Thinking Outside the Superbox

Nicolas Bordes, Joan Daemen, Daniël Kuijsters, Gilles van Assche

Thinking Outside the Superbox

Nicolas Bordes[1], Joan Daemen[2], Daniël Kuijsters[2], and Gilles Van Assche[3]

[1] Université Grenoble Alpes, France
[2] Radboud University, The Netherlands
[3] STMicroelectronics, Belgium
nicolas.bordes@univ-grenoble-alpes.fr, joan.daemen@ru.nl,
Daniel.Kuijsters@ru.nl, gilles-iacr@noekeon.org

Abstract. Designing a block cipher or cryptographic permutation can be approached in many different ways. One such approach, popularized by AES, consists in grouping the bits along the S-box boundaries, e.g., in bytes, and in consistently processing them in these groups. This aligned approach leads to hierarchical structures like superboxes that make it possible to reason about the differential and linear propagation properties using combinatorial arguments. In contrast, an unaligned approach avoids any such grouping in the design of transformations. However, without hierarchical structure, sophisticated computer programs are required to investigate the differential and linear propagation properties of the primitive. In this paper, we formalize this notion of alignment and study four primitives that are exponents of different design strategies. We propose a way to analyze the interactions between the linear and the nonlinear layers w.r.t. the differential and linear propagation, and we use it to systematically compare the four primitives using non-trivial computer experiments. We show that alignment naturally leads to different forms of clustering, e.g., of active bits in boxes, of two-round trails in activity patterns, and of trails in differentials and linear approximations.

Keywords: symmetric cryptography, permutations, block ciphers, round functions

1 Introduction

Modern block ciphers and cryptographic permutations consist of the iteration of a round function. In many cases this round function consists of a layer of nonlinear S-boxes, a mixing layer, a shuffle layer (AKA a bit transposition or bit permutation), and the addition of a round key (in block ciphers) or constant (in cryptographic permutations).

Many papers investigate S-boxes and try to find a good compromise between implementation cost and propagation properties or provide a classification of all invertible S-boxes of a given width, see, e.g., [27, 34]. Similarly, there is a rich literature on certain types of mixing layers. In particular, there have been many papers written about finding maximum-distance separable (MDS) mappings or near-MDS mappings with minimum implementation cost according to some metric, see, e.g., [28,37]. Building a good cipher starts with taking a good S-box and

mixing layer and the rich cryptographic literature on these components provides us with ample choice. However, how these building blocks are combined in a round function and the resulting propagation properties has received much less systematic attention.

A standard way for designing a good round function from an S-box and an MDS mapping is the one followed in the Advanced Encryption Standard (AES) [32] and is known as the *wide trail strategy* [14, 20]. This strategy gives criteria for the shuffle layer and comes with easy-to-verify bounds for the differential probability (DP) of differential trails (also known as characteristics) and the linear potential (LP) of linear trails. These bounds and its simplicity have made it one of the most applied design strategies, and AES has inspired a plethora of primitive designs, including lightweight ones. By adopting 4-bit S-boxes instead of 8-bit ones and modern lightweight MDS layers in a smart structure, multiple lightweight ciphers have been constructed. Many lessons were learned and this line of design has culminated in the block cipher of the NIST lightweight competition candidate Saturnin [12], a truly modern version of AES.

Naturally, there are alternative design approaches. A popular design approach is the one underlying the 64-bit lightweight block cipher Present [10]. Its round function has no MDS layer and simply consists of an S-box layer, a bit shuffle, and a key addition. It gets its diffusion from the combination of a smart choice of the bit shuffle and specific propagation criteria from its well-chosen S-box and doing many rounds. The Present line of design has also been refined in the form of the Gift (64- and 128-bit) block ciphers [1] and the cryptographic permutations of the Spongent lightweight hash function [9] that is used in Elephant [7].

Another distinctive design approach is that of the cryptographic permutation of the SHA-3 standard [33], Keccak-f. Unlike Present, its round function does have a mixing layer, and it actually has all ingredients that AES has. Specifically, in their rationale, the designers also refer to the wide trail design strategy [6]. However, this wide-trail flavor does not appear to come with the simple bounds as in the case of AES, and designers have to resort to tedious and time-consuming programming efforts to obtain similar bounds. This is related to the fact that AES operates on *bytes* and Keccak-f on *bits*. The Keccak-f designers have discussed the difference between these two design approaches in [18]. In that paper, they have coined the term *alignment* to characterize this difference and supported it with some propagation experiments on Keccak-f. The Keccak-f line of design has also been refined and led to the 384-bit permutation that is used in Xoodyak [15], namely Xoodoo [16], a truly modern version of Keccak-f.

This treatment is not exhaustive and other distinctive design strategies exist. Some of them do not even use S-boxes or mixing layers, but they are based on alternating Additions with Rotations and XOR (ARX) such as Salsa [3], or they iterate very simple round functions many times such as Simon [2].

In this paper we systematically analyze the impact of alignment on the differential and linear propagation properties of ciphers. We show that certain design choices regarding how the S-box and mixing layers are combined have a pro-

found impact on the propagation properties. We identify and name a number of effects that are relevant in this context. Furthermore, we believe that this makes it possible to give a meaningful and non-ambiguous definition of the term alignment.

To illustrate this, we study the four primitives Rijndael-256 [22], Saturnin, Spongent-384, and Xoodoo. They have comparable width and all have a non-linear layer consisting of equally-sized S-boxes that have the lowest known maximum DP and LP for their dimensions, see Section 2. They represent the three different design strategies, where we include both Rijndael-256 and Saturnin to illustrate the progress made in the last twenty years. We investigate their difference propagation and correlation properties, where for multiple rounds we adopt a *fixed-key* perspective. This, combined with the choice of relatively wide primitives, is geared towards their usage in permutation-based cryptography, but most findings are also relevant for the key-alternating block cipher case.

1.1 Outline and Contributions

After discussing notation and conventions, we review the notions of differential and linear cryptanalysis in Section 2. In Section 3 we show how the nonlinear layer defines a so-called *box partition*, and we present a non-ambiguous definition of alignment. In Section 4 we present our four ciphers from the perspective of alignment and compare the costs of their round functions. Surprisingly, Spongent, despite being specified at bit level like Keccak-f, turns out to be aligned.

In Section 5 we recall the notions of bit and box weight as a measure of the mixing power of a linear layer. We report on this mixing power by means of *histograms* of states by their weight before and after the linear layer, rather than the usual *branch number* criterion. For all ciphers we observe a decay in mixing power from bit to box weight and describe and name the effect that causes this: *huddling*. This effect is more pronounced in aligned ciphers. This translates directly to the two-round differential and linear trail weight distributions, and we list them for all four ciphers. For the two most competitive proposals, we include histograms for three-round trails and a comparison for four rounds. Remarkably, despite the fact that Saturnin has a more expensive S-box layer and a mixing layer with better bit-level mixing power, Xoodoo has better differential and linear trail histograms for more than two rounds.

In Section 6, we show that trails that cluster necessarily share the same activity pattern, and we introduce the *cluster histogram* as a quantitative tool for the relation between the linear layer and the clustering of two-round trails in ciphers. We see that there is more clustering in the aligned than in the unaligned ciphers. We present the cluster histogram of the four primitives and, for three of them, we also analyze their two-round trail weight histograms. We conclude with a discussion on the clustering of trails in two and three rounds, and show that, at least up to weight 50, differentials over three rounds of Xoodoo admit only one trail, hence they do not cluster.

Finally, in Section 7 we study the independence of round differentials in trails. We show that, again at least up to weight 50, three-round differentials of Xoodoo are independent.

The generation of our histograms was non-trivial and the computation methods could be considered a contribution in themselves. Due to space restrictions we could not treat them in the paper but we have added their description in the supplementary material A after the paper. The related software is available at https://github.com/ongetekend/ThinkingOutsideTheSuperbox under the CC0 license (public domain).

1.2 Notation and Conventions

In this paper, we use the following conventions and notation. We write $\mathbb{Z}_{\geq 0}$ for the nonnegative integers and $\mathbb{Z}_{>0}$ for the positive integers. We write k with $k \in \mathbb{Z}_{\geq 0}$ for nonnegative integer variables. In other words, k is used as a placeholder for any nonnegative integer value.

Whenever we use indices, they always begin at 0. We define $[0,\, k-1] = \{i \in \mathbb{Z}_{\geq 0} : 0 \leq i \leq k-1\}$. Given a set S and an equivalence relation $\sim$ on S, we write $[a]_\sim$ for the equivalence class of $a \in S$. We denote the cardinality of S by $\#S$.

We study permutations $f\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$. Any block cipher is transformed into a permutation by fixing the key, e.g., we fix all of its bits to 0.

We use the term *state* for a vector of b bits. It is either a vector that the permutation is applied to, a difference, or a linear mask (See Section 2). Given a state $a \in \mathbb{F}_2^b$, we refer to its ith component as a_i. In this paper, we consider index sets $\mathrm{B}_i \subseteq [0,\, b-1]$ that form an *ordered* partition. We write $P_i(a)\colon \mathbb{F}_2^b \to \mathbb{F}_2^{\#\,\mathrm{B}_i}$ for the *projection* onto the bits of a indexed by B_i.

We write e_i^k for the ith standard basis vector in $\mathbb{F}_2^k$, i.e., for $j \in [0,\, k-1]$ we have that $e_{ij}^k = 1$ if $i = j$ and 0 otherwise. We write $+$ for vector addition in $\mathbb{F}_2^k$.

Permutations are typically built by composing a number of lightweight *round functions*, i.e., $f = \mathrm{R}_{r-1} \circ \cdots \circ \mathrm{R}_1 \circ \mathrm{R}_0$ for some $r \in \mathbb{Z}_{>0}$. We write $f[r] = \mathrm{R}_{r-1} \circ \cdots \circ \mathrm{R}_0$ and define $f[0] = \mathrm{id}$ with id the identity function. A round function is composed of *step functions*, i.e., $\mathrm{R}_i = \iota_i \circ \mathrm{L}_i \circ \mathrm{N}_i$, where N_i is a nonlinear map, L_i is a linear map, and ι_i is addition of a round constant. Apart from the round constant addition, these round functions are often, but not always, identical. For this reason, we will often simply write N or L, without reference to an index if the context allows for this, and we call N the nonlinear layer of f and L the linear layer of f. We write n for the number of S-boxes of N and denote their size by m. In this context, we suppose that $\mathrm{B}_j = \{jm, \ldots, (j+1)m-1\}$.

Permutations of the index space are written as $\tau\colon [0,\, b-1] \to [0,\, b-1]$. By *shuffle (layer)*, we mean a linear transformation $\pi\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ given by $\pi(a) = \mathrm{P}_\tau\, a$, where P_τ is the permutation matrix associated with some τ, i.e., obtained by permuting the columns of the $(b \times b)$ identity matrix according to τ.

Given a linear transformation $\mathrm{L}\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$, there exists a matrix $\mathrm{M} \in \mathbb{F}_2^{b \times b}$ such that $\mathrm{L}(a) = \mathrm{M}\, a$. We define its transpose $\mathrm{L}^\top\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ by $\mathrm{L}^\top(a) = \mathrm{M}^\top\, a$ and we denote the inverse of $\mathrm{L}^\top$, when it exists, by $\mathrm{L}^{-\top}$.

2 Differential and Linear Cryptanalysis

A major motivation behind the tools developed in this paper is better understanding of the interplay between the linear and nonlinear layer in relation to differential and linear cryptanalysis. We want to be able to use the associated language freely when discussing these tools. Therefore, in this section, we go over the basic notions to make sure they are on hand when needed.

2.1 Differential Cryptanalysis

Differential cryptanalysis [8] is a chosen-plaintext attack that exploits the non-uniformity of the distribution of differences at the output of a permutation when it is applied to pairs of inputs with a fixed difference. We call an ordered pair of an input and output difference $(\Delta_{\text{in}}, \Delta_{\text{out}}) \in (\mathbb{F}_2^b)^2$ a differential.

Definition 1. *Let $f\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ be a permutation and define $U_f(\Delta_{in}, \Delta_{out}) = \{x \in \mathbb{F}_2^b : f(x) + f(x + \Delta_{in}) = \Delta_{out}\}$. We call $U_f(\Delta_{in}, \Delta_{out})$ the solution set of the differential $(\Delta_{in}, \Delta_{out})$.*

Definition 2. *The* differential probability *(*DP*) of a differential $(\Delta_{in}, \Delta_{out})$ over the permutation $f\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ is defined as $\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) = \frac{\#U_f(\Delta_{in}, \Delta_{out})}{2^b}$.*

If there exists an ordered pair $(x, x + \Delta_{\text{in}})$ with $x \in U_f(\Delta_{\text{in}}, \Delta_{\text{out}})$, then it is said to follow the differential $(\Delta_{\text{in}}, \Delta_{\text{out}})$. In this case, we say that the input difference Δ_{in} is *compatible* with the output difference Δ_{out} through f and call $(\Delta_{\text{in}}, \Delta_{\text{out}})$ a *valid* differential.

Definition 3. *A sequence $Q = (q^{(0)}, q^{(1)}, \ldots, q^{(k)}) \in \left(\mathbb{F}_2^b\right)^{k+1}$ that satisfies $\mathrm{DP}_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)}) > 0$ for $0 \leq i \leq k-1$ is called a k-round* differential trail.

Sometimes we specify a trail as $Q = (b_{-1}, a_0, b_0, \ldots, a_k, b_k)$ by giving the intermediate differences between N_i and L_i as well, where $b_i = \mathrm{L}_i(a_i) = q_{i+1}$. We write $\mathrm{DT}(\Delta_{\text{in}}, \Delta_{\text{out}})$ for the set of all differential trails in the differential $(\Delta_{\text{in}}, \Delta_{\text{out}})$, so with $q^{(0)} = \Delta_{\text{in}}$ and $q^{(k)} = \Delta_{\text{out}}$. We call $(\Delta_{\text{in}}, \Delta_{\text{out}})$ the *enveloping differential* of the trails in $\mathrm{DT}(\Delta_{\text{in}}, \Delta_{\text{out}})$. If $\#\mathrm{DT}(\Delta_{\text{in}}, \Delta_{\text{out}}) > 1$, then we say that trails *cluster* together in the differential $(\Delta_{\text{in}}, \Delta_{\text{out}})$.

By deleting the initial difference Δ_{in} and final difference Δ_{out} of a differential trail $(\Delta_{\text{in}}, q^{(1)}, \ldots, q^{(k-1)}, \Delta_{\text{out}})$ we are left with a *differential trail core*. A differential trail core obtained in this way is said to be in the differential $(\Delta_{\text{in}}, \Delta_{\text{out}})$. Note that a differential trail core actually defines a set of differential trails with the same inner differences.

We now define the DP of a differential trail. Each round differential $(q^{(i)}, q^{(i+1)})$ has a solution set $U_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)})$. Consider the transformed set of points $U_i = f[i]^{-1}(U_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)}))$ at the input of f. For an ordered pair $(x, x + q^{(0)})$ to follow the differential trail, it is required that $x \in U_f(Q) = \bigcap_{i=0}^{k-1} U_i$. The fraction of states x that satisfy this equation is the DP of the trail.

Definition 4. *The* DP *of a differential trail is defined as* $\mathrm{DP}_f(Q) = \frac{\#U_f(Q)}{2^b}$.

Definition 5. *The round differentials are said to be* independent *if*

$$\mathrm{DP}_f(Q) = \prod_{i=0}^{k-1} \mathrm{DP}_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)})\,.$$

Any given ordered pair $(x, x + \Delta_{\mathrm{in}})$ follows exactly one differential trail. Hence, the DP of the differential $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ is the sum of the DPs of all differential trails with initial difference Δ_{in} and final difference Δ_{out}.

$$\mathrm{DP}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) = \sum_{Q \in \mathrm{DT}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})} \mathrm{DP}_f(Q)\,.$$

Given any differential $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ over a round function R, it is easy to compute its DP value. By specifying the intermediate differences we obtain a differential trail $(\Delta_{\mathrm{in}}, b, c, \Delta_{\mathrm{out}})$. Thanks to the linearity of L, we have $c = \mathrm{L}(b)$ and due to the fact that a difference is invariant under addition of a constant, all valid such differential trails are of the form $(\Delta_{\mathrm{in}}, \mathrm{L}^{-1}(\Delta_{\mathrm{out}}), \Delta_{\mathrm{out}}, \Delta_{\mathrm{out}})$. Therefore, the differential $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ contains only a single trail and its DP is the DP of the differential $(\Delta_{\mathrm{in}}, \mathrm{L}^{-1}(\Delta_{\mathrm{out}}))$ over the S-box layer:

$$\mathrm{DP}_{\mathrm{R}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) = \prod_{0 \le j < n} \mathrm{DP}_{\mathrm{S}_j}(P_j(\Delta_{\mathrm{in}}), P_j(\mathrm{L}^{-1}(\Delta_{\mathrm{out}})))\,.$$

Hence, the DP of a round differential is the product of the DP values of its S-box differentials.

Definition 6. *The* restriction weight *of a differential* $(\Delta_{in}, \Delta_{out})$ *that satisfies* $\mathrm{DP}_f(\Delta_{in}, \Delta_{out}) > 0$ *is defined as* $\mathrm{w_r}(\Delta_{in}, \Delta_{out}) = -\log_2 \mathrm{DP}_f(\Delta_{in}, \Delta_{out})$.

For a differential trail, we sum the weights of the round differentials.

Definition 7. *The restriction weight of a differential trail* $Q = (q^{(0)}, q^{(1)}, \ldots, q^{(k)})$ *is defined as*

$$\mathrm{w_r}(Q) = \sum_{i=0}^{k-1} \mathrm{w_r}(q^{(i)}, q^{(i+1)})\,.$$

If the round differentials are independent in the sense of Definition 5, then we have that $\mathrm{DP}_f(Q) = 2^{-\mathrm{w_r}(Q)}$.

2.2 Linear Cryptanalysis

Linear cryptanalysis [29] is a known-plaintext attack. It exploits large correlations (in absolute value) between linear combinations of input bits and linear combinations of output bits of a permutation.

Definition 8. *The (signed)* correlation *between the linear mask $u \in \mathbb{F}_2^b$ at the input and the linear mask $v \in \mathbb{F}_2^b$ at the output of a function $f\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ is defined as*

$$\mathrm{C}_f(u,v) = \frac{1}{2^b} \sum_{x \in \mathbb{F}_2^b} (-1)^{u^\top x + v^\top f(x)} .$$

If $\mathrm{C}_f(u,v) \neq 0$, then we say that u is compatible with v. We call the ordered pair of linear masks (u,v) a *linear approximation*. We note that in the literature (e.g., in the linear cryptanalysis attack by Matsui [29]) the term linear approximation has several meanings. It should not be confused with what we call a linear trail.

Definition 9. *A sequence $Q = (q^{(0)}, q^{(1)}, \ldots, q^{(k)}) \in \left(\mathbb{F}_2^b\right)^{k+1}$ that satisfies $\mathrm{C}_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)}) \neq 0$ for $0 \leq i \leq k-1$ is called a* linear trail.

We write $\mathrm{LT}(u,v)$ for the set of all linear trails in the linear approximation (u,v), so with $q^{(0)} = u$ and $q^{(k)} = v$. We call (u,v) the *enveloping linear approximation* of the trails in $\mathrm{LT}(u,v)$. If $\#\mathrm{LT}(u,v) > 1$, then we say that trails *cluster* together in the linear approximation (u,v).

By deleting the initial linear mask u and final linear mask v of a linear trail $(u, q^{(1)}, \ldots, q^{(k-1)}, v)$ we are left with a *linear trail core*. A linear trail core obtained in this way is said to be in the linear approximation (u,v). Note that a linear trail core actually defines a set of linear trails with the same inner linear masks.

Definition 10. *The* correlation contribution *of a linear trail Q over f equals*

$$\mathrm{C}_f(Q) = \prod_{i=0}^{k-1} \mathrm{C}_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)}) .$$

From the theory of correlation matrices [14], it follows that

$$\mathrm{C}_f(u,v) = \sum_{Q \in \mathrm{LT}(u,v)} \mathrm{C}_f(Q) .$$

Given any linear approximation (u,v) over a round function R, it is easy to compute its correlation. By specifying the intermediate linear masks we obtain a linear trail (u,b,c,v). Thanks to the linearity of L, we have $b = \mathrm{L}^\top(c)$ and due to the fact that a linear mask is invariant under addition of a constant, all valid such linear trails are of the form $(u, \mathrm{L}^\top(v), v, v)$. Hence the linear approximation (u,v) contains only a single trail and its correlation contribution is the correlation of the linear approximation $(u, \mathrm{L}^\top(v))$ over the S-box layer, where the round constant addition affects the sign:

$$\mathrm{C}_{\mathrm{R}}(u,v) = (-1)^{v^\top \iota(0)} \prod_{0 \leq j < n} \mathrm{C}_{\mathrm{S}_j}(P_j(u), P_j(\mathrm{L}^\top(v))) .$$

Definition 11. *The* linear potential *(*LP*) of a linear approximation* (u, v) *is defined as* $\mathrm{LP}_f(u,v) = \mathrm{C}_f(u,v)^2$.

Analogous to the differential cryptanalysis case, we define a weight metric.

Definition 12. *The* correlation weight *of a linear approximation* (u, v) *with* $\mathrm{LP}_f(u,v) \neq 0$ *is given by* $\mathrm{w_c}(u,v) = -\log_2 \mathrm{LP}_f(u,v)$.

Definition 13. *The correlation weight of a linear trail* $Q = (q^{(0)}, q^{(1)}, \ldots, q^{(k)})$ *is defined as*

$$\mathrm{w_c}(Q) = \sum_{i=0}^{k-1} \mathrm{w_c}(q^{(i)}, q^{(i+1)})\,.$$

3 Box Partitioning and Alignment

In this section, we consider the partition of the index space defined by the nonlinear layer N. The *alignment* properties of the other step functions with respect to this partition have an important impact on the propagation properties of the round function.

The nonlinear layer N consists of the parallel application of n S-boxes of size m to disjoint parts of the state, indexed by B_i. Formally, this means that we can write N as $\mathrm{S}_0 \times \cdots \times \mathrm{S}_{n-1}$ and that it is characterized by

$$P_i \circ (\mathrm{S}_0 \times \cdots \times \mathrm{S}_{n-1}) = \mathrm{S}_i \circ P_i \text{ for } 0 \leq i \leq n-1\,.$$

Hence, N defines a unique *ordered* partition $\Pi_\mathrm{N} = (\mathrm{B}_0, \ldots, \mathrm{B}_{n-1})$ of the index space $[0,\, b-1]$. We call Π_N the *box partition* defined by N and the B_i N-boxes. If there is no ambiguity, we call the box partition Π and its members boxes.

Besides the box partition, it is clearly possible to define other partitions of the index space as well. We call a partition *non-trivial* if it has at least two members. Between any two partitions of the index space there may be a relation that we denote as *refinement*.

Definition 14. *We call* Π *a* refinement *of* Π' *and write* $\Pi \leq \Pi'$ *if for every* $(i, \mathrm{B}_i) \in \Pi$ *there exists a* $(j, \mathrm{B}'_j) \in \Pi'$ *such that* $\mathrm{B}_i \subseteq \mathrm{B}'_j$.

Let Π be a partition of the index space consisting of k boxes, each of size l. We call a shuffle layer a Π-shuffle if the associated permutation matrix can be partitioned into k identity matrices of dimension $(l \times l)$. If this is the case, then bit index permutation can be specified as a box index permutation.

Definition 15. *We call* $\phi\colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ aligned *to* Π *if we can decompose it as*

$$\phi_0 \times \cdots \times \phi_{k-1}\colon \bigtimes_{i=0}^{k-1} \mathbb{F}_2^l \to \bigtimes_{i=0}^{k-1} \mathbb{F}_2^l\,,$$

In this case, we call the ϕ_i *box functions*.

Definition 16. *Given a round function that is composed of the parallel application* N *of equally-sized S-boxes, a linear layer* L*, and the addition* ι *of a round constant, we say it is* aligned *if it is possible to decompose the linear layer* L *as* $\mathrm{L} = \pi \circ \mathrm{M}$ *in such a way that*

- π *is a* Π_{N}*-shuffle;*
- M *is aligned to a non-trivial partition* Π_{M} *that satisfies* $\Pi_{\mathrm{N}} \leq \Pi_{\mathrm{M}}$*.*

We assume that the split between the linear and nonlinear layer is chosen so as to maximize the number of S-boxes in N*.*

Note that ι does not play a role in the alignment properties. If all of the round functions of a primitive are aligned, then we call the primitive aligned. If the primitive is *not* aligned, then we call it *unaligned*.

Any aligned primitive has a *superbox* structure [35], that is helpful when investigating distributions and bounds on the DP of two-round differentials and the LP of two-round trails. We explain what this means. Consider a two-round structure: $\pi \circ \mathrm{M} \circ \mathrm{N} \circ \pi \circ \mathrm{M} \circ \mathrm{N}$. The final two linear steps π and M have no effect on the distributions, so we can simplify this expression to $\mathrm{N} \circ \pi \circ \mathrm{M} \circ \mathrm{N}$. Clearly, $\mathrm{N} \circ \pi = \pi \circ \mathrm{N}'$, with $\mathrm{N}' := \pi^{-1} \circ \mathrm{N} \circ \pi$. Hence, this is equivalent to $\pi \circ \mathrm{N}' \circ \mathrm{M} \circ \mathrm{N}$. Discarding the shuffle layer at the end gives $\mathrm{N}' \circ \mathrm{M} \circ \mathrm{N}$. Since $\Pi_{\mathrm{N}'} = \Pi_{\mathrm{N}} \leq \Pi_{\mathrm{M}}$, we can view this as the parallel application of a number of superboxes. We call this a *superbox layer*. In a sequence of two rounds, $\mathrm{N}' \circ \mathrm{M} \circ \mathrm{N}$ is a (composite) nonlinear layer and $\pi \circ \mathrm{M} \circ \pi$ is a (composite) linear layer. If the latter is aligned to a non-trivial partition Π such that $\Pi_{\mathrm{M}} \leq \Pi$, then we call this two-round structure aligned to Π_{M}.

4 The Ciphers We Investigate

In this section we describe the round functions of the ciphers we investigate in this paper, their alignment properties, and compare their implementation cost.

4.1 Rijndael

Rijndael [22] is a block cipher family supporting all block and key lengths of $b = 32k$ bits, with $4 \leq k \leq 8$, i.e., ranging from 128 up to and including 256 bits. The case $b = 128$ is of great importance as Rijndael with that block length is the ubiquitous AES [32]. In this paper we investigate Rijndael-256, the instance with $b = 256$, a width closer to those of the other ciphers we investigate. In the remainder of this paper we will write Rijndael for Rijndael-256.

The Rijndael round function consists of four steps: a nonlinear layer `SubBytes`, a box shuffle `ShiftRows`, a mixing layer `MixColumns`, and round key addition `AddRoundKey`. As its name suggest, $\Pi_{\texttt{SubBytes}}$ partitions the state in bytes and `ShiftRows` is a $\Pi_{\texttt{SubBytes}}$-shuffle. The mixing layer, `MixColumns`, is aligned to a non-trivial partition $\Pi_{\texttt{MixColumns}}$ that corresponds to the 8 *columns*, each containing 4 bytes, and we have $\Pi_{\texttt{SubBytes}} \leq \Pi_{\texttt{MixColumns}}$. It follows that Rijndael is aligned. Figure 1 shows Rijndael-128 that is is easier to draw due to its dimensions, but the alignment properties for Rijndael-256 are the same.

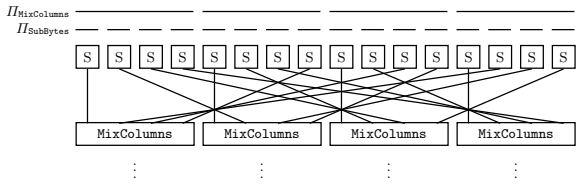

Fig. 1: Alignment properties of RIJNDAEL.

4.2 Saturnin

The SATURNIN [12] block cipher has a 256-bit key and block length. The state has several representations: three-dimensional, two-dimensional, and flat. In three dimensions, the 256-bit state is represented as a $4 \times 4 \times 4$ cube of 4-bit *nibbles*. Nibbles in the cube are indexed by triples (x, y, z). A *slice* is a subset of the nibbles with z constant. A *sheet* is a subset of the nibbles with x constant. A *column* is a subset of the nibbles with x and z constant.

The SATURNIN permutation is composed of a number of so-called super-rounds and a super-round consists of two consecutive rounds with indices $2r$ and $2r + 1$. Round $2r$ is composed as $\mathrm{MC} \circ \mathrm{S}$, where MC is a mixing layer and S is a nonlinear layer. There are two different rounds with odd indices. Round $4r + 1$ is composed as follows: $\mathrm{RC} \circ \mathrm{RK} \circ \mathrm{SR}_{\text{slice}}^{-1} \circ \mathrm{MC} \circ \mathrm{SR}_{\text{slice}} \circ \mathrm{S}$. Round $4r + 3$ consists of $\mathrm{RC} \circ \mathrm{RK} \circ \mathrm{SR}_{\text{sheet}}^{-1} \circ \mathrm{MC} \circ \mathrm{SR}_{\text{sheet}} \circ \mathrm{S}$. Here, RC denotes addition of a round constant, RK denotes addition of a round key, and $\mathrm{SR}_{\text{slice}}$ and $\mathrm{SR}_{\text{sheet}}$ shuffle nibbles. The partition Π_{S} divides the state into 64 nibbles. The shuffles $\mathrm{SR}_{\text{slice}}$ and $\mathrm{SR}_{\text{sheet}}$ are Π_{S}-shuffles. The mixing layer MC is aligned to a non-trivial partition Π_{MC} that divides the state into 16 columns, each consisting of 4 nibbles, and that satisfies $\Pi_{\mathrm{S}} \leq \Pi_{\mathrm{MC}}$. It follows that SATURNIN is aligned. In a super-round we identify the sequence $\mathrm{S} \circ \mathrm{MC} \circ \mathrm{S}$ as a superbox layer with partition Π_{MC} and the linear layer of such a round is $\mathrm{SR}_{\text{slice}}^{-1} \circ \mathrm{MC} \circ \mathrm{SR}_{\text{slice}}$. This is a mixing layer that is aligned to a non-trivial partition Π_{slice} that divides the state into 4 slices, each containing 4 columns, and we have $\Pi_{\mathrm{MC}} \leq \Pi_{\text{slice}}$. Similarly, for the other type of super-round, the mixing layer is aligned to a non-trivial partition Π_{sheet} that divides the state into 4 sheets, and we have $\Pi_{\mathrm{MC}} \leq \Pi_{\text{sheet}}$. It follows that the super-rounds of SATURNIN are aligned and hence have their own superboxes. These have width 64 bits and we call them *hyperboxes*. Figure 2 shows the alignment properties of the steps.

4.3 Spongent

SPONGENT [9] is a sponge-based hash function family that uses a PRESENT-like permutation. The permutation is defined for any b that is a multiple of 4. In this paper, we only consider the case $b = 384$, to match the state size of the largest of the other permutations that we investigate, XOODOO. The round function of SPONGENT consists of three steps: a round constant addition `lCounter`, a 4-bit S-box layer `sBoxLayer`, and a bit shuffle `pLayer`.

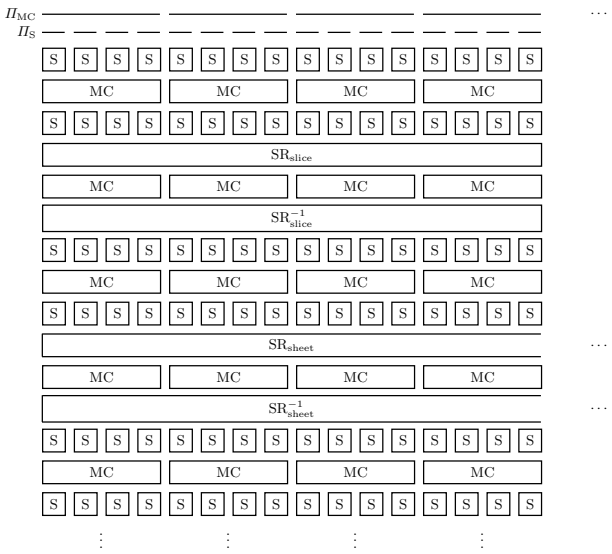

Fig. 2: Alignment properties of SATURNIN.

The index permutation of the bit shuffle `pLayer` is:

$$\texttt{pLayer}(j) = \begin{cases} 96j \bmod 383, & \text{if } j \in [0,\, 382] \\ 383, & \text{if } j = 383 \end{cases}$$

As indicated by the Spongent designers in [9], we can decompose it into a mixing layer, followed by a box shuffle:

1. `SpongentMixLayer` applies the same mixing function `SpongentMix` in parallel to the 24 *subgroups* (following the terminology of [9]). It is a bit shuffle associated with the index permutation $\tau_{\mathrm{subgroup}} \colon [0,\, 15] \to [0,\, 15]$:

$$\tau_{\mathrm{subgroup}}(j) = \begin{cases} 4j \bmod 15, & \text{if } j \in [0,\, 14] \\ 15, & \text{if } j = 15 \end{cases}$$

2. `SpongentBoxShuffle` is a box shuffle that is associated with the box index permutation $\tau_{\mathrm{box}} \colon [0,\, 95] \to [0,\, 95]$ defined by:

$$\tau_{\mathrm{box}}(j) = \left\lfloor \frac{j}{4} \right\rfloor + 24(j \bmod 4)\,.$$

The `sBoxLayer` defines a box partition $\Pi_{\texttt{sBoxLayer}}$ corresponding to the 96 4-bit boxes. The box shuffle `SpongentBoxShuffle` is a $\Pi_{\texttt{sBoxLayer}}$-shuffle. The bit shuffle `SpongentMixLayer` is aligned to a non-trivial partition $\Pi_{\texttt{SpongentMixLayer}}$ that divides the state into 96 16-bit subgroups, each grouping four consecutive boxes, and we have $\Pi_{\texttt{sBoxLayer}} \leq \Pi_{\texttt{SpongentMixLayer}}$. It follows that SPONGENT is aligned. Figure 3 shows these steps and their alignment properties.

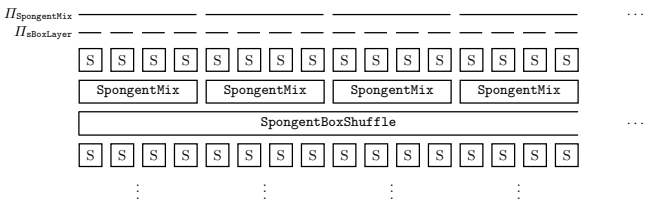

Fig. 3: Alignment properties of SPONGENT.

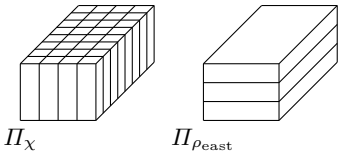

Fig. 4: Alignment properties of XOODOO.

4.4 Xoodoo

XOODOO [16] is a permutation with $b = 384$. The state consists of 3 equally sized horizontal *planes*, each one consisting of 4 parallel 32-bit *lanes*. Alternatively, the state can be seen as a set of 128 *columns* of 3 bits, arranged in a 4×32 array.

The round function of XOODOO consists of the following five steps: a mixing layer θ, a bit shuffle ρ_{east}, round constant addition ι, a nonlinear layer χ, and a bit shuffle ρ_{west}. The χ step applies the same 3-bit S-box to the columns of the state. The nonlinear layer χ defines a box partition Π_χ that corresponds to the 128 columns. The bit shuffles ρ_{east} and ρ_{west} perform translations of planes and are not aligned to Π_χ. The mixing layer θ defines no non-trivial box partition at all. Due to the properties of the ρ steps and θ it is impossible to split the linear layer in a column shuffle and a mixing layer that is aligned to a partition that Π_χ is a refinement of. In other words, XOODOO is unaligned. See Section E of the supplementary material for a more formal proof. Figure 4 shows the alignment properties of the steps.

4.5 Round Cost

In this section, we compare the implementation complexity of the round functions of the four ciphers. This depends on the platform and the requirements. Platforms may range from low-end 8-bit CPUs to multi-core high-end workstation CPUs, FPGAs, and even dedicated hardware. Requirements include throughput, latency, usage of resources such as power and energy consumption, area in hardware, and RAM/ROM usage in software. Moreover, protection against fault attacks and/or side channel attacks may be required.

In our comparison of the round functions we let their three layers guide us: the S-box layer, the mixing layer (if any), and the shuffle layer. We also discuss the presence of key addition in block ciphers and its relative cost.

Table 1: S-box computational cost comparison.

cipher	max DP/LP	operations in $\mathbb{F}_2$ ref.	# operations xor	and/or	not	2-layer nand circuit # nand gates per # inputs 2-in	3-in	4-in	5-in	6-in	7-in	totals gates	inp.
Rijndael	2^{-6}	[11,36]	81	32	4	?							
Saturnin	2^{-2}	[12]	6	6	-	4	5	6	1	-	-	16	52
Spongent	2^{-2}		?			-	6	8	-	3	1	18	75
Xoodoo	2^{-2}	[16]	3	3	3	3	6	-	-	-	-	9	24

S-box Layer Given that our ciphers have invertible S-boxes with lowest known maximum DP and LP values that can be achieved for their width, their implementation cost increases with width.

We report on the implementations with minimum number of binary XOR, binary AND/OR, and unary NOT operations that we found in the literature. For Spongent we found no such numbers. We have also determined a minimal sum-of-products (SOP) form in Boolean algebra of the S-boxes using the Espresso algorithm [30] for two-level logic optimization. For Rijndael, finding the minimal SOP was infeasible. We refer to Section B of the supplementary material for the SOP expressions. Using De Morgan's laws, the SOP form can be implemented by two layers of nand gates. Table 1 lists the number of nand gates per bit for each of the S-boxes.

We can see in Table 1 that the cost of the Saturnin and Spongent S-boxes is comparable. The cost of the Xoodoo S-box is roughly half of that, but is only 3 bits wide instead of 4. The Rijndael S-box is a roughly a factor 10 more costly than that of Saturnin and Spongent, a very high price for its better max DP/LP value. These numbers give an indication for the size of a hardware circuit and the number of cycles in bit-sliced software implementations. The number of and/or operations is related to the cost of masking countermeasures.

Mixing Layer Spongent has no mixing layer, so there is no cost. Xoodoo-θ requires 2 binary xor operations per bit, while Saturnin's MC can be implemented with 2.25 binary xor operations per bit [12]. The circuit depth for these computations is in both cases 4 xor gates. Despite the difference in design philosophy, their computational costs are almost the same.

A simple implementation of Rijndael's `MixColumns` takes 3.875 binary xor operations per bit and has a circuit depth of 3 xor gates. This was reduced to $97/32 \approx 3$ additions per bit [25] at the expense of a higher circuit depth. Despite the fact that both `MixColumns` and Saturnin's MC implement an MDS mapping operating on 5 boxes, their costs diverge. The main difference between the two is that `MixColumns` operates on bytes while MC operates on nibbles. However, this is not the reason for the higher cost per bit of `MixColumns`. The reason is

Table 2: The cost of a round in cycles per byte on the ARM Cortex-M4.

Cipher	# cycles/byte
Rijndael [38]	10.0
Saturnin [13]	2.7
Spongent	?
Xoodoo [5]	1.1

that there have been significant advances in building efficient MDS mappings and MC reaps the benefits of that.

Shuffle Layer Rijndael, Spongent, and Xoodoo consist of the iteration of a single round function. In a hardware architecture that implements the full round in combinatorial logic, a bit shuffle consists of wiring between gates. Saturnin has three different rounds, so this is more complex in a hardware architecture in which a single round is implemented in combinatorial logic. However, in a combinatorial block that implements a sequence of four rounds, the shuffle operations do correspond to wiring.

We compare software implementation on a particular platform: the ARM Cortex-M4 processor. We choose this because it is a popular lightweight platform for benchmarks and for three of our ciphers there is assembly code available. On this platform, it is difficult to assess the cost of the shuffle layer in isolation due to the *barrel shifter*. This feature of the ARM architecture allows applying (cyclic) shift operations to one of the two operands in arithmetic and bitwise Boolean instructions at no additional cost. To compare, we measure the number of cycles of the entire round function, revealing the marginal cost of the shuffle layer. Table 2 lists the performance of the round functions of our four ciphers expressed in number of cycles per byte as measured on a Cortex-M4 processor. In addition, it includes references to the bit-sliced implementations that we have used in order to measure the cycle counts. In Rijndael and Saturnin we removed any operations related to the key addition to make a fair comparison possible and in Saturnin we measured the number of cycles for 4 rounds and divided that by 4. We have not included Spongent because we do not have access to any (optimized) assembly code. However, considering that it was designed with hardware in mind, we do not believe it is competitive in software.

5 Huddling

In this section, we describe a phenomenon that we call *huddling*. We present the bit and box weight histograms as natural extensions of the bit and box branch numbers, respectively. Using these histograms, we analyze the huddling properties of the ciphers described in Section 4. We see that these properties

are more pronounced in ciphers that are aligned. Finally, we look at the relation between huddling and the distribution of trail weights.

5.1 Definitions of Bit Weight, Box Weight and their Histograms

The weight of a two-round trail $(q_{\text{in}}, a, b, q_{\text{out}})$ over $\mathrm{N} \circ \mathrm{L} \circ \mathrm{N}$ can be bounded from below by the sum of the number of active boxes at the input and output of L. This number is fully determined by a as $b = \mathrm{L}(a)$ in differential trails and $a = \mathrm{L}^\top(b)$ in linear trails. The distribution of states a according to this number determines the *mixing power* of the linear layer with respect to Π_{N}.

First, we formally define what it means for a box to be active. To this end, we define an *indicator function* $1_i \colon \mathbb{F}_2^b \to \mathbb{F}_2$ with respect to a box partition Π by $1_i(a) = 0$ if $P_i(a) = 0$ and $1_i(a) = 1$ otherwise. We call the box B_i *active* in the difference or linear mask $a \in \mathbb{F}_2^b$ if $1_i(a) = 1$ and *passive* otherwise. The natural metric associated with box activity is the *box weight* of a, defined by $\mathrm{w}_\Pi(a) = \#\{i \in [0, n-1] : 1_i(a) \neq 0\}$. Clearly, a box is active in a difference or linear mask if at least one of the bits in that box is non-zero. We call the bit i *active* in a if $a_i = 1$ and *passive* otherwise. The number of active bits is given by the *bit weight* of a, i.e., $\mathrm{w}_2(a) = \#\{i \in [0, b-1] : a_i \neq 0\}$. The *activity pattern* of a is defined by $r_\Pi(a) = \sum_{i=0}^{n-1} 1_{\mathrm{B}_i}(a) e_i^n$. It is the vector whose ith component is one if box B_i is active and zero otherwise.

In order to quantify the mixing power of a linear transformation L, we consider the weight distribution of $(a, \mathrm{L}(a))$ over all differences or linear masks $a \in \mathbb{F}_2^b$ and embed it in a histogram. This is a well-known concept in coding theory, where weight distributions are embedded in so-called weight enumerator polynomials that classify the code [23].

Definition 17. *The* weight histogram *of a linear transformation* $\mathrm{L} \colon \mathbb{F}_2^b \to \mathbb{F}_2^b$ *is a function* $\mathcal{N}_{\cdot,\mathrm{L}} \colon \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ *given by*

$$\mathcal{N}_{\cdot,\mathrm{L}}(k) = \#\{a \in \mathbb{F}_2^b : \mathrm{w}_\cdot(a) + \mathrm{w}_\cdot(\mathrm{L}(a)) = k\}\,.$$

The cumulative version on the same domain and codomain is given by

$$\mathcal{C}_{\cdot,\mathrm{L}}(k) = \sum_{l \leq k} \mathcal{N}_{\mathrm{L}}(l)\,.$$

Here, $\cdot$ *denotes either* 2 *or* Π*.*

The *tail* of the histogram consists of the left-most values that correspond to low weight.

If the primitive is aligned, then π is a box shuffle and this implies that the box weight histograms of $\mathrm{L} = \mathrm{M} \circ \pi$ and M are the same. The superbox structure of an aligned primitive makes it possible to use a divide-and-conquer approach to compute the weight histograms. Indeed, let $S(\mathrm{w}) = \{v \in \mathbb{Z}_{\geq 0}^s : \sum_{i=0}^{s-1} v_i = \mathrm{w}\}$

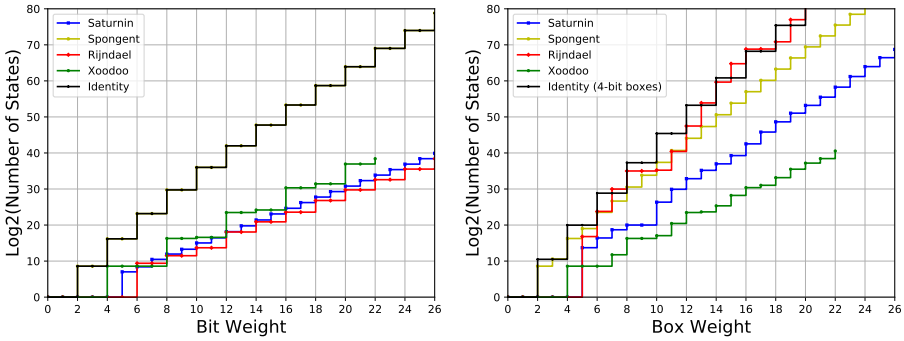

Fig. 5: Cumulative bit weight and box weight histograms.

with s the number of superboxes. Then we can compute the weight histograms of M by *convolving* the weight histograms of its box functions:

$$\mathcal{N}_{\cdot,\mathrm{M}}(\mathrm{w}) = \sum_{v \in S(\mathrm{w})} \prod_{i=0}^{s-1} \mathcal{N}_{\cdot,\mathrm{M}_i}(v_i)\,. \tag{1}$$

We note that the *differential branch number* [14] is simply the smallest non-zero entry of this histogram, i.e., $\min\{\mathrm{w} > 0 \;:\; \mathcal{N}_{\cdot,\mathrm{L}}(\mathrm{w}) > 0\}$. The linear branch number is the smallest non-zero entry in the corresponding histogram of $\mathrm{L}^\top$ and can be different from its differential counterpart. This is not the case for the mappings in this paper and we will omit the qualifier in the remainder. A higher branch number typically implies higher mixing power. However, the weight histogram is more informative than just the branch number. The number of differences or linear masks meeting the branch number is valuable information as well. In general, the weight histogram allows a more nuanced comparison of mixing layers than the branch number.

The box weight histogram is the relevant histogram in the context of the wide trail design strategy [20]. A linear layer that systematically has lower values in the tail of its box weight histogram than the other does typically has fewer two-round trails with low weight, given equal nonlinear layers.

5.2 Bit and Box Weight Histograms

We discuss the cumulative bit and box weight histograms for the linear layers of our four ciphers, given in Figure 5. We include the histogram for the identity function, assuming 4-bit S-boxes for the box weight to allow for comparison with Spongent and Saturnin.

The bit weight histogram for Spongent coincides with that of the identity permutation. This is because its linear layer is a bit shuffle. As the identity permutation maps inputs to identical outputs, it has only non-zero entries for

even bit weights. Its bit branch number is 2. In conclusion, its mixing power is the lowest possible.

The bit branch number of the mixing layer of Rijndael, `MixColumns`, is 6, that of Saturnin-MC is 5, and that of Xoodoo-θ is 4.

Similar to Spongent, the bit weight histograms of Rijndael and Xoodoo have only non-zero entries at even bit weights. This is because both Xoodoo-θ and Rijndael-`MixColumns` can be modeled as $a \mapsto (\mathrm{I}+\mathrm{M})a$ for some matrix $\mathrm{M} \in \mathbb{F}_2^{b\times b}$ with the property that the bit weight of $\mathrm{M}\,a$ is even for all $a \in \mathbb{F}_2^b$. Saturnin-MC cannot be modeled in that way and does have non-zero entries at odd bit weights.

The bit weight histograms of Rijndael and Saturnin are very close and that of Xoodoo is somewhat higher. The ranking per bit weight histogram reflects the computational resources invested in the mixing layer: Rijndael uses 3.5 additions per bit, Saturnin 2.25, Xoodoo 2, and Spongent 0.

In the box weight histograms we see the following. For Spongent the box branch number is 2, the same as the bit branch number. However, the box weight histogram of Spongent has a lower tail than the identity permutation. What it shows is the mixing power of `SpongentMixLayer` in our factorization of `pLayer`, operating on 4-box superboxes.

The box branch number of the linear layers of Rijndael, `MixColumns`, and of Saturnin-MC are both 5, while for Xoodoo it is 4.

The discrepancy between the bit and box weight histogram brings us to the notion of *bit huddling*: many active bits huddle together in few active boxes. We say that the bit huddling in a linear layer is *high* if the concentration is high and we say that the bit huddling is *low* otherwise.

Huddling has an effect on the contribution of states a to the histogram, i.e., by definition we have that $\mathrm{w}_{\Pi}(a)+\mathrm{w}_{\Pi}(\mathrm{L}(a)) \leq \mathrm{w}_2(a)+\mathrm{w}_2(\mathrm{L}(a))$. In words, from bit to box weight, huddling moves states to the left in the histogram, thereby raising the tail. Huddling therefore results in the decay of mixing power at box level as compared to bit level. In the absence of huddling, the bit and box weight histogram would be equal. However, huddling cannot be avoided altogether as states do exist with multiple active bits in a box (note that $m \geq 2$).

We see Rijndael has *high* bit huddling. In moving from bit weights to box weights, the branch number decreases from 6 to 5 and the tail rises from being the lowest of the four to the highest. This is a direct consequence of the large width of the Rijndael S-boxes, namely 8, and the byte alignment. Indeed, `MixColumns` only mixes bits within the 32-bit columns. We call this the *superbox huddling effect*. Of course, there is a reason for these large S-boxes: they have low maximum DP/LP values. They were part of a design approach assuming table-lookup implementations where the main impact of the S-box size is the size of the lookup tables. Unfortunately table-lookups are expensive in dedicated hardware and on modern CPUs lookup tables are kept in cache making such implementations susceptible to cache-timing attacks [4].

Saturnin, with its Rijndael-like structure also exhibits the superbox huddling effect, though less pronounced than Rijndael. From bits to boxes the

branch number does not decrease and the tail rises less than for Rijndael. Clearly, its smaller S-box size, namely 4, allows for less bit huddling. Due to its alignment, Spongent exhibits the superbox huddling effect, but less so than Saturnin. The reason for this is the already high tail in the bit weight histogram, due to the absence of bit-level diffusion in the mixing layer.

Finally, Xoodoo has the *lowest* bit huddling of the four primitives studied. This is the consequence of two design choices: having very small S-boxes (3-bit) and the absence of alignment, avoiding the superbox huddling effect altogether.

5.3 Two-round Trail Weight Histograms

We define the trail weight histogram analogous to Definition 17 with the change that $\mathcal{N}_\cdot(k) = \#\{\text{trails } Q : \text{w}_\cdot(Q) = k\}$, where $\cdot$ is either r for differential trails or c for linear trails. Like for the other diagrams, the lower the tail, the lower the number of states with small weights, the better.

Figure 6 reports on the distribution of the weight of two-round differential and linear trails of our four ciphers. To compute the trail weight histograms of the aligned ciphers, we convolved the histograms of the superbox structures (See Equation 1). The distribution of the linear trails for Rijndael is an approximation that was obtained by first taking the integer part of the correlation weights of its S-box to allow for integer arithmetic. The other distributions are exact.

While Rijndael performed the worst with respect to the box weight metric, we see that it performs the best with respect to the trail weights. The reasons are the low maximum DP/LP value of its S-box and its high branch number. However, as seen in Section 4.5, one pays a price in terms of the implementation cost. The relative ranking of the other ciphers does not change in moving from box weight to trail weights. Still, Xoodoo loses some terrain due to its more lightweight S-box layer.

Despite the difference in design approach, Xoodoo and Saturnin have quite similar two-round trail weight histograms. It is therefore interesting how the trail weight histograms compare for three and four rounds.

5.4 Three-round Trail Weight Histograms

We have computed the three-round differential and linear trail weight histograms for Saturnin and Xoodoo and give them in Figure 7. We did not do it for Rijndael due to the prohibitively high cost of its round function and neither for Spongent due to its non-competitive bounds for multiple-round trails as reported in [9]. Hence, we focus on Saturnin and Xoodoo as exponents of the aligned and unaligned wide-trail design approaches. Computing the three-round Saturnin trail histograms turned out to be very computationally intensive for higher weights (see Subsection A.3 for more details) and we were forced to stop at weight 36. Still, the diagrams show the big difference in histograms between Saturnin and Xoodoo.

Despite the fact that the box branch number of Xoodoo is 4 and that of Saturnin is 5, we see that for three-round trails, Xoodoo performs much

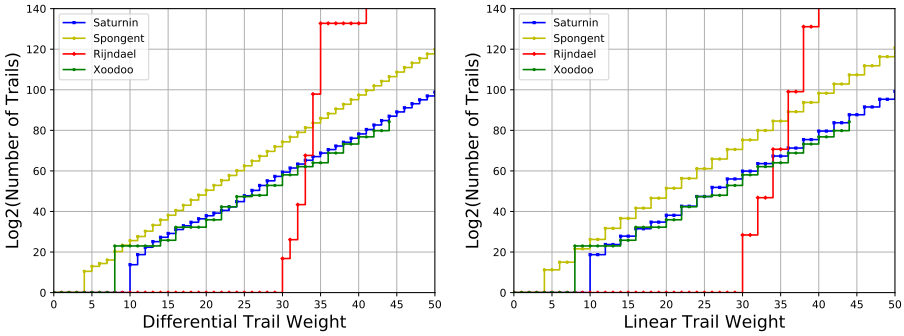

Fig. 6: Two rounds: cumulative differential and linear trail weight histograms.

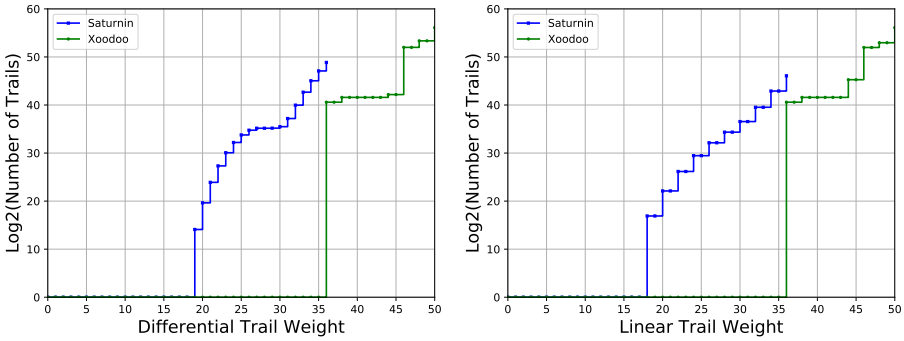

Fig. 7: Three rounds: cumulative differential and linear trail weight histograms.

better than Saturnin. In particular, Xoodoo has no trails with weight below 36, whereas Saturnin has about 2^{43} linear trails with weight below 36, starting from weight 18. Moreover, it has about 2^{47} differential trails with weight below 36, starting from weight 19. This confirms the idea that branch number alone does not paint the whole picture and that these histograms prove to be very useful in comparing the different design approaches.

5.5 Four Rounds and Beyond

We did not conduct experiments for four or more rounds, but can make use of available information. According to [15], there exist no differential or linear trails over four rounds of Xoodoo with weight below 74. In contrast, Saturnin has roughly 2^{82} four-round differential trails with 25 active S-boxes and it has more than $2^{94.5}$ such linear trails. See Section C for a derivation of this estimate. Since each S-box has a weight of 2 or 3, this implies many four-round differential trails with weights in the range [50, 75]. The linear trails have weights in the range [50, 100] due to the fact that active S-boxes have weight 2 or 4. Naturally, in both cases there are also trails with $26, 27, \ldots$ active S-boxes and their number

grows quickly with the box weight due to the additional degrees of freedom in building them. It follows that the trend we see in three-round trails persists for four-round trails: unaligned Xoodoo has a significantly lower tail than aligned Saturnin, despite its lighter round function and lower branch number.

For trails over five rounds and more we report on the known lower bounds on weight in Table 6 in Section D of the supplementary material. We see that up to 6 rounds Xoodoo remains ahead of Saturnin. For higher weights the trail scan programs in Xoodoo reach their computational limit and Saturnin overtakes Xoodoo. Advances in trail scanning are likely to improve the bounds for Xoodoo while for Saturnin the currently known bounds are much more tight. For the whole range Rijndael is well ahead and Spongent is invisible with its weight of 28 for 6 rounds.

6 Clustering

In this section, we investigate clustering of differential trails and of linear trails. The occurrence of such clustering in two-round differentials and linear approximations requires certain conditions to be satisfied. In particular, we define an equivalence relation of states with respect to a linear layer and an S-box partition that partitions the state space in candidate two-round trail cores and the size of its equivalence classes upper bounds the amount of possible trail clustering. This is the so-called cluster partition. We present the partitions of our four ciphers by means of their cluster histograms. For all four ciphers, we report on two-round trail clustering and for Xoodoo in particular we look at the three-round case. With its unaligned structure, we found little clustering in Xoodoo. However, the effects of clustering are apparent in the aligned primitives Rijndael, Saturnin, and Spongent, with them being most noticeable in Rijndael.

6.1 The Cluster Histogram

To define the cluster histogram we need to define two equivalence classes.

Definition 18. *Two states are box-activity equivalent if they have the same activity pattern with respect to a box partition Π:*

$$a \sim a' \text{ if and only if } r_\Pi(a) = r_\Pi(a') \,.$$

We denote the set of states that are box-activity equivalent with a by $[a]_\sim$ and call it the box-activity class *of a.*

Box-activity equivalence has an application in the relation between trail cores and differentials and linear approximations.

Lemma 1. *Two trail cores $(a_0, b_0 \ldots, a_{r-2}, b_{r-2})$ and $(a_0^*, b_0^* \ldots a_{r-2}^*, b_{r-2}^*)$ over a function $f = \mathrm{N}_{r-1} \circ \mathrm{L}_{r-2} \circ \mathrm{N}_{r-2} \circ \cdots \circ \mathrm{L}_0 \circ \mathrm{N}_0$ that are in the same differential (or linear approximation) satisfy $a_0 \sim a_0^*$ and $b_{r-2} \sim b_{r-2}^*$.*

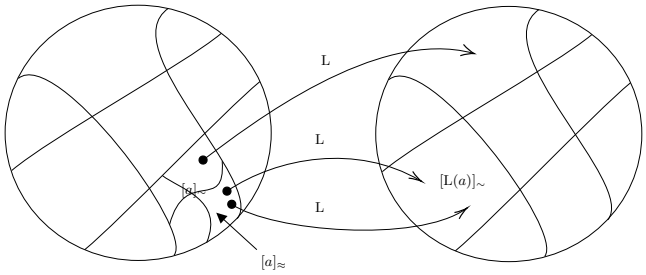

Fig. 8: Partitions of $\mathbb{F}_2^b$ defined by $\sim$ and $\approx$.

Proof. Let $(\Delta_{\text{in}}, \Delta_{\text{out}})$ be the differential over f that the trail cores are in. Since N_0 and N_{r-2} preserve activity patterns, we have that $\Delta_{\text{in}} \sim a_0$, and $\Delta_{\text{in}} \sim a_0^*$, and $\Delta_{\text{out}} \sim b_{r-2}$, and $\Delta_{\text{out}} \sim b_{r-2}^*$. From the symmetry and transitivity of $\sim$ it follows that $a_0 \sim a_0^*$ and $b_{r-2} \sim b_{r-2}^*$. □

Considering the case $r = 2$ in Lemma 1 immediately gives rise to a refinement of box-activity equivalence.

Definition 19. *Two states are cluster-equivalent with respect to a linear mapping* $\mathrm{L} : \mathbb{F}_2^b \to \mathbb{F}_2^b$ *and a box partition* Π *if they are box-activity equivalent before* L *and after it (See Figure 8):*

$$a \approx a' \text{ if and only if } a \sim a' \text{ and } \mathrm{L}(a) \sim \mathrm{L}(a') \,.$$

We denote the set of states that are cluster-equivalent with a *by* $[a]_\approx$ *and call it the* cluster class *of* a*. The partition of* $\mathbb{F}_2^b$ *according to these cluster classes is called the* cluster partition*.*

Corollary 1. *If two two-round trail cores* $(a, \mathrm{L}(a))$ *and* $(a^*, \mathrm{L}(a^*))$ *over* $f = \mathrm{N} \circ \mathrm{L} \circ \mathrm{N}$ *are in the same differential, then* $a \approx a^*$*.*

Proof. If we apply Lemma 1 to the case $r = 2$, we have $a \sim a^*$ and $\mathrm{L}(a) \sim \mathrm{L}(a^*)$. It follows that $a \approx a^*$. □

Corollary 1 shows that the defining differences of any two-round trail cores that cluster together are in the same cluster class. It follows that if these cluster classes are small, then there is little clustering.

For all $a' \in [a]_\approx$ the box weight $\mathrm{w}_\Pi(a') + \mathrm{w}_\Pi(\mathrm{L}(a'))$ is the same. We denote this weight by $\widetilde{\mathrm{w}}([a]_\approx)$.

Definition 20. *Let* $\mathrm{L} : \mathbb{F}_2^b \to \mathbb{F}_2^b$ *be a linear transformation. Let* $\approx$ *be the equivalence relation given in Definition 19. The* cluster histogram $\mathrm{N}_{\Pi,\mathrm{L}} : \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ *of* L *with respect to the box partition* Π *is given by*

$$\mathrm{N}_{\Pi,\mathrm{L}}(k, c) = \#\{[a]_\approx \in \mathbb{F}_2^b/\!\approx : \widetilde{\mathrm{w}}([a]_\approx) = k \wedge \#[a]_\approx = c\} \,.$$

For a fixed box weight, the cluster histogram shows the distribution of the sizes of the cluster classes with that box weight. Ideally, for small box weights, the cluster classes are all very small. Large cluster classes of small weight may lead to two-round trails with a large DP or LP.

Table 3: The cluster histograms of RIJNDAEL and SATURNIN.

$\widetilde{\mathrm{w}}$	$N \times C_{m,n}$ RIJNDAEL superbox $m=8,n=4$	SATURNIN superbox $m=4,n=4$	SATURNIN hyperbox $m=16,n=4$
5	(56×255)	(56×15)	(56×65535)
6	$(28 \times 64005))$	(28×165)	(28×4294574085)
7	(8×16323825)	(8×2625)	$(8 \times 281444913315825)$
8	(1×4162570275)	(1×39075)	$(1 \times 18444492394151280675)$

Table 4: The cluster histogram of `SpongentMix` of SPONGENT.

$\widetilde{\mathrm{w}}$	$N \times C$
2	(16×1)
3	(48×1)
4	$(32 \times 1)\,(36 \times 7)$
5	$(8 \times 1)\,(48 \times 25)$
6	$(12 \times 79)\,(16 \times 265)$
7	(8×2161)
8	(1×41503)

Table 5: Partial cluster histogram (up to translation equivalence) of XOODOO.

$\widetilde{\mathrm{w}}$	$N \times C$
4	(3×1)
7	(24×1)
8	(600×1)
9	(2×1)
10	(442×1)
11	(10062×1)
12	(80218×1)
13	(11676×1)
14	$(228531 \times 1)\,(3 \times 2)$
15	$(2107864 \times 1)\,(90 \times 2)$
16	$(8447176 \times 1)\,(702 \times 2)$
⋮	⋮

6.2 The Cluster Histograms of Our Ciphers

Next, we present the cluster histograms of the superboxes of RIJNDAEL, SATURNIN, and SPONGENT and of the SATURNIN hyperbox. Moreover, we present a partial cluster histogram of XOODOO. The results for RIJNDAEL and SATURNIN are found in Table 3, for SPONGENT in Table 4, and for XOODOO in Table 5. In these tables, C denotes the cardinality of a cluster class and N denotes the number of cluster classes with that cardinality. For instance, an expression such as $(32 \times 1)\,(36 \times 7)$ means that there are 32 cluster classes of cardinality 1 and 36 classes of cardinality 7. Looking at $\widetilde{\mathrm{w}} = 8$ across the three tables, we see that RIJNDAEL, SATURNIN, and SPONGENT have only a single cluster class containing all the states with $\mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{L}(a)) = 8$. In contrast, for XOODOO, each state a sits in its own cluster class. This means that $\mathrm{L}(a)$ is in a different box activity class than $\mathrm{L}(b)$ for any $b \in [a]_{\sim}$ and $b \neq a$.

Thanks to the fact that the mixing layers of RIJNDAEL and SATURNIN have the MDS property, the entries of their cluster histograms are combinatorial expressions of m, the box size, and n, the number of boxes. We describe these methods in detail in Subsection A.2 of the supplementary material.

Table 4 gives the cluster histogram of SPONGENT's superbox. For weights above 4 we see large cluster equivalence classes.

Now, consider the cluster histogram of Xoodoo in Table 5. We see that up to and including box weight 13, we have $\#[a]_{\approx} = 1$. For box weight 14, 15, and 16, we see that $\#[a]_{\approx} \leq 2$. Due to its unaligned structure, it is less likely that equal activity patterns are propagated to equal activity patterns. Therefore, many cluster classes contain only a single state.

6.3 Two-round Trail Clustering

Two-round trail clustering in the keyed Rijndael superbox was investigated in [19]. In that paper the *expected* DP values of trails and differentials are studied, where expected means averaged over all keys. We see considerable clustering in differentials with 5 active S-boxes. For these, the maximum expected DP of differentials is more than a factor 3 higher than the maximum expected DP of 2-round trails, with differentials containing up to 75 trails. For more active S-boxes the number of trails per differential is much higher and hence clustering is worse, but their individual contributions to the expected DP are much smaller and all differentials have expected DP very close to 2^{-32}. For fixed keys or in an unkeyed superbox these differentials and trails have a DP that is a multiple of 2^{-31}. For trails this effect was studied in [21].

In this section we report on our experiments on the other three of our ciphers where we compare two-round differentials with differential trails and linear approximations with linear trails. Figure 9 shows the number of differentials and differential trails up to a given weight of the Saturnin and the Spongent superboxes. In both cases, we see that for low weight the histograms are close and as the weight grows, these histograms diverge. For Saturnin there are roughly 50 times more differentials with weight 15 or less than differential trails with weight 15 or less. For Spongent this ratio is roughly 20. This divergence is due to two reasons: clustering and what we call *clipping*. Due to the large number of differential trails and the limited width of the superbox, the trails cluster. This effect is especially strong for trails with almost all S-boxes active and would give rise to many differentials with DP close to 2^{-16} as the superbox has width 16. What we observe is a majority of differentials with DP equal to 2^{-15}. This is the result of the fact that any differential over a superbox has an even number of ordered pairs and hence the minimum DP is 2^{-15}, yielding weight 15. We call this effect clipping: the weight of differentials cannot be strictly greater than 15. A trail over a k-bit superbox with weight $\mathrm{w} > k - 1$ cannot have a DP $= 2^{-\mathrm{w}}$ as this would imply a fractional number of pairs. This effect has been studied in AES and we refer to Section 7 for a discussion.

Figure 10 shows the weight histograms for two-round differentials and linear approximations. The full-state correlation weight histogram of Saturnin was obtained from that of any of its columns by first rounding the correlation weights to the nearest integer to make integer arithmetic possible. The full-state correlation weight histogram of Spongent was obtained in a similar manner. The remainder of the histograms is exact. Table 5 shows that in Xoodoo almost all differentials contain only a single trail. This means that the clustering is negligible. Therefore, there is no difference between Figures 6 and 10 for Xoodoo.

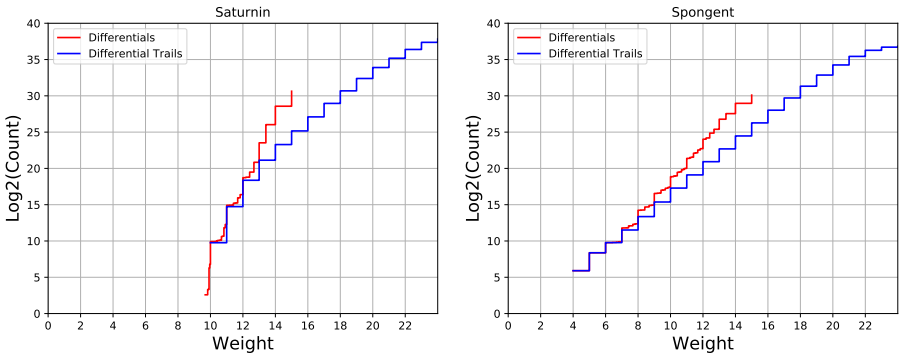

Fig. 9: Differentials and differential trails in the superboxes of SATURNIN and SPONGENT.

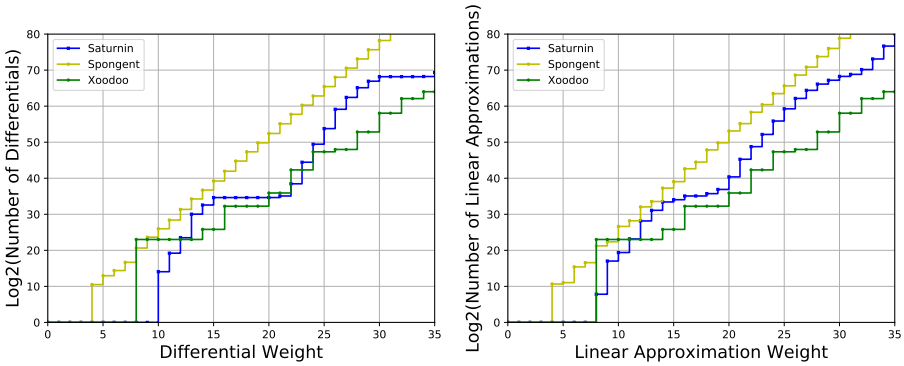

Fig. 10: Two rounds: cumulative restriction and correlation weight histograms.

For SATURNIN the clustering is the most striking. For linear trails we observe a similar effect. For SPONGENT the clustering is less outspoken due to the fact that the trail weight histogram is quite bad to start with.

The effect of clustering in four-round (or two super-round) SATURNIN is interesting. Four-round SATURNIN consists of the parallel application of four 64-bit hyperboxes. The consequence is that for a fixed key, the roughly $2^{127} \cdot 4$ differentials that are active in a single hyperbox and have non-zero DP, all have weight below 63. When computing expected DP values averaging the DP over all round keys, this is closer to 64.

The cluster classes also determine the applicability of the very powerful *truncated differential attacks* [24]. These attacks exploit sets of differentials that share the same box activity pattern in their input difference and the same box activity pattern in their output difference. Despite the fact that the individual trails in these truncated differentials may have very low DP, the joint probability can be significant due to the massive numbers. For two-round differentials the cluster classes are exactly the trail cores in a given truncated differential. In Table 3 we

see that the cluster classes for the Rijndael superbox and Saturnin hyperbox are very large. This clustering leads to powerful distinguishers for e.g., 4-round AES and 8-round Saturnin. The latter can be modeled as 4 hyperboxes followed by an MDS mixing layer followed by 4 hyperboxes and an input difference with a single active hyperbox will have 4 active hyperboxes after 8 rounds, with probability 1. In contrast, if the cluster classes are small, as in the case of the unaligned Xoodoo permutation, it is very unlikely that truncated differential attacks would have an advantage over ordinary differential attacks.

6.4 Three-round Trail Clustering in Xoodoo

Recall that for Xoodoo, no 4-round trails exist with weight below 74 and Table 5 showed that trail clustering in two-round differentials in Xoodoo is negligible, as expected because of its unaligned design. We investigate the conjecture that it is also the case for three rounds.

First, we present a generic technique to find all trails that have an enveloping differential compatible with a given three-round trail core. We apply the technique to Xoodoo, for which it is very efficient.

Given the trail core $(a_1^*, b_1^*, a_2^*, b_2^*)$, Lemma 1 shows that we can restrict ourselves to those (a_1, b_1, a_2, b_2) with $a_1 \sim a_1^*$ and $b_2 \sim b_2^*$. The difference a_1^* defines a vector space A' of all the states in which a box is passive whenever it is passive in a_1^*. If $a_1 \in [a_1^*]_\sim$, then $a_1 \in A'$. Similarly, b_2^* defines a vector space B'. If $b_2 \in [b_2^*]_\sim$, then $b_2 \in B'$. The vector space $B = \mathrm{L}(A')$ contains the candidate values for b_1. Similarly, the vector space $A = \mathrm{L}^{-1}(B')$ contains candidate values for a_2. Because it preserves activity patterns, N restricts the set of candidate values to those satisfying $b_1 \sim a_2$. Hence, we can limit the search to those $x \in B$ and $y \in A$ with $x \sim y$.

To find all valid trails of the form $(\Delta_{\mathrm{in}}, a_1, b_1, a_2, b_2, \Delta_{\mathrm{out}})$, we first reduce the size of the space of all trail cores (a_1, b_1, a_2, b_2) using a necessary condition. When this space is small enough, we exhaustively search for a valid trail.

We write $\overline{B}$ for a basis of B and $\overline{A}$ for a basis of A. To reduce the dimension of the spaces, we will apply an algorithm directly on their bases. First, we need the notion of *isolated active bit*.

Definition 21. *A bit i of $b \in \overline{B}$ is said to be an isolated active bit if $b_i = 1$ and $b'_i = 0$ for all $b' \in \overline{B} \setminus \{b\}$.*

A basis vector having an isolated active bit determines the box activity of any linear combination that includes it.

Lemma 2. *If $b \in \overline{B}$ has an isolated active bit in position i, then any vector in the affine space $b + \mathrm{span}(\overline{B} \setminus \{b\})$ has the corresponding box activated.*

Proof. If b has an isolated active bit in position i, then the ith bit of any vector in $b + \mathrm{span}(\overline{B} \setminus \{b\})$ is active. As a result, the box containing this bit is active. □

Similar to how an isolated active bit always activates the corresponding box, a box is never activated if no basis vector activates it.

Lemma 3. *If the ith box is passive in every vector of $\overline{A}$, then the ith box is passive in all vectors of A. We say that box i is passive in $\overline{A}$.*

We define a condition that makes it possible to remove a basis vector from the basis without excluding potentially valid trails.

Condition 1 (reduction condition) *We say that a basis vector $b \in \overline{B}$ satisfies the* reduction condition *if and only if it has an isolated active bit in a box that is passive in $\overline{A}$. The same is true when swapping the role of $\overline{B}$ and $\overline{A}$.*

The following lemma shows that the reduction condition is sufficient to reduce the dimension of the vector space we consider.

Lemma 4. *If a basis vector $b \in \overline{B}$ satisfies Condition 1, then all valid differences before the* N *in the middle are in span$(\overline{B} \setminus \{b\})$. The same is true when swapping the role of $\overline{B}$ and $\overline{A}$.*

Proof. As a consequence of Lemma 2 and Lemma 3, a valid difference before the nonlinear layer cannot be constructed from $b^{(i)}$ because it would contradict the fact that the activity pattern is preserved through the nonlinear layer. □

The algorithm now consists in repeatedly removing basis vectors from $\overline{B}$ and $\overline{A}$ that satisfy Condition 1 until this is no longer possible. This can be done efficiently by searching for pivots for a Gaussian elimination among indices of vectors from $\overline{A'}$ (respectively $\overline{B'}$) that correspond to never activated boxes in $\overline{B'}$ (respectively $\overline{A'}$). Indeed, these pivots can be used to row-reduce the corresponding basis along them, thus revealing an isolated active bit.

If the algorithm sufficiently decreased the dimensions, then we can exhaustively test all pairs $(b_1, a_2) \in B \times A$ (after reduction) according to the following criteria:

- (b_1, a_2) is a valid differential over N;
- There exists a Δ_{in} such that both $(\Delta_{\text{in}}, a_1^*)$ and $(\Delta_{\text{in}}, a_1)$ are valid differentials over N;
- There exists a Δ_{out} such that both $(b_2^*, \Delta_{\text{out}})$ and $(b_2, \Delta_{\text{out}})$ are valid differentials over N.

Applying our method to all three-round trail cores of Xoodoo up to weight 50 [17] shows that there exists no cluster for all these trails.

7 Dependence of Round Differentials

In this section we study the dependence of round differentials in the sense of Definition 5 in Section 2.1. It has been found in [21] that the vast majority of trails over the Rijndael superbox have dependent round differentials. We will investigate this for differential trails over three-round Xoodoo. We expect that the dependence effects observed in Rijndael disappear in an unaligned cipher. Hence, we now investigate this for differential trails over three-round Xoodoo.

7.1 Masks for differentials over nonlinear components

We note $V_{\mathrm{N}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ the set of output states that follow the differential $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ over N, i.e. $V_N(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) = \mathrm{N}(U_{\mathrm{N}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}))$. From [21], we have that $U_{\mathrm{N}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ and $V_{\mathrm{N}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ are affine if $\#U_{\mathrm{S}_i}(P_i(\Delta_{\mathrm{in}}), P_i(\Delta_{\mathrm{out}})) \leq 4$ for each S-box. Since this assumption holds for our four ciphers, both $U_{\mathrm{N}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ and $V_{\mathrm{N}}(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ are affine and can be described by a system of affine equations on the bits of the state x. Each affine equation can be written as $u^\top x + c$ with u a b-bit vector called mask and c a bit.

Given a three-round differential trail $Q = (\Delta_{\mathrm{in}}, a_1, b_1, a_2, b_2, \Delta_{\mathrm{out}})$, one can define four sets of masks:

- A_1, the masks that come from $V_{\mathrm{N}}(\Delta_{\mathrm{in}}, a_1)$;
- B_1, the masks that come from $U_{\mathrm{N}}(b_1, a_2)$;
- A_2, the masks that come from $V_{\mathrm{N}}(b_1, a_2)$;
- B_2, the masks that come from $U_{\mathrm{N}}(b_2, \Delta_{\mathrm{out}})$.

These masks are said to be all *independent* if

$$\#U_{\mathrm{N}\circ\mathrm{L}\circ\mathrm{N}\circ\mathrm{L}\circ\mathrm{N}}(Q) = 2^{b-(\#A_1+\#B_1+\#B_2)} = 2^{b-(\#A_1+\#A_2+\#B_2)}\,.$$

which is, per Definition 5, equivalent to the independence of round differentials.

We first present an efficient generic method for determining whether three-round trail masks are independent. Then we apply this method to Xoodoo. Since L is linear, A_1 can be linearly propagated through it to obtain a set of masks A'_1 at the input of the second nonlinear layer. Similarly, we can propagate B_2 through the inverse linear layer to obtain a set of masks B'_2 at the output of the second nonlinear layer.

7.2 Independence of masks over a nonlinear layer

B_1 and A'_1 form sets of masks at the input of the second nonlinear layer. If the rank of $C_1 = B_1 \cup A'_1$ is the sum of the ranks of B_1 and A'_1, then C_1 contains independent masks. The same strategy can be used to test for dependence of masks in $C_2 = A_2 \cup B'_2$.

As for the independence of masks of the complete trail, we need to check for dependence between C_1 and B'_2 or between A'_1 and C_2. We will apply an algorithm similar to the one we used in Section 6.4 to reduce bases. However, here we use it to reduce the cardinalities of the mask sets.

The following lemma makes this possible.

Lemma 5. *Let C_1 and B'_2 be two sets of masks before and after an S-box layer. If a mask u in C_1 satisfies Condition 1, then the number of states that satisfy the equations associated with the masks in both $C_1 \setminus \{u\}$ and B'_2 is exactly two times the number of states before removing u. The same is true by swapping the role of C_1 and B'_2.*

Proof. Since u satisfies Condition 1, let i be the index of the isolated bit, j be the index of the corresponding S-Box and k the number of masks in B_2'. No mask in B_2' is putting a constraint on any of the m bits of the jth S-Box, thus the 2^{b-k} solutions can be seen as 2^{b-k-m} groups of 2^m different states that only differ in the m bits of the jth S-box. Since the S-box is invertible, the application of the inverse of the nonlinear layer to a whole group of 2^m vectors results in a group of 2^m different states that, again, only differ on the value of the jth S-box.

We can further divide those 2^{b-k-m} groups each into 2^{m-1} subgroups of 2 different states that only differ in the value of the ith bit. By definition on an isolated bit, either both or none of the two states inside a subgroup satisfy all equations associated with the masks in $C_1 \setminus \{u\}$. Finally, inside a subgroup exactly one of the two states will satisfy the equation associated with mask u. Thus, the number of solutions by removing u is multiplied by exactly two. □

We first check for linear dependence inside C_1 by computing its associated rank. Then, we recursively check if some mask in either C_1 or B_2' satisfies Condition 1 and if it is the case we remove them from the sets of masks.

There are three possible outcomes when applying this process to a three-round differential trail:

- If C_1 is not full rank, we can conclude that masks in B_1 and A_1' are dependent;
- Else, if either set is empty, Lemma 5 applied at each step guarantees us that the number of states satisfying the equations associated with the masks in both C_1 and B_2' is equal to $2^{b-(\#C_1+\#B_2')}$, that is to say the masks are independent;
- If none of the two conditions above are met, we cannot directly conclude about (in)dependence between remaining masks but we can apply the same method to A_1 and C_2 and hope for a better outcome.

7.3 Application to Xoodoo

This process is used to check for independence in differential trails over three rounds of Xoodoo. It has been applied to the same differential trails as processed in Section 6.4. In all cases, the masks, and thus round differentials, were found to be independent. This was not obtained by sampling, but instead by counting the number of solutions, hence this independence is exact in the sense of Definition 5. As a result, the DP of each such trail is the product of the DP values of its round differentials, which implies that $\mathrm{DP}(Q) = 2^{-\mathrm{w_r}(Q)}$.

8 Conclusion

We put forward alignment as a crucial property that characterizes the interactions between linear and nonlinear layers w.r.t. the differential and linear propagation properties. We conducted experiments on four S-box based primitives that otherwise represent different design approaches. We precisely defined what

it means for a primitive to be aligned and showed that Rijndael, Saturnin, and Spongent are aligned, whereas Xoodoo is unaligned. Through these examples, we highlighted and analyzed different effects of alignment on the propagation properties.

Acknowledgements. We thank Bart Mennink for helpful comments. Moreover, we would like to thank the anonymous reviewers of an earlier version of this paper for their useful feedback. Joan Daemen and Daniël Kuijsters are supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA. This work is partially supported by the French National Research Agency in the framework of the *Investissements d'avenir* programme (ANR-15-IDEX-02).

References

1. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: CHES 2017
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch. 2013, 404 (2013)
3. Bernstein, D.J.: The Salsa20 family of stream ciphers. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists
4. Bernstein, D.J.: Cache-timing attacks on AES. Tech. rep. (2005)
5. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Extended Keccak code package. https://github.com/XKCP/XKCP
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference (January 2011)
7. Beyne, T., Chen, Y.L., Dobraunig, C., Mennink, B.: Dumbo, jumbo, and delirium: Parallel authenticated encryption for the lightweight circus. IACR Trans. Symmetric Cryptol. 2020(S1), 5–30 (2020), https://doi.org/10.13154/tosc.v2020.iS1.5-30
8. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: CRYPTO '90. https://doi.org/10.1007/3-540-38424-3_1
9. Bogdanov, A., Knezevic, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: SPONGENT: the design space of lightweight cryptographic hashing. IACR Cryptol. ePrint Arch. 2011, 697 (2011)
10. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: CHES 2007
11. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: SEA 2010
12. Canteaut, A., Duval, S., Leurent, G., Naya-Plasencia, M., Perrin, L., Pornin, T., Schrottenloher, A.: Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. IACR ToSC 2020(S1)
13. Canteaut, A., Duval, S., Leurent, G., Naya-Plasencia, M., Perrin, L., Pornin, T., Schrottenloher, A.: Saturnin implementations. https://project.inria.fr/saturnin/files/2019/05/saturnin.zip
14. Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995)

15. Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Xoodyak, a lightweight cryptographic scheme. IACR ToSC 2020(S1)
16. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: The design of Xoodoo and Xoofff. IACR Trans. Symmetric Cryptol. 2018(4), 1–38 (2018)
17. Daemen, J., Hoffert, S., Van Assche, G., Van Keer, R.: XooTools. https://github.com/KeccakTeam/Xoodoo/tree/master/XooTools (2018)
18. Daemen, J., Peeters, M., Van Assche, G., Bertoni, G.: On alignment in Keccak. Note (2011)
19. Daemen, J., Rijmen, V.: Understanding two-round differentials in AES. In: SCN 2006
20. Daemen, J., Rijmen, V.: The wide trail design strategy. In: Honary, B. (ed.) Cryptography and Coding, Proceedings (2001)
21. Daemen, J., Rijmen, V.: Plateau characteristics. IET Information Security 1(1), 11–17 (2007)
22. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020)
23. Huffman, W.C., Pless, V.: Fundamentals of Error-Correcting Codes. Cambridge University Press (2003)
24. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. Lecture Notes in Computer Science, vol. 1008, pp. 196–211. Springer (1994), https://doi.org/10.1007/3-540-60590-8_16
25. Kranz, T., Leander, G., Stoffelen, K., Wiemer, F.: Shorter linear straight-line programs for MDS matrices. IACR Trans. Symmetric Cryptol. 2017(4), 188–211 (2017)
26. Künzer, Martin, Tentler, Wahrheit: Zassenhaus-algorithmus. https://mo.mathematik.uni-stuttgart.de/inhalt/beispiel/beispiel1105/
27. Leander, G., Poschmann, A.: On the classification of 4 bit s-boxes. In: Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Proceedings
28. Li, C., Wang, Q.: Design of lightweight linear diffusion layers from near-mds matrices. IACR Trans. Symmetric Cryptol. 2017(1), 129–155 (2017)
29. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Proceedings
30. McGeer, P.C., Sanghavi, J.V., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: ESPRESSO-SIGNATURE: a new exact minimizer for logic functions. IEEE Trans. Very Large Scale Integr. Syst. 1(4), 432–440 (1993)
31. Mella, S., Daemen, J., Van Assche, G.: New techniques for trail bounds and application to differential trails in Keccak. IACR ToSC 2017(1)
32. NIST: Federal information processing standard 197, advanced encryption standard (AES) (November 2001)
33. NIST: Federal information processing standard 202, SHA-3 standard: Permutation-based hash and extendable-output functions (August 2015)
34. Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Proceedings
35. Park, S., Sung, S.H., Chee, S., Yoon, E., Lim, J.: On the security of rijndael-like structures against differential and linear cryptanalysis. In: Advances in Cryptology - ASIACRYPT 2002
36. Schwabe, P., Stoffelen, K.: All the AES you need on Cortex-M3 and M4. In: SAC 2016 - 23rd International Conference, Revised Selected Papers
37. Shamsabad, M.R.M., Dehnavi, S.M.: Dynamic MDS diffusion layers with efficient software implementation. Int. J. Appl. Cryptogr. 4(1), 36–44 (2020)
38. Stoffelen, K.: AES implementations. https://github.com/Ko-/aes-armcortexm

Supplementary Material

A Histogram computations

In this section, we describe methods for computing histograms. First, we describe a general method to obtain the histogram of an aligned function from the histograms of its box functions. Second, we describe some methods to obtain the cluster histograms of the ciphers described in Section 4.

Almost all of our computations were done to full precision. The only exception is the case of computing the correlation weights for Rijndael, Saturnin, and Spongent. In this case, we took the integer part of the intermediate results and computed on those numbers.

As a rule of thumb, the most interesting part of any histogram is its left tail, i.e., the part containing the distribution of the low weights. As a consequence of this, we are satisfied if we are able to compute a partial histogram that includes this tail. We see in the case of Rijndael and Xoodoo that this is frequently all we can hope to expect.

A.1 Convolution

Let $\mathrm{L} = \mathrm{L}_0 \times \cdots \times \mathrm{L}_{n-1} : \mathbb{F}_2^b \to \mathbb{F}_2^b$ be a function composed of box functions with respect to an ordered partition Π. Computing a histogram of L exhaustively is often computationally infeasible. However, the histograms of the box functions L_i typically are feasible to compute thanks to the small box width. Given the histograms of the L_i, it is possible to combine them to get the histogram of L itself. This combining operation is a form of convolution and is therefore denoted as $*$. The idea is best illustrated by an example.

Example 1. Consider Rijndael and suppose we wish to compute the convolution of the restriction weight histograms of two of its S-boxes for a fixed output difference. We represent the histograms in two-column notation, where we list the restriction weights in the first column, and for each one its image, i.e., the number of input differences with the given weight, in the second column. To make the representation finite, a necessity for performing computations on these objects, we restrict ourselves to those weights for which the image is non-zero.

$$\begin{bmatrix} 0 & 1 \\ 6 & 1 \\ 7 & 126 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ 6 & 1 \\ 7 & 126 \end{bmatrix} = \begin{bmatrix} 0+0 & 1 \cdot 1 \\ 0+6 & 1 \cdot 1 \\ 0+7 & 1 \cdot 126 \\ 6+0 & 1 \cdot 1 \\ 6+6 & 1 \cdot 1 \\ 6+7 & 1 \cdot 126 \\ 7+0 & 126 \cdot 1 \\ 7+6 & 126 \cdot 1 \\ 7+7 & 126 \cdot 126 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 6 & 2 \\ 7 & 252 \\ 12 & 1 \\ 13 & 252 \\ 14 & 15876 \end{bmatrix}$$

To the end of abstracting the notion showcased in the example, suppose that there exist t associative binary operations $\oplus_j : \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ for $0 \leq j \leq t-1$. For our purposes, these operations are just $+$ (regular addition) or $\cdot$ (regular multiplication) and we restrict ourselves to either $t = 2$ or $t = 3$. Furthermore, suppose that $T^i \subseteq \mathbb{Z}_{\geq 0}^t$ is an encoding of some histogram of L_i for $0 \leq i \leq n-1$. We think of T^i as an $s \times t$ matrix for some $s \in \mathbb{Z}_{\geq 0}$ and index its rows as T^i_j and its entries as T^i_{jk}. We define the binary convolution operator

$$* : \mathbb{Z}_{\geq 0}^t \times \mathbb{Z}_{\geq 0}^t \to \mathbb{Z}_{\geq 0}^t$$

that combines sequences as

$$(l_0, \ldots, l_{t-1}) * (m_0, \ldots, m_{t-1}) = (l_0 \oplus_0 m_0, \ldots, l_{t-1} \oplus_{t-1} m_{t-1})$$

We note that $*$ is associative, since the $\oplus_j$ are. Using this operator, it is possible to combine sequences from different histograms in a sensible way. Any sequence in the histogram of L is related to the sequences in the histograms of the L_i in the following way:

$$(n_0, \ldots, n_{t-1}) = \mathop{\ast}_{\substack{0 \leq i \leq n-1 \\ j \in C_i(n_0, \ldots, n_{t-1})}} T^i_j$$

where

$$C_i(n_0, \ldots, n_{t-1}) = \{j \in \mathbb{Z}_{\geq 0} : \bigoplus_{i=0}^{n-1}{}_l \, T^i_{jl} = n_l \text{ for } 0 \leq l \leq t-1\}$$

A.2 Mixing Layers Based on MDS Codes

We show how to compute the box weight histogram and the cluster histogram of a mixing layer L that is based on MDS codes. In the cipher built on top of L, we suppose that there is an S-box layer N. Let the ordered partition Π_0 be defined by N. Moreover, we suppose that the size of the boxes of Π_0 is m. Consider

$$\mathrm{L} = \mathrm{L}_0 \times \cdots \times \mathrm{L}_{s-1} : \bigtimes_{i=0}^{s-1} \mathbb{F}_2^{km} \to \bigtimes_{i=0}^{s-1} \mathbb{F}_2^{km} .$$

Now, L defines an ordered partition Π_1. Indeed, each of the s boxes of Π_1 is the union of k boxes of Π_0 of size m. It follows that $\Pi_0 \leq \Pi_1$, N is box-aligned with respect to both Π_0 and Π_1, and L is box-aligned with respect to Π_1.

Definition 22. *A function $f : \mathbb{F}_{2^m}^k \to \mathbb{F}_{2^m}^k$ is called an MDS function if the set $\{(x, f(x)) : x \in \mathbb{F}_{2^m}^k\} \subseteq \mathbb{F}_{2^m}^{2k} \cong \mathbb{F}_2^{2km}$ is an MDS code over $\mathbb{F}_{2^m}$ of minimum distance d.*

The minimum distance d is precisely the box branch number. Henceforth, we make the assumption that the box functions L_i, with $0 \leq i \leq s-1$, are MDS functions, i.e., that have branch number $d = k + 1$.

First, Theorem 2 shows how to compute the box weight histogram of L_i with respect to the subset of Π_0 consisting of the boxes indexed by $ik, \ldots, (i+1)k-1$ (they form a partition of the input space of L_i).

Theorem 2. *Let C be an $[2k, k, k+1]$ MDS code over $\mathbb{F}_q$. The weight distribution of C is given by $A_0 = 1$, $A_w = 0$ for $1 \leq w < k+1$, and*

$$A_w = \binom{2k}{w} \sum_{i=0}^{w-k-1} (-1)^i \binom{w}{i} (q^{w-k-i} - 1)$$

for $k+1 \leq w \leq 2k$.

Proof. This is a specific case of Theorem 7.4.1 in [23]. □

As an example, in both `MixColumns` of Rijndael and MC of Saturnin we have $k = 4$. Using convolution, it is now possible to obtain the box weight histogram of L from those of the L_i.

Next, we show how to compute the L_i-box histogram. We put $C_{m,k}(\mathrm{w}) = \#[a]_{\approx}$ for any $a \in \mathbb{F}_2^{km}$ with $\mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{L}(a)) = \mathrm{w}$. In other words, $C_{m,k}(\mathrm{w})$ does not depend on the box activity pattern of a, but only on its box weight. A box activity pattern can be chosen in $\binom{2n}{k}$ Before stating the main result, we prove some lemmas.

Lemma 6. *We have $C_{m,k}(k+1) = 2^m - 1$.*

Proof. Since L_i is an MDS function, there exists a $k \times k$ matrix M such that $H = (M I_k)$ is a parity-check matrix of an MDS code of dimension k. Let $S \subseteq [0, 2k-1]$ with $\#S = k$ be a subset of the column index space. The columns of H indexed by S form a $k \times k$ sub-matrix. Since H defines an MDS code, this sub-matrix is invertible. This means that the columns of the reduced row echelon form of H indexed by S form the identity matrix I_k. By permuting the columns of the reduced row echelon form, we obtain a parity-check matrix of an equivalent code, $H' = (M' I_k)$. This defines a linear map $\mathrm{M} : \mathbb{F}_{2^m}^k \to \mathbb{F}_{2^m}^k$ given by $\mathrm{M}(a) = M'^{\top} a$. Now, let $a \in \mathbb{F}_2^{km}$ with $\mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{L}(a)) = k+1$. Pick the subset S in such a way that it contains the index of one active box of $(a, \mathrm{L}(a))$ and such that the other indices correspond to $k-1$ passive boxes. The other k active boxes are completely determined by this single active box through M. Hence, we have only $2^m - 1$ degrees of freedom. □

Lemma 7. *We have $C_{m,k}(k+2) = (2^m - 1)(2^m - 1 - k)$.*

Proof. Let $a \in \mathbb{F}_2^{km}$ with $\mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{L}(a)) = k+2$. By the same argument as given in Lemma 6, we pick S in such a way that it contains the indices of two active boxes of a. There are $(2^m - 1)^2$ ways of choosing the vector a such that it is active in two boxes. Then M determines the other k boxes. Clearly $k+1 \leq \mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{M}(a)) \leq k+2$ (as there are only k boxes at the output of M). We subtract the number of vectors that lead to a box weight of $k+1$ According to Lemma 6, this number is $2^m - 1$ for a fixed position of an active box. We can choose this position in k ways. Hence, in total, we need to subtract $k(2^m - 1)$ inputs. The result readily follows. □

Theorem 3 states the main result. The L_i-box histogram follows directly from it.

Theorem 3. *For $k+1 \leq w \leq 2k$, the following recurrence relation holds:*

$$C_{m,k}(\mathrm{w}) = (2^m - 1)^{w-k} - \sum_{1 \leq i \leq w-k-1} \binom{k}{i} C_{m,k}(\mathrm{w} - i)$$

Moreover, $C_{m,k}(0) = 1$.

Proof. Let $a \in \mathbb{F}_2^{km}$ with $\mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{L}(a)) = \mathrm{w}$. By the same argument as given in Lemma 6, pick S such that it contains the indices of $w-k$ active boxes. There are $(2^m-1)^{w-k}$ ways of choosing the vector a such that it is active in $w-k$ boxes. It follows that $k+1 \leq \mathrm{w}_{\Pi}((a, \mathrm{M}(a))) \leq w$. We subtract the number of vectors that lead to a box weight of $k+i$ for $1 \leq i \leq w-k-1$ and obtain the result. □

Again, using convolution, it is possible to obtain the cluster histogram from the L_i-box histograms.

A.3 Exhaustive Search

Cluster Histogram up to Given Box Weight Let $\mathrm{L} : \mathbb{F}_2^b \to \mathbb{F}_2^b$ be a linear transformation. Suppose that we want to determine the cluster histogram, but that it is infeasible to construct the whole histogram because L is not box-aligned nor does it have any other properties that make it easy to do so. In this case, it is still possible to construct the cluster histogram up to a given box weight. To this end, suppose that we have an algorithm similar to the Trail Search described in [31] that generates a list of vectors up to a given weight.

For a given difference $a \in \mathbb{F}_2^b$, consider the vector $(a, \mathrm{L}(a)) \in \mathbb{F}_2^{2b}$ with $\mathrm{w}_{\Pi}(a) + \mathrm{w}_{\Pi}(\mathrm{L}(a)) = \mathrm{w}$. We wish to compute $\#[a]_{\approx}$. Consider the vector space spanned by the basis vectors that have the same box activity pattern as a:

$$V(a) = \left\langle \bigcup_{\substack{0 \leq i \leq n-1 \\ r_{\Pi}(a)_i \neq 0}} \{e^b_{im}, e^b_{im+1}, \ldots, e^b_{(i+1)m-1}\} \right\rangle$$

We compute a basis for $\mathrm{L}(V(a)) \cap V(\mathrm{L}(a))$ using Zassenhaus algorithm [26]. Then

$$c = \#[a]_{\approx} = \#\{v \in \mathrm{L}(V(a)) \cap V(\mathrm{L}(a)) : v \in [\mathrm{L}(a)]_{\sim} \wedge \mathrm{L}^{-1}(v) \in [a]_{\sim}\}$$

and we increment $N_{\Pi,\mathrm{L}}(\mathrm{w}, c)$ by one. Once we have considered all vectors of box weight w, the values of $N_{\Pi,\mathrm{L}}(\mathrm{w}, \cdot)$ are exact.

Three-round Trail Search Saturnin During the first three rounds of Saturnin, all step functions are applied, in parallel, to disjoint slices. Since we are interested in the tail of the differential or linear trail weight histogram, we may limit our search for trails to a single slice. The cipher applies sixteen S-boxes to a slice and we write Π for the corresponding partition. Clearly, an activity pattern

with respect to Π can be encoded as a vector of sixteen bits. We represent this vector as the following 4×4 binary matrix:

$$\begin{bmatrix} r_\Pi(a)_3 & r_\Pi(a)_2 & r_\Pi(a)_1 & r_\Pi(a)_0 \\ r_\Pi(a)_6 & r_\Pi(a)_5 & r_\Pi(a)_4 & r_\Pi(a)_7 \\ r_\Pi(a)_9 & r_\Pi(a)_8 & r_\Pi(a)_{11} & r_\Pi(a)_{10} \\ r_\Pi(a)_{12} & r_\Pi(a)_{15} & r_\Pi(a)_{14} & r_\Pi(a)_{13} \end{bmatrix}$$

In this representation, each matrix row corresponds to the activity pattern of differences or linear masks at the input of a single MC in the first mixing layer. Similarly, a matrix column corresponds to the activity pattern of differences or linear masks at the output of a single MC in the second mixing layer. This allows us to compute candidate activity patterns for which the weight of any differential or linear trail that contains a difference or linear mask contained in that pattern does not exceed an upper bound on the differential or linear trail weight that we set beforehand. Indeed, we generate all possible 4×4 binary matrices, encoding all possible activity patterns. Each matrix row and each matrix column has a bit weight, which corresponds to the box weight of the differences and linear masks at the input or output of a single MC. Since MC defines an MDS code of minimum distance 5, the differences or linear masks associated with a matrix row of bit weight w contribute at least $5 - \mathrm{w}$ to the differential or linear trail weight. A similar argument can be given for the columns. This gives a lower bound on the actual differential or linear trail weight of any trail comprising a difference or linear mask contained in that activity pattern. We determine whether this lower bound is smaller than or equal to the upper bound that we set. The result is a collection of candidate activity patterns, the lower bound of which does not exceed our fixed upper bound. To the end of determining the actual trail weights, we first switch back to the following sequential representation as this makes it easier to apply the step functions:

$$\begin{bmatrix} r_\Pi(a)_0 & r_\Pi(a)_1 & r_\Pi(a)_2 & r_\Pi(a)_3 \\ r_\Pi(a)_4 & r_\Pi(a)_5 & r_\Pi(a)_6 & r_\Pi(a)_7 \\ r_\Pi(a)_8 & r_\Pi(a)_9 & r_\Pi(a)_{10} & r_\Pi(a)_{11} \\ r_\Pi(a)_{12} & r_\Pi(a)_{13} & r_\Pi(a)_{14} & r_\Pi(a)_{15} \end{bmatrix}$$

Using convolution and exhaustive search within the candidate activity patterns, we are able to find all the differential and linear trails within that slice up to a given weight.

B Minimal Sum-of-product Forms

We have used the Espresso algorithm to get a minimal sum-of-products (SOP) form of the three ciphers below. Note that addition denotes 'or', multiplication denotes 'and', and an overline denotes negation.

Xoodoo:

$$
\begin{aligned}
Y_0 &= X_0X_1 + X_0\overline{X_1X_2} + \overline{X_0X_1}X_2 \\
Y_1 &= X_1X_2 + X_0\overline{X_1X_2} + \overline{X_0}X_1\overline{X_2} \\
Y_2 &= X_0X_2 + \overline{X_0X_1}X_2 + \overline{X_0}X_1\overline{X_2}
\end{aligned}
$$

Saturnin:

$$
\begin{aligned}
Y_0 &= X_0\overline{X_1X_2X_3} + X_0\overline{X_2}X_3 + \overline{X_0}X_1\overline{X_2X_3} + X_1X_2X_3 + \overline{X_1}X_2\overline{X_3} \\
Y_1 &= X_0\overline{X_1}X_2 + \overline{X_0X_1X_2}X_3 + \overline{X_1}X_2\overline{X_3} + \overline{X_0}X_1 \\
Y_2 &= \overline{X_0X_1X_2}X_3 + \overline{X_0}X_2\overline{X_3} + \overline{X_0}X_1\overline{X_2X_3} + X_0X_1 \\
Y_3 &= X_0\overline{X_1X_2X_3} + \overline{X_0}X_1\overline{X_2X_3} + X_2X_3 + X_1X_2
\end{aligned}
$$

Spongent:

$$
\begin{aligned}
Y_0 &= X_0X_1\overline{X_2}X_3 + X_0\overline{X_1X_2}X_3 + X_0X_1\overline{X_2X_3} + \overline{X_0}X_1X_2X_3 + \overline{X_1}X_2\overline{X_3} + \overline{X_0X_1X_2} \\
Y_1 &= X_0X_1\overline{X_2}X_3 + X_0\overline{X_1}X_2X_3 + \overline{X_0}X_1X_2 + X_0\overline{X_1X_2X_3} + \overline{X_0X_1X_2} + X_1X_2X_3 \\
Y_2 &= X_0\overline{X_1X_2}X_3 + \overline{X_0X_1}X_2\overline{X_3} + X_0X_1X_2\overline{X_3} + \overline{X_0X_2X_3} + X_1X_2X_3 + X_0\overline{X_1X_2X_3} \\
Y_3 &= X_0X_1\overline{X_2X_3} + \overline{X_0}X_1X_2X_3 + X_0\overline{X_1}X_2X_3 + \overline{X_0X_1}X_2\overline{X_3} + X_0X_1X_2\overline{X_3} + \overline{X_0X_2}X_3 \\
&\quad + X_0\overline{X_1X_2X_3}
\end{aligned}
$$

From De Morgan's laws, it follows that $XY + ZW = \overline{\overline{XY} \cdot \overline{ZW}}$. In other words, the circuits of depth two that can be derived from the SOPs above, consisting of a layer of (possibly multi-input) and gates, followed by a layer of (possibly multi-input) or gates, can be converted into a circuit of depth two in which each layer consists of (possibly multi-input) nand gates.

C Estimating the number of trails with 25 active S-boxes in 4-round Saturnin

A trail over 4 rounds of Saturnin is a trail in a hyperbox that has superboxes as S-boxes. This trail is active in 5 superboxes and has 5 active boxes in each superbox.

There are $\binom{8}{5} = 56$ ways to select the 5 active superboxes from the 8 superboxes.

In the central mixing layer there can be $x \in \{1, 2, 3, 4\}$ MC instances active.

In each active MC, we have one degree of freedom as the choice of a single difference among the 5 active ones fixes the four others. This gives 15^x choices for the middle differences.

Each active superbox now has x active boxes at its input (or output), and shall have $5-x$ active boxes at its output (or input). Consider the case $x = 1$. For

Table 6: Known lower bounds for weights of differential trails.

cipher	source	number of rounds								
		1	2	3	4	5	6	7	8	12
SPONGENT	[9]	2	4	8	12	≥ 20	28	-	-	≥ 72
AES	[22]	6	30	54	150	≥ 156	≥ 180	≥ 204	≥ 300	≥ 450
SATURNIN	[12]	2	10	19	≥ 50	≥ 58	≥ 90	≥ 122	250	≥ 300
XOODOO	[15]	2	8	36	$[74, 80]$	≥ 94	≥ 104	≥ 110	≥ 148	≥ 222

a given choice of the middle difference, the difference at the input (or output) of a single S-box is fixed. For differential trails, the number of compatible output differences depends on the concrete output difference but ranges from 6 to 8 with an average of exactly 7. We think it is reasonable in this estimation to approximate this by exactly 7. For linear trails, the number of input masks compatible with a given output mask is always 10.

So given a choice of the intermediate difference, there are 7^5 differential trail cores and 10^5 linear trail cores.

This gives in total $56 \times 15 \times 7^5$ differential trail cores and $56 \times 15 \times 10^5$ Each of these trail cores has in total 20 active S-boxes in the outer S-box layers. Every difference at the inside of such an active S-box has 7 compatible differences at the outside. It follows that there are in total $56 \times 15 \times 7^5 \times 7^{20} \approx 2^{80}$ such differential trails per hyperbox and as there are 4 hyperboxes, this totals to 2^{82}.

Every mask at the inside of such an active S-box has 10 compatible masks at the outside. It follows that there are in total $56 \times 15 \times 10^5 \times 10^{20} \leq 2^{92.5}$ such linear trails per hyperbox and as there are 4 hyperboxes, this totals to $2^{94.5}$.

These are only the trails with a single active MC in the middle mixing layer. We do not count the other classes as their analysis is more involved and the final total number is much less. Hence this class dominates the total number.

D Known trail bounds for up to 12 rounds

In Table 6 we list the trail bounds for our four ciphers for up to 12 rounds. For AES the numbers are based on the existence of periodic trails with period 4 where the profile of the number of active S-boxes is $(1, 4, 16, 4)$ and the fact that the minimum weight for the AES S-box is 6. For SATURNIN the numbers are based on the existence of periodic trails with period 8 where the profile of the number of active S-boxes is $(1, 4, 16, 4, 16, 64, 16, 4)$ and the fact that the minimum weight for the SATURNIN S-box is 2.

E Why Xoodoo is not aligned

In this appendix, we provide a computer-assisted proof that XOODOO is not aligned.

Let us assume that we can factor the linear layer of Xoodoo into $\mathrm{L} = \pi \circ \mathrm{M}$ with M operating on non-trivial superboxes. We can identify the input bits of M that lie in the same superbox with the two following rules:

1. The output bits of L in the same box (column)depend on input bits from the same superbox;
2. Any two output bits that depend on the same input bit must also depend on input bits from the same superbox.

Therefore, we construct a bipartite graph with the 128 output boxes on one side and the 384 input bits on the other side, with edges connecting an output box to the input bits that it depends on. We explicitly constructed this graph (see Figure 11) and checked that it is connected. This contradicts the assumption that M operates on non-trivial superboxes.

```
def buildGraph():
    G = Graph()
    for x in range(4):
        for z in range(32):
            G.add_vertex("out-{0}-{1}".format(x, z))
            for y in range(3):
                G.add_vertex("in-{0}-{1}-{2}".format(x, y, z))
    for x in range(4):
        for z in range(32):
            out = "out-{0}-{1}".format(x, z)
            G.add_edge(out, "in-{0}-{1}-{2}".format(x, 0, z))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 0, (z+27)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 0, (z+18)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 1, (z+26)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 1, (z+17)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+1)%4, 2, (z+19)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+1)%4, 2, (z+10)%32))

            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 1, (z+31)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+2)%4, 0, (z+27)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+2)%4, 0, (z+18)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+2)%4, 1, (z+26)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+2)%4, 1, (z+17)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+0)%4, 2, (z+19)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+0)%4, 2, (z+10)%32))

            G.add_edge(out, "in-{0}-{1}-{2}".format((x+2)%4, 2, (z+13)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 0, (z+16)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 0, (z+ 7)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 1, (z+15)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+3)%4, 1, (z+ 6)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+1)%4, 2, (z+ 8)%32))
            G.add_edge(out, "in-{0}-{1}-{2}".format((x+1)%4, 2, (z+31)%32))
    return G

G = buildGraph()
G.is_connected()
```

Fig. 11: Sage code to construct the graph detailed in the text and to check its connectivity.