_1Y_3, & Z'_1 &= Z_2 \oplus \overline{X_0}Y_1 \oplus \overline{X_0}Y_2 \oplus X_1Y_1 \oplus X_1Y_2, \\ Z'_2 &= Z_1 \oplus X_2Y_0 \oplus X_2Y_3 \oplus X_3Y_0 \oplus X_3Y_3, & Z'_3 &= Z_3 \oplus X_2Y_1 \oplus X_2Y_2 \oplus X_3Y_1 \oplus X_3Y_2. \end{aligned}$$

For the final output $\mathbf{E}'$ (corresponding to the input $(\mathbf{E}, \mathbf{A}, \mathbf{B})$), the share update is:

$$\begin{aligned} Z'_0 &= Z_3 \oplus \overline{X_0}Y_0 \oplus \overline{X_0}Y_3 \oplus X_1Y_0 \oplus X_1Y_3, & Z'_1 &= Z_0 \oplus \overline{X_0}Y_1 \oplus \overline{X_0}Y_2 \oplus X_1Y_1 \oplus X_1Y_2, \\ Z'_2 &= Z_1 \oplus X_2Y_0 \oplus X_2Y_3 \oplus X_3Y_0 \oplus X_3Y_3, & Z'_3 &= Z_2 \oplus X_2Y_1 \oplus X_2Y_2 \oplus X_3Y_1 \oplus X_3Y_2. \end{aligned}$$

Here, $\overline{X_j}$ denotes the bitwise complement $1 \oplus X_j$. Note that the order of XOR operations (from left to right) is relevant and must be preserved to ensure probing security.

C 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 6, 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.

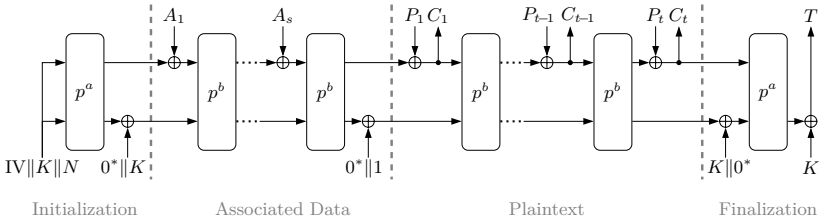

Fig. 6: Ascon. p^a and p^b are applications of the permutation p in a and b rounds.

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 7) 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 [21].

Ascon's S-box. Figure 7 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.

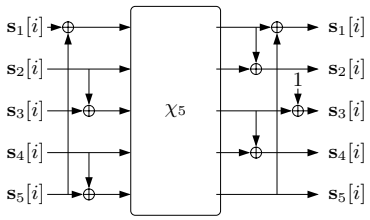

Fig. 7: Ascon's S-box.