IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2023, No. 4, pp. 187–214. DOI:10.46586/tosc.v2023.i4.187-214

Tighter Trail Bounds for Xoodoo

Silvia Mella[1], Joan Daemen[1] and Gilles Van Assche[2]

[1] Radboud University, Nijmegen, The Netherlands
silvia.mella@ru.nl, joan@cs.ru.nl
[2] STMicroelectronics, Diegem, Belgium
gilles-tosc@noekeon.org

Abstract. Determining bounds on the differential probability of differential trails and the squared correlation contribution of linear trails forms an important part of the security evaluation of a permutation. For Xoodoo, such bounds were proven using the trail core tree search technique, with a dedicated tool (XooTools) that scans the space of all r-round trails with weight below a given threshold T_r. The search space grows exponentially with the value of T_r and XooTools appeared to have reached its limit, requiring huge amounts of CPU time to push the bounds a little further. The bottleneck was the phase called trail extension where short trails are extended to more rounds, especially in the backward direction. In this work, we present a number of techniques that allowed us to make extension much more efficient and as such to increase the bounds significantly. Notably, we prove that the minimum weight of any 4-round trail is 80, the minimum weight of any 6-round trail is at least 132 and the minimum weight of any 12-round trail is at least 264, both for differential and linear trails. As a byproduct we found families of trails that have predictable weight once extended to more rounds and use them to compute upper bounds for the minimum weight of trails for arbitrary numbers of rounds.

Keywords: lightweight cryptography · permutation-based cryptography · differential cryptanalysis · linear cryptanalysis · trail bounds

1 Introduction

The Xoodoo cryptographic permutation [DHVV18a] is the core of Xoodyak [DHP+20, DHP+21] and Xoofff [DHP+20]. Xoodyak is a versatile cryptographic object based on the Cyclist construction [DHP+20], intended for lightweight applications, and notably one of the ten finalists of the NIST Lightweight Crypto Standardization process [Nat21]. It can be used to build most symmetric-key functionalities, including hashing, pseudo-random bit generation, authentication, encryption and authenticated encryption. Another cryptographic object that internally uses Xoodoo is the deck function Xoofff [DHVV18a], an instance of Farfalle [BDH+17]. Xoofff is very efficient on a wide range of platforms and can be used for building stream ciphers, MAC functions and (session) authenticated encryption schemes.

The Xoodoo permutation has a classical iterated structure, namely it repeatedly applies a round function to a state, where the number of rounds is a parameter. The choice for the number of rounds is motivated by performance and security requirements. Namely, when plugged in a construction, the resulting permutation shall offer sufficient security margin and still be fast. The number of rounds was chosen such that the constructed primitive (Xoofff or Xoodyak) offers a comfortable safety margin with respect to all known attacks. In Xoofff this is 6 rounds and in Xoodyak this is 12 rounds.

In terms of differential and linear cryptanalysis, this means that it shall not have high-probability differential trails or high-correlation linear trails (or equivalently, low-weight

Received: 2023-09-01 Accepted: 2023-11-01 Published: 2023-12-08

trails), that can be exploited in attacks. Roughly speaking, the data and/or computational complexity of an attack that uses a given trail is exponential in the weight of the trail. So, the higher the weight, the higher the cost of the attack.

The non-existence of low-weight trails is usually proved by determining lower bounds on the weight of trails. This is not enough to guarantee security for the function built on top of the permutation, but it can help in evaluating the resistance against some specific attacks. For instance, in the Farfalle construction used by Xoofff, the expected data complexity to generate internal collisions is directly linked to bounds on the weight of differential trails [DHVV18a, FRD23].

1.1 Prior art

An initial analysis on the differential and linear propagation properties of Xoodoo was presented in [DHVV18a], where lower bounds on the weight of trails were proved for several numbers of rounds. The analysis was later extended and improved bounds were reported in [DHP+20] an in [The21]. We summarize these results in Table 1.

All these results were obtained by using a computer-aided approach for scanning the space of all trails with weight below a given threshold T_r and for a given number of rounds r in a tree based fashion. Since this approach is used in different works [MDV17, DHVV18a, MMGD22, HMMD22, BFR22], we decided to give it a name: *Trail core tree search*. If such space of trails is non-empty, then the trail with the smallest weight defines a tight bound on the weight of all r-round trails. Otherwise T_r defines a bound that is not necessarily tight. The size of the search space increases exponentially with increasing T_r, and so does the computational cost of the search. It follows that the value of T_r that can be achieved in practical time is usually below the actual minimum weight of the trails. In particular, for Keccak-p, Xoodoo, and Ascon, up to now no tight bounds were found for more than 3 rounds.

1.2 The contribution of this paper

In this work, we present a number of methods that allow us to make the search more efficient and, as a consequence, target higher values of T_r in a time that is still practical.

In Section 5 we improve the definition of stability masks introduced in [DHVV18a] and present an optimization technique to extend trails more efficiently in the backward direction, which is the main bottleneck in trail core tree search as conducted until now. This technique leverages the propagation properties of column parity mixers to compute a more accurate lower bound on the weight of trails, allowing us to discard non-useful trails earlier in the search.

In Section 6.3, we present a new approach to scan the space of all 6-round trails with weight below T_6 starting from the space of all 2-round trails with weight below $T_6/3$ instead of the usual approach of starting from 3-round trails of weight below $T_6/2$. This approach allows us to split the search space in subspaces that are smaller and partially overlap, making their exploration faster.

Thanks to such methods, we can improve over known bounds for Xoodoo for different numbers of rounds and we present such results in Section 6. Notably, we prove tight bounds for the weight of linear and differential trails over 4 rounds. Whereas, for 6 and 12 rounds, we prove for the first time (non-tight) bounds beyond 128 and 256, respectively. We summarize our improved bounds in Table 1.

In Section 7 we report on all differential and linear trails we found for 2, 3, and 4 rounds. It is worth noticing that the histograms for the linear trails and the differential trails are very close to each other.

During our search, we encountered some trails with interesting properties. The weight profile of such trails presents a regular and predictable behavior through the rounds. We

Table 1: Lower bounds on the weight of differential and linear trails and the weight of best trails as a function of the number of rounds.

# rounds	Previous works lower bound		Previous works best known		This work lower bound	This work best known
1	2	[DHVV18a]	2	[DHVV18a]	-	-
2	8	[DHVV18a]	8	[DHVV18a]	-	-
3	36	[DHVV18a]	36	[DHVV18a]	-	-
4	74	[DHP+20]	-		80	80
5	94	[DHP+20]	-		98	120
6	108	[The21]	-		132	160
8	148	[DHP+20]	-		176	264
10	188	[DHP+20]	-		220	400
12	222	[DHP+20]	-		264	568

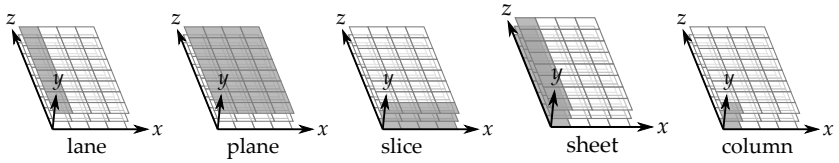


Figure 1: Toy version of the Xoodoo state, with lanes reduced to 8 bits, and different parts of the state highlighted.

discuss such trails in Section 8 and use them to provide upper bounds, which are reported in the last column of Table 1.

The methods that we introduce in this work already appeared in its pre-print version [DMA22] and some of them were later also used to perform trail search in Ascon in [HMMD22]. In fact, these methods are more widely applicable. In particular, the technique used to explore the space of 6-round trail cores (Section 6.3) can be applied to any iterated cipher. The new definition of stability mask, (Eq. 1 in Section 5.1), can be applied to permutations where the set of compatible differences forms an affine space. This is the case in Keccak-f and Ascon for forward extension, but also in ciphers with S-boxes that are differentially 4-uniform. This is not the case in backward extension in Keccak-f and Ascon, but there the definition can be adapted in a natural way by replacing the basis vectors in Eq. 1 by all the compatible differences per S-box. The technique presented in Section 5.2 is instead limited to permutations that have a column parity mixer as mixing layer, like Xoodoo and Keccak-f.

2 Xoodoo

Xoodoo is a bit-oriented iterated permutation operating on a three-dimensional state of $4 \times 3 \times 32$ bits. A bit of the state is specified by (x, y, z) coordinates. A column is composed by 3 bits with given (x, z) coordinates, while a plane is composed by 128 bits with given y coordinate. The Xoodoo state is illustrated in Figure 1 with its different parts. In the following examples, we depict the states as seen from the top with columns collapsing to single cells. The value in a column is indicated by a number: a value (a_0, a_1, a_2), with indices denoting the plane, is encoded as $a_0 + 2a_1 + 4a_2$. Additionally we give them colors to make the figures easier to read. For simplicity, cells with value 0 are left empty. An example is given in Figure 2.

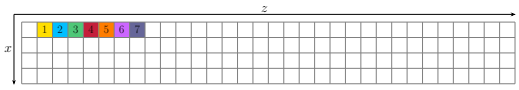


Figure 2: A Xoodoo state seen from the top. Column (0,0) has all bits set to 0. Column (0,1) has bit $y = 0$ set to 1 and the other two set to 0. Column (0,2) has bit $y = 1$ set to 1 and the other two set to 0. Column (0,3) has bit $y = 0$ and $y = 1$ set to 1 and bit $y = 2$ set to 0. Etc.

The round function of Xoodoo is composed by five steps: the mixing layer θ, two plan shifting ρ_{west} and ρ_{east}, the round constants addition ι, and the non-linear layer χ. The number of rounds is a parameter. It is 12 in Xoodyak, and 6 in Xoofff. The i-th round function is defined as $\mathrm{R_i} = \rho_{\text{east}} \circ \chi \circ \iota \circ \rho_{\text{west}} \circ \theta$ with

$$
\begin{aligned}
&\theta: \\
&\qquad P \leftarrow A_0 + A_1 + A_2 \\
&\qquad E \leftarrow P \lll (1,5) + P \lll (1,14) \\
&\qquad A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\} \\
&\rho_{\text{west}}: \\
&\qquad A_1 \leftarrow A_1 \lll (1,0) \\
&\qquad A_2 \leftarrow A_2 \lll (0,11) \\
&\iota: \\
&\qquad A_0 \leftarrow A_0 + \mathrm{C}_i \\
&\chi: \\
&\qquad B_0 \leftarrow \overline{A_1} \cdot A_2 \\
&\qquad B_1 \leftarrow \overline{A_2} \cdot A_0 \\
&\qquad B_2 \leftarrow \overline{A_0} \cdot A_1 \\
&\qquad A_y \leftarrow A_y + B_y \text{ for } y \in \{0,1,2\} \\
&\rho_{\text{east}}: \\
&\qquad A_1 \leftarrow A_1 \lll (0,1) \\
&\qquad A_2 \leftarrow A_2 \lll (2,8).
\end{aligned}
$$

The step θ is a column parity mixer [SD18]. It adds a plane E, called θ-*effect*, to each plane A_y of the state. The θ-effect is computed as the sum of two translated versions of the *parity plane* P, which is obtained by adding the three planes of the state together. On states whose parity plane is all-zero, θ acts as the identity. Such states are those with zero or two non-zero bits in all columns. We say that such states are in the (column parity) kernel. Otherwise, we say that a state is outside the kernel.

The nonlinear layer χ can be seen as the parallel application of 4×32 3-bit S-boxes operating on 3-bit columns, that we denote by χ_3. It is an instance of the transformation χ that was described and analyzed in [Dae95], with interesting propagation properties.

The two steps ρ_{east} and ρ_{west} act as dispersion layers after every application of θ and of χ. Both consists in plane translations over a given offset. Since the state bits interact only within columns in θ and χ, ρ_{east} and ρ_{west} are used to dislocate bits of the same column to different columns. Such breaking up of columns results in weak alignmet [BDPV11].

The step ι consists in the addition of a round constant C_i, that is a plane with a single non-zero lane at $x = 0$.

All steps, except ι, are translation-invariant, that is they operate on bits of the state independently of their position. More formally, a step α is *translation-invariant* over (x, y, z) if $\alpha \circ \tau_{(x,y,z)} = \tau_{(x,y,z)} \circ \alpha$, where $\tau_{(x,y,z)}$ is a (cyclic) translation of the state by (x, y, z). The translation-invariance in horizontal directions of the step mappings results in high symmetry, which is destroyed by the addition of round constants in the step ι. In

Table 2: Offset o and basis vectors v_1, v_2 for the affine spaces $\mathcal{A}(\Delta_{\text{in}})$ and $\mathcal{A}(u_{\text{out}})$.

$\Delta_{\text{in}}/u_{\text{out}}$	o	v_1	v_2
001	001	010	100
011	001	011	100
111	001	011	101

trail search, the round constants additions can be ignored and we can take advantage of this symmetry properties.

2.1 Propagation properties of χ

In Xoodoo, the χ transformation operates on state columns independently. Therefore, its propagation properties can be analyzed at column level through χ_3. The mapping χ_3 is an involution and its propagation properties hold also for its inverse.

Difference propagation As for any transformation with domain $\mathbb{Z}_2^b$, the restriction weight of a differential $(\Delta_{\text{in}}, \Delta_{\text{out}})$ over χ is defined as the inverse of the logarithm of its differential probability (DP):

$$\mathrm{w_r}(\Delta_{\text{in}}, \Delta_{\text{out}}) = b - \log_2|\{x\colon \chi(x) \oplus \chi(x + \Delta_{\text{in}}) = \Delta_{\text{out}}\}|.$$

If the DP is non-zero, we say that Δ_{in} and Δ_{out} are *compatible*, or that Δ_{out} is an output difference compatible with Δ_{in}, or that Δ_{in} is an input difference compatible with Δ_{out}. We refer to a non-zero column in a difference as an *active* column. Since χ acts on each column independently, the DP and weight of a differential are the product and sum of the column differentials over χ_3. For any given input difference δ_{in}, the set of compatible differences over χ_3 forms an affine space $\mathcal{A}(\delta_{\text{in}})$ of size 4 [Dae95]. Table 2 gives such affine spaces in form of offset and basis vectors, for any possible input of χ_3 up to rotation. For a given compatible difference δ_{out}, the restriction weight of $(\delta_{\text{in}}, \delta_{\text{out}})$ depends only on δ_{in} and is equal to $\log_2|\mathcal{A}(\delta_{\text{in}})| = 2$ [Dae95]. As a consequence, the weight of a differential over χ is twice the number of non-zero columns in the input (or output) state difference in the differential.

Linear masks propagation As for any transformation with domain $\mathbb{Z}_2^b$, the correlation weight of a linear approximation $(u_{\text{in}}, u_{\text{out}})$ over χ is defined as the inverse of the logarithm of its squared correlation (C):

$$\mathrm{w_c}(u_{\text{in}}, u_{\text{out}}) = -\log_2 \frac{|\{x \in \mathbb{Z}_2^b \mid {u_{\text{in}}}^T x + {u_{\text{out}}}^T f(x) = 0|}{2^{b-1}} - 1\,.$$

If the correlation is non-zero, we say that u_{in} and u_{out} are *compatible*, or that u_{out} is an output mask compatible with u_{in}, or that u_{in} is an input mask compatible with u_{out}. We refer to a non-zero column in a mask as an *active* column. A state mask over χ is composed by column masks over χ_3 and its correlation and correlation weight depend on the active columns. For any given output mask u_{out}, the set of compatible input masks over χ_3 forms an affine space $\mathcal{A}(u_{\text{out}})$ of size 4 [Dae95]. Offset and basis vectors specifying such affine space are again those in Table 2. The correlation weight of any pair of masks $(u_{\text{in}}, u_{\text{out}})$ depends only on u_{out} and is equal to $\log_2|\mathcal{A}(u_{\text{out}})| = 2$ [Dae95]. As in the differential case, the weight of a state mask is twice the number of non-zero columns in the mask.

3 Trail cores and search strategy

When looking for all trails with weight below a given threshold T_r, the search space grows exponentially with increasing T_r. Trail search can take advantage of the fact that the set of r-round trails can be split in equivalence classes, so that the weight of trails in the same class can be easily bounded. Generating one representative for each class is thus enough for the purpose of proving bounds. Even if the number of such representatives still grows exponentially with T_r, we can reach higher values of T_r compared to what we can achieve when we generate all trails. In [DV12], the set of trails is split in equivalence classes called *trail cores* and methods to lower bound the weight of all trails in a core are presented. In Xoodoo, all trails in a trail core have the same weight and its value can be immediately derived by the number of active columns in the states composing the trail [DHVV18a]. In this section we first recall what trail cores are and how their weight can be computed for Xoodoo, as discussed in [DHVV18a]. Then, we recall the generic strategy introduced in [DV12] to perform trail search, that we will use also in this work.

3.1 Differential probability and differential trail cores

For the study of difference propagation, it is convenient [DHVV18a] to rephase Xoodoo round function as starting with ρ_{east} and ending in χ. Therefore, the round function becomes $\chi \circ \iota \circ \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$. The sequence of linear mappings are grouped in $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$ and we call λ the *linear layer*. Therefore, the re-phased round function becomes $\chi \circ \iota \circ \lambda$. Since constant addition does not influence the trail weight, ι can be ignored.

An r-round differential trail is a sequence $(a_0, a_1, \ldots, a_r)$ of r round differentials and its restriction weight is the sum of the restriction weights of the round differentials that compose the trail. We use a redundant representation of trails, where we also specify the differences after the linear layer: $(a_0, b_0, a_1, b_1 \ldots, b_{r-1}, a_r)$ with $b_i = \lambda(a_i)$.

Since λ is linear, the restriction weight of the trail only depends on the differentials over χ and is given by $\sum_i \mathrm{w_r}(b_{i-1}, a_i)$.

Two differences b_i and a_{i+1} are compatible over χ only if they have the same column activity pattern, i.e., they have the same active column indexes, and each pair of columns is compatible over χ_3. As shown in Section 2.1, the restriction weight $\mathrm{w_r}(b_i, a_{i+1})$ is twice the number of active columns in b_i, or equivalently, in a_{i+1}. It follows that the restriction weight of an r-round trail $(a_0, b_0, a_1, b_1, \ldots, b_{r-1}, a_r)$ is fully determined by the sequence $b_1, \ldots, b_{r-1}$ and can be expressed as $\mathrm{w_r}(a_1) + \sum_{1 \le i < r} \mathrm{w_r}(b_i)$. It is thus independent of the value of a_0, b_0, and a_r. As in [DV12], we call *differential trail core* the sequence:

$$Q = a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda} b_2 \xrightarrow{\chi} a_3 \xrightarrow{\lambda} \ldots a_{r-1} \xrightarrow{\lambda} b_{r-1} \ .$$

We call the sequence $\mathrm{w_r}(a_1), \mathrm{w_r}(b_1), \ldots, \mathrm{w_r}(b_r)$ the *weight profile* of the trail core.

A differential trail core defines a set of $2^{\mathrm{w_r}(a_1)} \times 2^{\mathrm{w_r}(b_{r-1})}$ r-round trails with the same weight. Such trails are all trails of the form (a_0, b_0, Q, a_r) for any value of a_0, b_0, and a_r such that b_0 is compatible with a_1, $a_0 = \lambda^{-1}(b_0)$ and a_r is compatible with b_{r-1}. With the goal of lower bounding the weight of trails, in our analysis we can thus limit ourselves to bound the weight of trail cores.

A differential trail core can be extended to an $r+1$-round differential trail core by either pre-pending a couple a_0, b_0 with b_0 compatible with a_1 and $a_0 = \lambda^{-1}(b_0)$, or by appending a couple a_r, b_r with a_r compatible with b_{r-1} and $b_r = \lambda(a_r)$. In the former case, we speak about extension in the backward direction, while in the latter case we speak about extension in the forward direction.

3.2 Correlation and linear trail cores

An r-round linear trail is a sequence of r linear approximations. A linear approximation over round i is defined by a mask a_i at its output and a mask a_{i+1} at its input and we denote its correlation value $\mathrm{C}(a_i, a_{i+1})$. The correlation weight of trail is the sum of the correlation weight of its round linear approximations.

As for differential trails, we use a redundant representation of trails by including the masks after the linear layer. Also in this case, it is convenient to rephase the Xoodoo round function to make notation consistent between linear and differential trails [DHVV18a]. As linear propagation is studied from the output to the input, the round is rephased starting with χ and ending with λ, so that the trail first encounters λ and then χ of each round (as in the differential case).

A trail is of the form $(a_0, b_0, a_1, b_1 \ldots, b_{r-1}, a_r)$, where a_0 is the mask after the last round and a_r the mask before the first round. A mask a_i at the output of λ maps to a mask $b_i = \lambda^\top(a_i)$ before λ, where $\lambda^\top$ denotes the linear mapping obtained by the multiplication of a matrix that is the transpose of the matrix defining the linear mapping λ. It follows that $\lambda^\top = \rho_{\text{east}}^\top \circ \theta^\top \circ \rho_{\text{west}}^\top = \rho_{\text{east}}^{-1} \circ \theta^\top \circ \rho_{\text{west}}^{-1}$ since the inverse of a bit transposition matrix is its transpose.

The correlation weight of a round linear approximation (a_i, a_{i+1}) depends only on the correlation weight of (b_i, a_{i+1}) over χ, which is non-zero only if b_i and a_{i+1} have the same column activity pattern. Since the weight of each column is 2, the correlation weight is twice the number of active columns in b_i, or equivalently, in a_{i+1}. Therefore, the correlation weight of an r-round trail $(a_0, b_0, a_1, b_1, \ldots, b_{r-1}, a_r)$ is fully determined by the sequence $b_1, \ldots, b_{r-1}$ and is given by $\mathrm{w_c}(a_1) + \sum_{1 \le i < r} \mathrm{w_c}(b_i)$. A *linear trail core* is defined as the sequence

$$Q = a_1 \xrightarrow{\lambda^\top} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda^\top} b_2 \xrightarrow{\chi} a_3 \xrightarrow{\lambda^\top} \ldots a_{r-1} \xrightarrow{\lambda^\top} b_{r-1} \ .$$

It defines a set of $2^{\mathrm{w_c}(a_1)} \times 2^{\mathrm{w_c}(b_{r-1})}$ r-round trails that share the sequence Q and that differ by the initial masks a_0, b_0 and the final mask a_r. Similarly to differential trail cores, linear trail cores can be extended to $r+1$ rounds by extension in the forward or backward direction. We call the sequence $\mathrm{w_c}(a_1), \mathrm{w_c}(b_1), \ldots, \mathrm{w_c}(b_r)$ the *weight profile* of the trail core.

3.3 Trail search strategy

Given the similarities between the study of differential and linear trails, a unified notation is introduced in [DHVV18a] and we adopt that notation in this paper. A trail core is denoted by

$$Q = a_1 \xrightarrow{\lambda^\star} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda^\star} b_2 \xrightarrow{\chi} a_3 \xrightarrow{\lambda^\star} \ldots a_{r-1} \xrightarrow{\lambda^\star} b_{r-1}$$

where $\lambda^\star = \rho_{late} \circ \theta^\star \circ \rho_{early}$ and

- $\lambda^\star = \lambda$, $\rho_{early} = \rho_{\text{east}}$, $\theta^\star = \theta$ and $\rho_{late} = \rho_{\text{west}}$ for differential trails, and
- $\lambda^\star = \lambda^\top$, $\rho_{early} = \rho_{\text{west}}^{-1}$, $\theta^\star = \theta^\top$ and $\rho_{late} = \rho_{\text{east}}^{-1}$ for linear trails.

This convention is illustrated in Figure 3.

We denote the weight of a trail by w, where $\mathrm{w} = \mathrm{w_r}$ for differential trails and $\mathrm{w} = \mathrm{w_c}$ for linear trails.

Let r be an even number. For any r-round trail core $(a_1, b_1, \ldots, b_{r-1})$ of weight below T_r, either $\mathrm{w}(a_1, \ldots, b_{r/2}) < T_r/2$ or $\mathrm{w}(a_{r/2}, \ldots, b_{r-1}) < T_r/2$, otherwise their sum would exceed T_r. It follows that such r-round trail cores can be generated by generating all $\frac{r}{2}$-round trail cores with weight below $T_r/2$ and extending them in the forward and backward direction to reach the desired number of rounds.

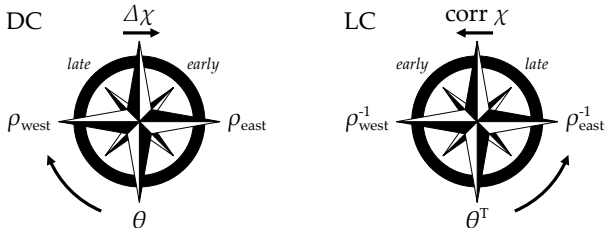


Figure 3: Conventions for differential (DC) and linear (LC) trails in the round function.

This reasoning can of course be generalized to other partitionings, and it can be iterated until reaching the shortest trail cores which are 2-round trail cores. See Section 6 for more details.

A 2-round trail core is fully specified by a state a and its weight is given by $\mathrm{w}(a) + \mathrm{w}(\lambda^\star(a))$. If τ is a translation over x and z directions, then $\mathrm{w}(\tau(a)) + \mathrm{w}(\lambda^\star(\tau(a))) = \mathrm{w}(a) + \mathrm{w}(\lambda^\star(a))$. In other words, the 2-round trail cores specified by a and $\tau(a)$ have the same weight. We can thus partition all states a in equivalence classes where states in the same class are translated versions of each other. For the goal of lower bounding the weight of trails, it is thus sufficient to generate only one representative per class. We call such representative state (and its corresponding trail core) *canonical*.

Canonical states a such that $\mathrm{w}(a) + \mathrm{w}(\lambda^\star(a)) < T_2$ are iteratively generated by using a tree-based approach with an efficient method to lower bound the weight of nodes that allows to efficiently prune branches. In this work we rely on the 2-round trail cores generation of [DHVV18a] and refer to that paper for details.

Once 2-round trail cores are generated, they have to be extended. Extension is in general an expensive operation, whose cost depends on the number of trail cores to extend and the number of active columns in each trail core. In fact, for any state b at the input of χ there are $2^{\mathrm{w}(b)}$ compatible states at the output of χ. In the next sections we recall how trail extension works for Xoodoo and how we optimized it in order to achieve higher target weights.

4 Trail core extension

In this section we recall the methods presented in [DHVV18a] to extend trail cores. These are up to now the most efficient ones to perform trail core extension in Xoodoo.

4.1 Affine space of compatible states

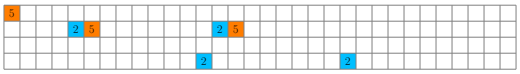
Extending a trail core means looking for all the possible states that are compatible through χ with its first or with its last state. The latter is called *forward extension*, while the former is *backward extension*.

In forward extension, each active column $b[x, z]$ at the input of χ defines an affine space $o + \langle v_1, v_2 \rangle$ in the same column at the output of χ, according to Table 2. Gathering all the active columns, a description of the affine space $\mathcal{A}(b)$ of states a at the output of χ is given by $\mathcal{A}(b) = O + \langle V_1, \ldots, V_{\mathrm{w}} \rangle$, where the offset O and the basis vectors V_i's are defined as follows. The offset is a state O with column (x, z) equal to the offset specified by column $b[x, z]$; if the column is passive, then also the corresponding column in O is passive. The offset has thus the same number of active columns as b. Then, for the two column vectors v_1, v_2 specified by the active column $b[x, z]$, we build two states V_1, V_2 whose columns are all zero except column (x, z) that gets the value of one of the two column vectors:

$V_1[x,z] = v_1$ and $V_2[x,z] = v_2$. The dimension of the basis is the weight of b, that is twice the number of its active columns.

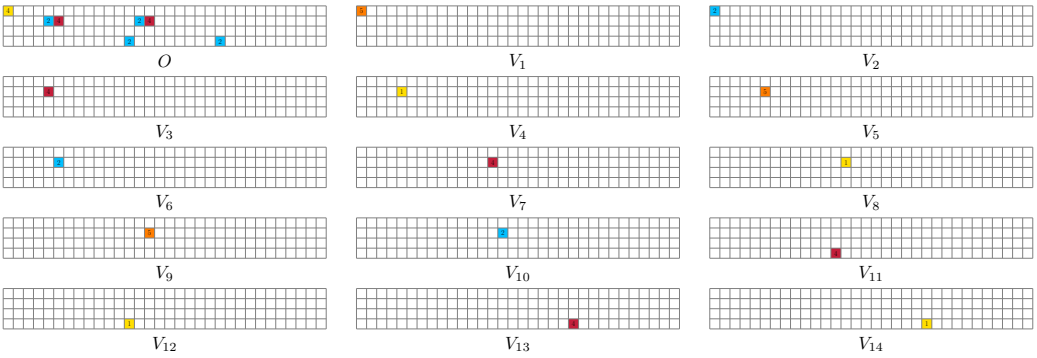
Backward extension works similarly since χ is an involution. From a state a at the output of χ, we define the set of states b at the input of χ as an affine space $\mathcal{B}(a)$.

Example 1. Consider the following difference a that we want to extend in the backward direction.



a

The state a has seven active columns, each with value 2 or 5. Therefore, the affine space $\mathcal{B}(a)$ has an offset with seven active columns and fourteen basis vectors. They are the following:

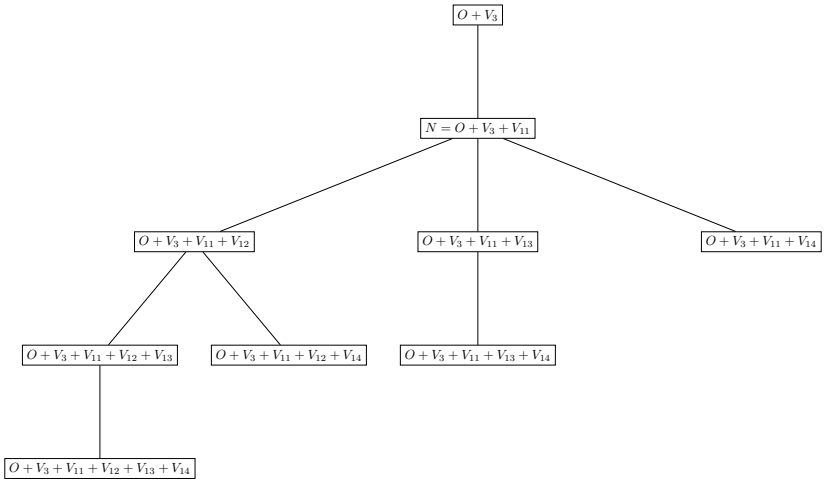


△

4.2 Extension as a tree-based search

The affine space $\mathcal{A}(b)$ or $\mathcal{B}(a) = O + \langle V_1, \ldots, V_{\mathrm{w}} \rangle$ can be scanned in a tree-based search. At the root of the tree there is the offset O. A node at level i is composed by the offset plus i vectors of the basis. To avoid duplicates, we introduce an order relation among basis vectors; arbitrarily, we say that $V_i \prec V_j$ if and only if $i < j$. The children of a node are obtained by adding a new basis vector to the node, running over all remaining vectors that are larger than the last vector composing the node, according to the order relation adopted. More formally, let a node be defined by $O + V_{i_1} + \ldots + V_{i_k}$. Its children are all the nodes of the form $O + V_{i_1} + \ldots + V_{i_k} + V_{i_{k+1}}$ with $i_{k+1} \in \{i_k + 1, \ldots, \mathrm{w}\}$. Alternatively, the parent of node $O + V_{i_1} + \ldots + V_{i_{k-1}} + V_{i_k}$ is $O + V_{i_1} + \ldots + V_{i_{k-1}}$ if $i_k = \max\{i_1, \ldots, i_k\}$.

Example 2. Consider the affine space in Example 1 and consider the node $N = O + V_3 + V_{11}$. The parent of this node is $O + V_3$ and its children are $O + V_3 + V_{11} + V_{12}$, $O + V_3 + V_{11} + V_{13}$ and $O + V_3 + V_{11} + V_{14}$. They are depicted below with the other descendants of N.



△

4.3 Limiting the search using the far view

Scanning the entire affine space takes $2^{\mathrm{w}(a)}$ (or $2^{\mathrm{w}(b)}$) iterations and becomes prohibitively expensive if a (or b) has high weight. However, we do not need to scan the whole space, but we need to build only the compatible states with a weight below the target, when we look at them after $\lambda^\star$ (or $\lambda^{\star-1}$ for backward extension). As a convention, we say that the *near view* is when we look at states just after χ (or just before χ for backward extension), and the *far view* after $\lambda^\star$ (or $\lambda^{\star-1}$).

Since $\lambda^\star$ is a linear operation, we can easily transpose the affine space $\mathcal{A}(b)$ through the different steps mappings by applying the step mappings to the offset and the basis elements. This goes up to the input to the next χ and we obtain the spaces $\rho_{early}(\mathcal{A}(b))$, $\theta^\star(\rho_{early}(\mathcal{A}(b)))$, and $\lambda^\star(\mathcal{A}(b))$. If

$$\mathcal{A}(b) = O^{\text{near}} + \langle V_1^{\text{near}}, \ldots, V_{\text{w}}^{\text{near}} \rangle .$$

then we have

$$\mathcal{B}(b) = \lambda^\star(\mathcal{A}(b)) = O^{\text{far}} + \left\langle V_1^{\text{far}}, V_2^{\text{far}}, \ldots, V_{\text{w}}^{\text{far}} \right\rangle$$

with $O^{\text{far}} = \lambda^\star(O^{\text{near}})$ and $V_i^{\text{far}} = \lambda^\star(V_i^{\text{near}})$. This is depicted in Figure 4.

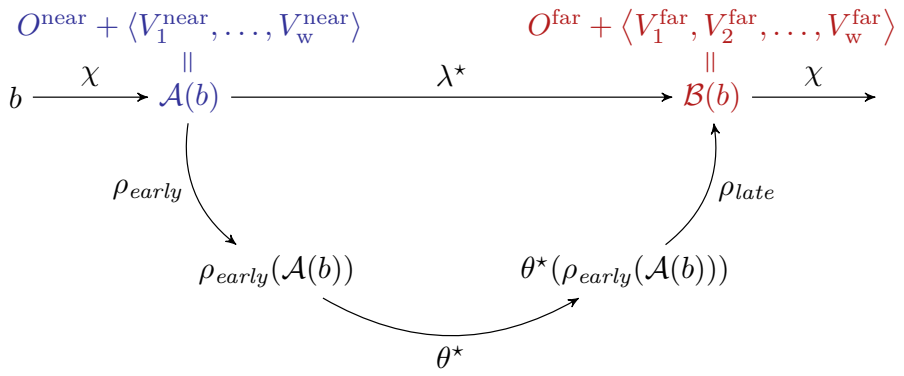


Figure 4: Near view and far view for forward extension.

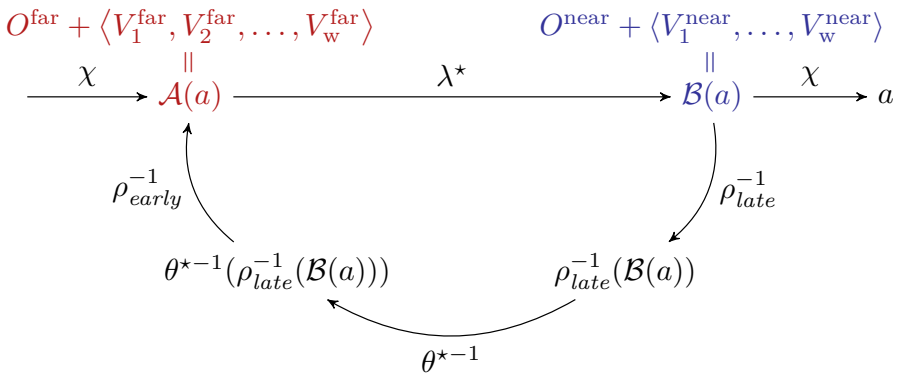


Figure 5: Near view and far view for backward extension.

Similarly for the backward extension, we can transpose

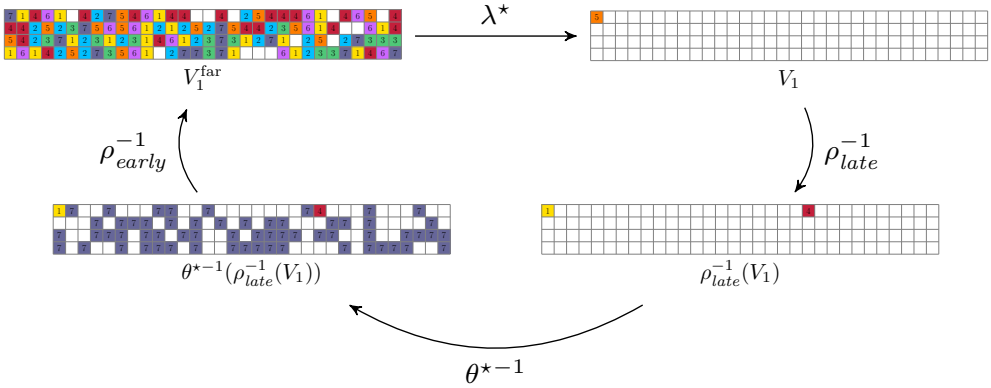
$$\mathcal{B}(a) = O^{\text{near}} + \langle V_1^{\text{near}}, \ldots, V_{\text{w}}^{\text{near}} \rangle$$

to $\rho_{late}^{-1}(\mathcal{B}(a))$, $\theta^{\star -1}(\rho_{late}^{-1}(\mathcal{B}(a)))$, and

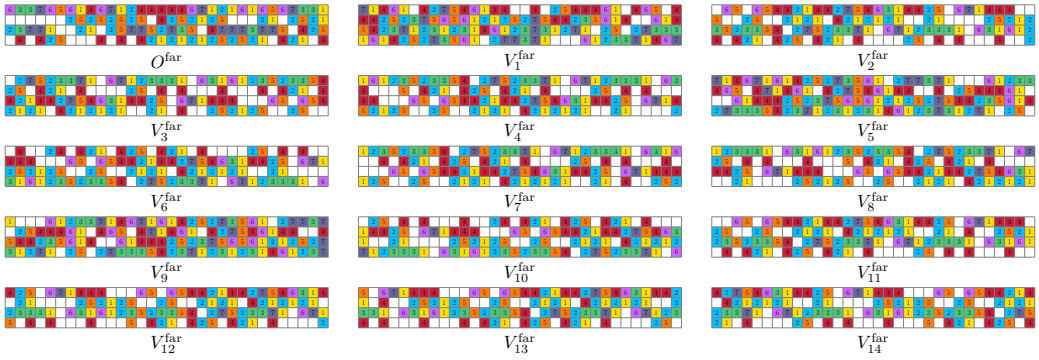
$$\mathcal{A}(a) = \lambda^{\star -1}(\mathcal{B}(a)) = O^{\text{far}} + \langle V_1^{\text{far}}, V_2^{\text{far}}, \ldots, V_{\text{w}}^{\text{far}} \rangle$$

with $O^{\text{far}} = \lambda^{\star -1}(O^{\text{near}})$ and $V_i^{\text{far}} = \lambda^{\star -1}(V_i^{\text{near}})$. This is depicted in Figure 5.

Example 3. Consider the difference a of Example 1 and assume we want to extend it in the backward direction. Therefore the offset and basis vectors in Example 1 represent the space $\mathcal{B}(a)$ in the near view. We consider the basis vector V_1 in the near view and we map it through the different steps of $\lambda^{\star -1}$. We obtain the following differences.



Similarly, we obtain the offset and the other basis vectors in the far view. The resulting representation of $\mathcal{A}(a)$ is the following.



△

All the elements of the space in the far view can again be built by a tree-based approach, and naturally we say that $V_i^{\text{far}} \prec V_j^{\text{far}}$ if and only if $i < j$.

Notice that a node can have weight smaller than the weight of its parent, since the addition of a new basis vector can turn some active bits into passive bits, possibly decreasing the number of active columns. This means that when a node is encountered whose weight is higher than the target weight, we still need to build all its descendants and check if their weight is below the target. However, it is possible to lower bound the weight of the descendants of a node, so that if such lower bound is higher than the target weight, we can safely discard them [DHVV18a]. The more accurate such bound is, the more efficient the search becomes. In this work, we will use the term *score* to refer to such lower bound, as introduced in [MMGD22].

4.4 Computing the score based on stability masks

For a given node $O + V_{i_1} + \ldots + V_{i_k}$ in the far view (we omit the "far" superscript here), we define its *stable bits* as the set of bits that have in all the node's descendants the same value as the bits in the node. Note that the stability of bits depends only on the index of the last basis element in the node encountered, in this case i_k, as only basis elements with index $j > i_k$ can be further added.

We represent the stable bits as a *stability mask*, i.e., a state value where a bit has a value 1 if it is stable and 0 otherwise. We denote the stability mask of a node with last basis element with index i by S_i. S_0 is the stable bit mask for the entire basis and S_{w} is the all-one mask.

For a given node $N = O + V_{i_1} + \ldots + V_{i_k}$, we therefore have that $N \wedge S_{i_k}$ gives the active bits of N that are guaranteed to remain active in all its descendants. It follows that the weight of any descendant of N has weight at least $\text{w}(N \wedge S_{i_k})$. This is the score of the node N, which lower bounds the weight of all nodes in the subtree under N. Similarly, the score of the root node is $\text{w}(O \wedge S_0)$.

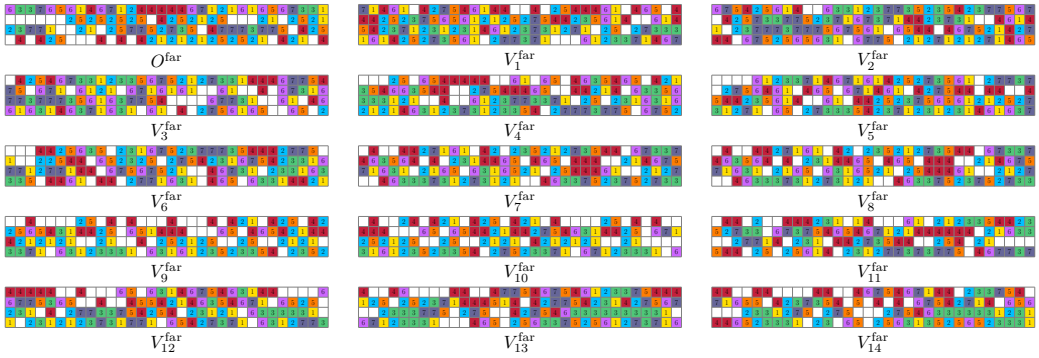
4.5 Triangularization

The stability masks S_0 through S_{w} depend on the basis $\langle V_1, V_2, \ldots, V_{\text{w}} \rangle$ in the far view. To limit the search as much as possible, we would like that the number of stable bits in S_i grows quickly with i. This way, the score computes the weight over many columns and the search can prune early.

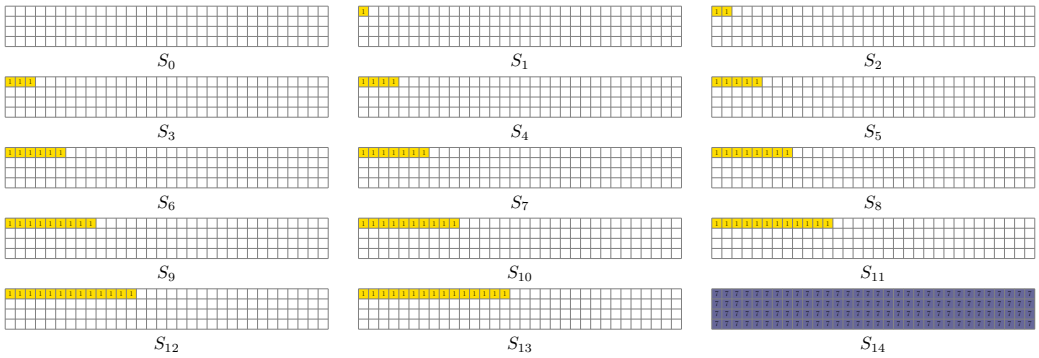
To this end, we proposed in [DHVV18a] to triangularize the basis $\langle V_1, V_2, \ldots, V_{\text{w}} \rangle$. Triangularization of the basis $\langle V_1, V_2, \ldots, V_{\text{w}} \rangle$ requires the establishment of an order relation of the bit positions $p = (x, y, z)$ in the state, for instance the lexicographic order relation. Let $\langle V_1, V_2, \ldots, V_{\text{w}} \rangle$ be a triangularized basis and let p_i be the smallest active

bit in V_i. Then by construction, all bits in positions $\leq p_i$ are passive in all vectors in $V_{i+1}, \ldots, V_{\mathrm{w}}$, and they are consequently included in the stability mask S_i.

Example 4. Consider the basis of Example 3. If we triangularize it following the lexicographic order $[y, x, z]$ we obtain the following offset and vectors.



These lead to the following stability masks.



The number of stable bits in each mask S_i is given by the Hamming weight of S_i. The sequence of such Hamming weights is therefore

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 384.$$

△

5 Optimizing the trail core extension

In this section we describe optimization techniques to make the number of stable bits grow more quickly.

5.1 Redefining stability masks

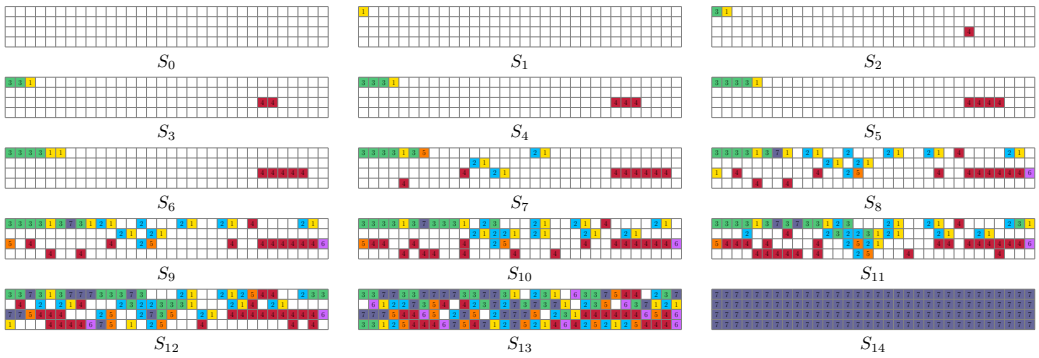
We noticed that the implementation of the trail extension in [DHVV18a] is suboptimal. It computes the score using a stability mask that includes the bits in positions smaller than p_{i+1}, but there can be more bits that are stable. As a matter of fact, for a given node with last basis vector V_i, all bits in positions where the basis vectors V_{i+1} to V_{w} are all

zero, are stable. Hence, we define the stability mask of a node as the bitwise AND of the complements of the basis vectors that can be added to the node to form its descendents:

$$S_i = \bigwedge_{j>i} \overline{V_j} \ . \tag{1}$$

The example below shows how, with this new definition of stability mask, the number of stable bits grows more quickly with i.

Example 5. Consider the triangularized basis given in Example 4. By applying the definition given in (1) we obtain the following stability masks.



The sequence of Hamming weights of the stability masks, for increasing i, is therefore

$$0, 1, 4, 7, 10, 13, 15, 27, 46, 49, 66, 85, 122, 212, 384$$

△

5.2 Triangularization based on θ-effect

To speed things up even further, we introduce new ideas to improve the triangularization. The bottleneck is the backward extension, so we focus on that one.

As shown in Table 2, the affine basis elements of $\mathcal{B}(a)$ contain only one or two active bits each in the near view. In the latter case, the two bits end up in different columns at the input of $\theta^{\star -1}$ because of ρ_{late}^{-1}, so in both cases none of the basis elements are in the kernel. As the inverse of $\theta^\star$ is dense on states outside the kernel, computing $(\lambda^\star)^{-1}$ over such a basis gives elements with many active bits. See for instance Example 3. As all basis vectors have many active bits, their complements typically have only half of their bits equal to 1 and due to the fact that the stability masks are the bitwise AND, they are rather sparse unless i is close to w. See for instance Example 5.

We can make the triangularization more effective by adopting a different ordering of bit positions, leading to more dense stability masks. For this purpose, we are going to look at columns before and after $\theta^\star$. When we look at states at the input of $\theta^\star$ we call it the *mid view*.

The θ-effect is defined as the sum of an input and an output of $\theta^\star$, i.e., $E(x) = x \oplus (\theta^\star)^{-1}(x)$ for a given output x of $\theta^\star$. Because $\theta^\star$ is a column parity mixer, the bits in any given column of $E(x)$ are either all three active or all three passive. So, the basis elements in the mid view have one or two active bits plus a number of completely active columns.

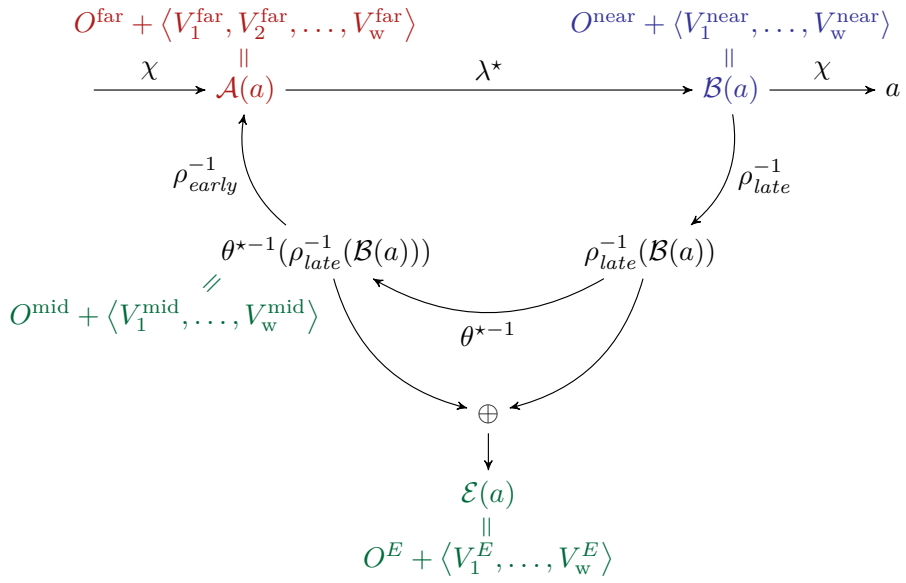
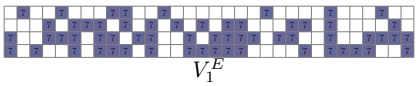


Figure 6: Mid view and θ-effect for backward extension.

Example 6. Consider the basis vector V_1^{near} as in Example 3. In the mid view, $V_1^{\text{mid}} = \theta^{\star -1}(\rho_{late}^{-1}(V_1^{\text{near}}))$ has two columns with one active bit and all other active columns have three active bits. It is in fact given by the sum of $\rho_{late}^{-1}(V_1^{\text{near}})$ and the corresponding θ-effect $V_1^E = E(\rho_{late}^{-1}(V_1^{\text{near}}))$:



△

The idea is therefore to triangularize in a way that takes a whole active column in the mid view as a pivot. This eliminates the three bits of the column in all other basis elements, and therefore stabilizes three bits in three different columns in the far view, i.e., after going through ρ_{early}^{-1}. To do this, we perform triangularization over the θ-effect.

Let us describe this procedure more precisely. First, we let

$$\mathcal{E}(a) = E(\rho_{late}^{-1}(\mathcal{B}(a)))$$

be the affine space representing the θ-effect of $\mathcal{B}(a)$. In other words, if

$$\mathcal{B}(a) = O^{\text{near}} + \langle V_1^{\text{near}}, \ldots, V_{\text{w}}^{\text{near}} \rangle .$$

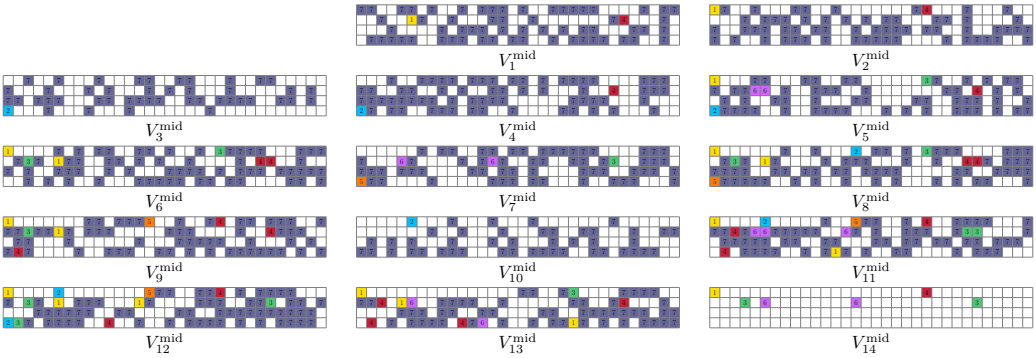
then

$$\mathcal{E}(a) = O^E + \langle V_1^E, \ldots, V_{\text{w}}^E \rangle$$

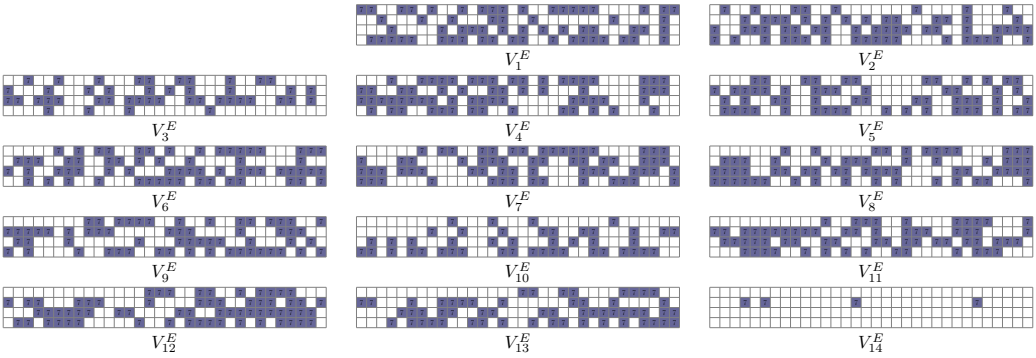
with $O^E = E(\rho_{late}^{-1}(O^{\text{near}}))$ and $V_i^E = E(\rho_{late}^{-1}(V_i^{\text{near}}))$. This is depicted in Figure 6.

Then, we triangularize $\langle V_1^E, \ldots, V_{\text{w}}^E \rangle$ and modify the representation of $\mathcal{B}$ accordingly: Every time we add V_i^E to V_j^E in $\mathcal{E}(a)$, we add V_i^{near} to V_j^{near} in $\mathcal{B}(a)$. Finally, we transpose $\mathcal{B}(a)$ to $\mathcal{A}(a) = \lambda^{\star -1}(\mathcal{B}(a))$.

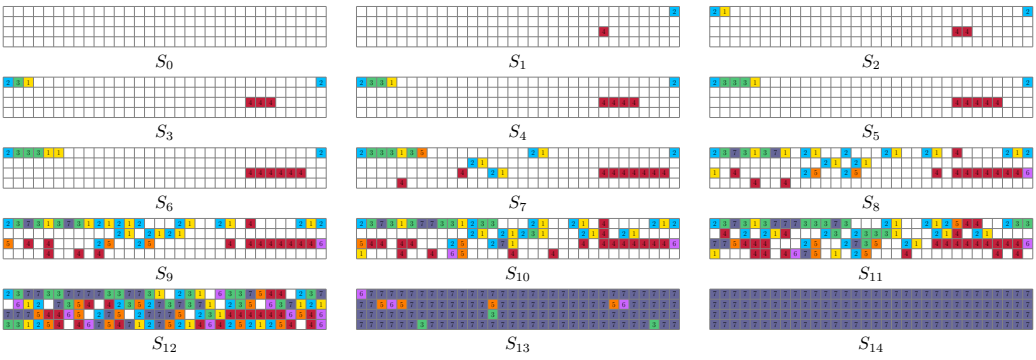
Example 7. Consider the basis given in Example 1. For each basis vector V_i^{near} we can compute the corresponding $\rho_{late}^{-1}(V_i^{\text{near}})$, V_i^{mid}, and V_i^E as we did in Example 3 and 6 for V_1^{near}. Then, by performing triangularization based on the θ-effect following the lexicographic order $[y, x, z]$, we obtain the following basis for the mid view.



The corresponding θ-effects are the following. We can notice that the pivot column for E_1 is in position $(0, 0)$, for E_2 it is in position $(0, 1)$, for E_3 it is in position $(0, 2)$, and so on.



Once the basis for the mid view is mapped to the far view, we obtain the following sequence of stability masks.



The sequence of Hamming weights of the stability masks, for increasing i, is therefore

$$0, 2, 5, 8, 11, 14, 16, 28, 50, 59, 81, 119, 207, 374, 384$$

△

We can notice that the number of active bits in the stability masks in Example 7 does not increase by at least 3 for each i, which would be the optimal behavior. Moreover, we can also notice that even if the number of stable bits increases by 3 from S_i to S_{i+1}, some of these bits may end up in the same column. Hence, the number of new active columns

can be smaller than 3, and the score will be computed on less active columns than the optimal situation.

We now discuss what are the conditions that lead to such and other non-optimal behavior and explain how we can refine our method to overcome them. Both our refinements are related to the order in which we consider the columns (x, z) when choosing a pivot.

5.2.1 Refinement based on go-columns

First, we notice that vector V_i^{mid} cannot lead to 3 new stable bits when there are active bits in V_j^{mid} in the same pivot column of V_i^{mid} for some $j > i$.

Example 8. Consider the basis for the mid view in Example 7. The pivot column of V_1^{mid} is column $(0, 0)$. The basis vectors V_j^{mid} with $j \in \{2, 5, 6, 8, 9, 11, 12, 13, 14\}$ all have an active bit in this column, i.e. the bit $(0, 0, 0)$. Therefore, the three bits of $V_1^E[0, 0]$ cannot be considered all stable, but only the two bits in $y = 1$ and $y = 2$. These are in fact the two bits that appear in S_1 (mapped through ρ_{early}^{-1}).

△

Moreover, if the offset O^{mid} already contains one or more bits in the pivot column of V_i^{mid}, the addition of V_i^{mid} will cancel them. It follows that less bits will be considered when computing the score.

To overcome these problems, when looking at $\rho_{late}^{-1}(\mathcal{B}(a))$, we divide the columns (x, z) in two sets: the *go* columns are those that do not contain any active bit in either the offset or the basis elements of $\rho_{late}^{-1}(\mathcal{B}(a))$, and the *no-go* columns are the remaining ones. When choosing a pivot in the triangularization of $\mathcal{E}(a)$, we look at go columns in priority. Only if there are no more go columns available, then we choose a no-go column.

The reason for favoring go columns can be explained more formally as follows. Assume that (x, z) is a go column and that it is taken as a pivot in the triangularization of $\mathcal{E}(a)$. Because it is a go column, we have:

$$\rho_{late}^{-1}(b)[x, z] = (0, 0, 0) \text{ for all } b \in \mathcal{B}(a).$$

Since it is taken as pivot, it means that

$$V_i^E[x, z] = 1 \text{ and } V_i^{\text{mid}}[x, z] = (1, 1, 1)$$

for some basis vector V_i^{near} of $\mathcal{B}(a)$, and that the column (x, z) is going to be inverted by $\theta^{\star -1}$ when adding $\rho_{late}^{-1}(V_i^{\text{near}})$. Then ρ_{early}^{-1} will move these three bits to three different columns in the far view. Furthermore, they are stable after taking the i-th basis element as pivot, because they do not appear in any later basis vector. So S_i includes them. Three stable active bits in three different columns count as 6 in the score $\mathrm{w}(N \wedge S_i)$, so this is optimal when aiming at pruning the tree as soon as possible.

Choosing a no-go column is less favorable, so they are used as pivot near the end of the triangularization procedure. When the offset $\rho_{late}^{-1}(O^{\text{near}})[x, z]$ is non-zero, the active bits in that column will be turned passive after applying $\theta^{\star -1}$, therefore counting as less than 6 in the score. As another case, assume that V_i^{near} with $V_i^E[x, z] = 1$ is taken as pivot and that some basis vector $\rho_{late}^{-1}(V_j^{\text{near}})[x, z]$ with $j > i$ is non-zero. Then, at least one of these bits will not be stable and cannot be taken into account when computing the score.

5.2.2 Refinement based on a diagonal pattern

Another non-optimal situation that can happen is due to the fact that we usually follow a horizontal pattern during triangularization, i.e., after pivot column (x, z), we consider pivot column $(x, z + 1)$. In the differential case, the three bits $(x, 0, z)$, $(x, 1, z)$, $(x, 2, z)$ of the first pivot column will be mapped to $(x, 0, z)$, $(x, 1, z - 1)$, $(x + 2, 2, z - 8)$, respectively,

after $\rho_{early}^{-1} = \rho_{\text{east}}^{-1}$. But for the second pivot column, $(x, 1, z+1)$ will be mapped to $(x, 1, z)$, sitting in the same column as $(x, 0, z)$ after ρ_{early}^{-1}. Hence, because of this overlap, the score will be computed on at most 5 columns in the far view instead of 6.

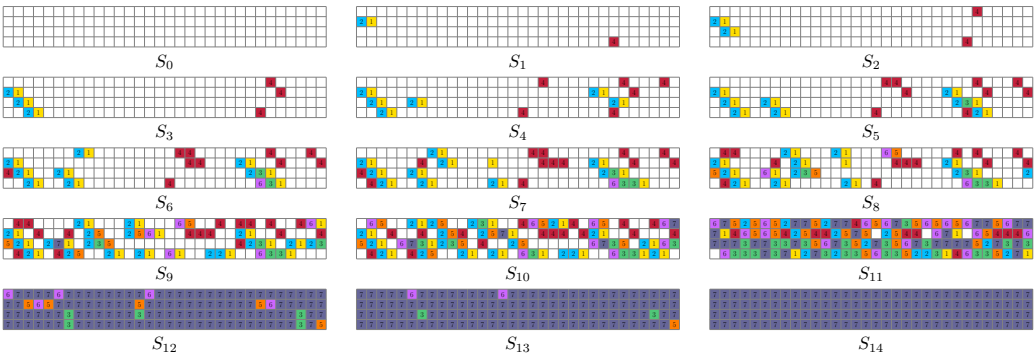
Example 9. Consider the basis in the mid view given in Example 7. The pivot column of V_2^{mid} is $(0, 1)$. Its three bits are mapped to bits $(0, 0, 1)$, $(0, 1, 0)$, $(2, 2, 25)$ after ρ_{early}^{-1}. They are the bits that appear in S_2 but not in S_1. The pivot column of V_3^{mid} is $(0, 2)$. Its three bits are mapped to bits $(0, 0, 2)$, $(0, 1, 1)$, $(2, 2, 26)$ after ρ_{early}^{-1}. They are the bits that appear in S_3 but not in S_2. Bit $(0, 1, 1)$ goes in the same column of bit $(0, 0, 1)$. Therefore the number of active columns increases from 5 in S_2 to 7 in S_3.

△

To limit this effect, when we look for a pivot column we follow a diagonal pattern in the θ-view. Namely, we loop over $(x, z) = (k \bmod 4, (k + \lfloor k/32 \rfloor) \bmod 32)$ for $k \in \mathbb{Z}_{128}$.

To avoid potential misunderstanding due to similar terminology, we highlight that this diagonal pattern differs from the diagonal coordinate introduced in [BFR22, Section 5.3]. In fact, the authors of [BFR22] adopt a diagonal ordering of units to maximize the number of stable coordinates in their 2-round trail cores generation, whereas our ordering is meant for trail extension instead. Therefore, while the final goal of both approaches is to compute more accurate lower bounds for the weight during the trail search, they focus on different aspects of the search.

Example 10. Consider the affine space given in Example 1. By applying the new triangularization technique based on the θ-effect and the refinements based on the go-columns and the diagonal pattern, we finally obtain the following stability masks.



The sequence of Hamming weights of the stability masks, for increasing i, is therefore

$$0, 3, 6, 9, 18, 27, 33, 43, 58, 86, 119, 251, 369, 379, 384 \ .$$

We can notice that the number of stable bits in this example increases by at least 3 with each i, unlike in the previous examples.

△

Clearly, we can see that the number of stable bits increases much faster in Example 10 than in Example 4. As active columns count for 2 in the weight, each stability bits potentially contributes to 2 to the score. Once the score reaches the threshold weight of the extension, the entire subtree of descendants is pruned. Since the number of states to investigate increases exponentially as we go down the tree, it is clear that pruning earlier has a direct and significant reduction on the search time. This highlights the benefits of having a tighter score and a faster increasing number of stability bits.

Table 3: Details on the generation of r-round canonical differential trail cores with weight below T_r. Timings are rounded to the closest integer and in the fourth column they are meant to be incremental. Timings between parenthesis means that the same computation has been already done in a previous step, so it is counted only once. - means that the step is not performed since the starting set is empty. * the time with XooTools on a desktop PC equipped with an Intel® Core™ i5-6500 CPU.

search space				details				
r	T_r	# cores found	time	step	r	T_r	# cores found	time
4	88	4	95d	generation	2	44	2,983,444,980	301h
				forw.ext.	3	86	2,283,985	27h
				forw.ext.	4	88	3	3m
				back.ext.	3	86	291,994	1942h
				back.ext.	4	88	1	1m
5	98	0	21d	generation	2	44	2,983,444,980	(301h)
				forw.ext.	5	98	0	28h
				generation	3	54	201	480h*
				back.ext.	5	98	0	1s
6	132	0	21s	generation	3	54	201	(480h*)
				forw.ext.	6	132	0	3s
				back.ext.	6	132	0	1s
				generation	2	44	2,983,444,980	(301h)
				ext. from start	6	132	0	(27h+) 16s
				ext. from middle	6	132	0	(27h+) 1s
				ext. from end	6	132	0	(1942h)
8	176	0	2s	generation	4	88	4	(95d)
				forw.ext.	8	176	0	1s
				back.ext.	8	176	0	1s
10	220	0	12m	generation	4	88	4	(95d)
				forw.ext.	10	220	0	12m
				generation	6	132	0	(115d)
				back.ext.	10	220	0	-
12	264	0	-	generation	6	132	0	(115d)
				forw.ext.	12	264	0	-
				back.ext.	12	264	0	-

6 Improved bounds

In this section, we report on our results for both differential and linear trail cores. The improved bounds resulting from our search are reported in Table 1. Details on our search, as the number of (canonical) trail cores found and the execution time for the different steps, are reported in Table 3 for differential trails and in Table 4 for linear trails.

We will integrate our optimizations into the public software XooTools [DHVV18b]. All our experiments are run on a server equipped with an AMD EPYC 7552 48-Core Processor @2.2GHz. We exploited the multicore architecture to run some of our experiments in parallel. However, execution times are reported as single-core time.

As starting point, we consider the set of all 2-round trail cores with weight $\mathrm{w}(a_1) + \mathrm{w}(b_1) < 44$ and all 3-round trail cores below weight 54. The former are generated using XooTools. The latter were produced at the end of 2021 [The21] and can be found in the XooTools repository [DHVV18b].

Table 4: Details on the generation of r-round canonical linear trail cores with weight below T_r. Timings are rounded to the closest integer and in the fourth column they are meant to be incremental. Timings between parenthesis means that the same computation has been already done in a previous step, so it is counted only once. - means that the step is not performed since the starting set is empty. * the time with XooTools on a desktop PC equipped with an Intel© Core™ i5-6500 CPU.

search space				details				
r	T_r	# cores found	time	step	r	T_r	# cores found	time
4	88	3	94d	generation	2	44	2,983,073,628	344h
				forw.ext.	3	86	2,456,384	27h
				forw.ext.	4	88	2	3m
				back.ext.	3	86	274,104	1878h
				back.ext.	4	88	1	1m
5	98	0	21d	generation	2	44	2,983,073,628	(344h)
				forw.ext.	5	98	0	28h
				generation	3	54	204	480h*
				back.ext.	5	98	0	1s
6	132	0	23s	generation	3	54	204	(480h*)
				forw.ext.	6	132	0	3s
				back.ext.	6	132	0	2s
				generation	2	44	2,983,073,628	(344h)
				ext. from start	6	132	0	(27h+) 17s
				ext. from middle	6	132	0	(27h+) 1s
				ext. from end	6	132	0	(1878h)
8	176	0	4s	generation	4	88	3	(94d)
				forw.ext.	8	176	0	1s
				back.ext.	8	176	0	3s
10	220	0	2m	generation	4	88	3	(94d)
				forw.ext.	10	220	0	2m
				generation	6	132	0	(114d)
				back.ext.	10	220	0	-
12	264	0	-	generation	6	132	0	(114d)
				forw.ext.	12	264	0	-
				back.ext.	12	264	0	-

6.1 Bounds on 4 rounds

We generated all 4-round differential and linear trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) < 88$. We found 2 differential trail cores of weight 80, one of weight 82, and one of weight 86. We also found 2 linear trail cores of weight 80 and one of weight 82. This finally solves the open problem of proving a tight bound for 4-round trails in Xoodoo. The trails of weight 80 that we found have some interesting properties that we will discuss in Section 8.

To perform our search, we observe that any 4-round trail core with weight $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) < 88$ has $\mathrm{w}(a_1) + \mathrm{w}(b_1) < 44$ or $\mathrm{w}(b_2) + \mathrm{w}(b_3) < 44$, otherwise their sum would give at least 88. Therefore, all 4-round trail cores with weight smaller than 88 can be generated by extending all 2-round trail cores with $\mathrm{w}(a) + \mathrm{w}(b) < 44$ either in the forward or backward direction. Extension is performed by first extending all 2-round trail

cores to 3 rounds below 86 (because the remaining round will contribute at least 2 to the weight) and then by extending the obtained 3-round trail cores to 4 rounds below weight 88. This search took in total 95 CPU days.

6.2 Bounds on 5 rounds

We scanned the space of all 5-round differential and linear trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) < 98$. We found that both spaces are empty. It follows that any 5-round trail has weight at least 98.

We split our search space into two sub-spaces. The first sub-space contains the 5-round trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) < 44$. This space can be covered by extending all 2-round trail cores with $\mathrm{w}(a) + \mathrm{w}(b) < 44$ in the forward direction to 5 rounds below weight 98. The second sub-space contains the 5-round trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) \geq 44$. For such trail cores, $\mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) < 54$ otherwise their weight would be at least 98. We can thus cover this space by extending all 3-round trail cores with weight below 54 in the backward direction to 5 rounds below weight 98. This search took 21 CPU days, considering that we reused the 2-round trail cores with $\mathrm{w}(a) + \mathrm{w}(b) < 44$ already generated.

6.3 Bounds on 6 rounds

We scanned the space of all 6-round differential and linear trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_6) < 132$. We found that such spaces are empty. It follows that any 6-round trail has weight at least 132.

We split our search space into two sub-spaces. The first sub-space contains the 6-round trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) < 54$ or $\mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 54$. Such trail cores can be generated by extending all 3-round trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) < 54$ in the forward and backward direction by three rounds below T_6.

The second sub-space contains the 6-round trail cores with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) \geq 54$ and $\mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 54$. We further split this space into three parts. In fact, we observe that any 6-round trail core of weight $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132$ has $\mathrm{w}(a_1) + \mathrm{w}(b_1) < 44$, or $\mathrm{w}(b_2) + \mathrm{w}(b_3) < 44$, or $\mathrm{w}(b_4) + \mathrm{w}(b_5) < 44$. Otherwise their sum would give at least 132. Any such 6-round trail core can be built by starting from a 2-round trail core with $\mathrm{w}(a) + \mathrm{w}(b) < 44$ placed at the beginning, or in the middle, or at the end of the trail. When such 2-round trail core is at the beginning, we extend it by four rounds in the forward direction. When it is in the middle, we extend it by two rounds in the forward direction and two rounds in the backward direction. When it is at the end, we extend it by four rounds in the backward direction.

Notice that we could start from any of these three positions, but we tried to prioritize forward extension over backward extension, since it is cheaper. When we extended a trail core, we performed extension by one round at a time, limiting the weight up to which we perform extension, taking into account the minimal weight contribution of the remaining rounds. A detailed description of the steps performed is given in the following.

1 **Starting from** $\mathrm{w}(a_1) + \mathrm{w}(b_1) < 44$ **and forward extension by 4 rounds:**

- 1a First, we extend to 3 rounds in the forward direction with $54 \leq \mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) < 132 - 54 = 78$, because we are in the case where $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) \geq 54$ and $\mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 54$.
- 1b Then, we extend the obtained 3-round trail cores to 4 rounds in the forward direction with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) < 132 - 8 = 124$, because we know that $\mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 8$.
- 1c Then, we extend the obtained 4-round trail cores to 5 rounds in the forward direction with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) < 132 - 2 = 130$, because $\mathrm{w}(b_5) \geq 2$.

1d Finally, we extend the obtained 5-round trail cores to 6 rounds in the forward direction with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132$.

2 **Starting from** $\mathrm{w}(a_3) + \mathrm{w}(b_3) < 44$ **and forward extension by 2 rounds and backward extension by 2 rounds**: we can assume that $\mathrm{w}(a_1) + \mathrm{w}(b_1) \geq 44$ because the opposite is already covered in the previous step.

2a First, we extend to 3 rounds in the forward direction with $\mathrm{w}(a_3)+\mathrm{w}(b_3)+\mathrm{w}(b_4) < 132 - 44 - 2 = 86$, because $\mathrm{w}(a_1) + \mathrm{w}(b_1) \geq 44$ and $\mathrm{w}(b_5) \geq 2$.

2b Then, we extend the obtained 3-round trail cores to 4 rounds in the forward direction with $\mathrm{w}(a_3) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132 - 44 = 88$, because $\mathrm{w}(a_1) + \mathrm{w}(b_1) \geq 44$. Among the trail cores found, we consider only those satisfying $\mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 54$. Moreover, we consider only the lightest trail for a given state a_3, since any other will lead to a trail core with higher weight once extended.

2c Then, we extend the remaining 4-round trail cores to 5 rounds in the backward direction with $\mathrm{w}(a_2)+\mathrm{w}(b_2)+\mathrm{w}(b_3)+\mathrm{w}(b_4)+\mathrm{w}(b_5) < 132-2$ because $\mathrm{w}(a_1) \geq 2$.

2d Finally, we extend the obtained 5-round trail cores to 6 rounds in the backward direction with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132$.

3 **Starting from** $\mathrm{w}(a_5) + \mathrm{w}(b_5) < 44$ **and backward extension by 4 rounds:** we can assume that $\mathrm{w}(a_1) + \mathrm{w}(b_1) \geq 44$ and $\mathrm{w}(b_2) + \mathrm{w}(b_3) \geq 44$ because the opposite is already covered in the previous steps.

3a First, we extend to 3 rounds in the backward direction with $54 \leq \mathrm{w}(a_4) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132 - 54 = 78$, because $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) \geq 54$ and $\mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 54$.

3b Then, we extend the obtained 3-round trail cores to 4 rounds in the backward direction with $\mathrm{w}(a_3) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132 - 44$, because $\mathrm{w}(a_1) + \mathrm{w}(b_1) \geq 44$. Among the trail cores found, we consider only those satisfying $\mathrm{w}(a_3) + \mathrm{w}(b_3) \geq 44$.

3c Then, we extend the remaining 4-round trail cores to 5 rounds in the backward direction with $\mathrm{w}(a_2)+\mathrm{w}(b_2)+\mathrm{w}(b_3)+\mathrm{w}(b_4)+\mathrm{w}(b_5) < 132-2$, because $\mathrm{w}(a_1) \geq 2$.

3d Finally, we extend the obtained 5-round trail cores to 6 rounds in the backward direction with $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) < 132$.

We observe that the set of trail cores in step 1a is a subset of the set of trail cores in step 2a. Such set is already covered when we prove bounds for 4 rounds. Also, the set of trail cores in steps 2b, 3a, and 3b are already covered during that search. Therefore, we did not need to perform these steps, but we extracted the trail cores satisfying the required properties from the previously built sets.

Both in the differential and linear case, step 1c did not output any trail core. Therefore, we did not perform step 1d. Similarly, we did not need to perform step 3c and 3d since the output of step 3b was empty. All in all, this search took a few seconds for both the differential and linear case.

6.4 Bounds on 8 rounds

We generated all 8-round differential and linear trail cores with weight below 176 and found that there are no such trail cores. Therefore, 176 defines a (non-tight) lower bound on the weight of any 8-round trail.

To perform our search, we observe that a 8-round trail core with weight below 176 has $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) + \mathrm{w}(b_3) < 88$ or $\mathrm{w}(b_4) + \mathrm{w}(b_5) + \mathrm{w}(b_6) + \mathrm{w}(b_7) < 88$, otherwise their

sum would give at least 176. Therefore, all 8-round trail core with weight smaller than 176 can be generated by extending all 4-round trail cores with weight below 88 by four rounds in the forward or backward direction. Since there are only four 4-round differential trail cores and three 4-round linear trail cores with weight below 88, extension to 8 rounds took a few seconds in each case.

6.5 Bounds on 10 rounds

We scanned the space of all 10-round differential and linear trail cores with weight below 220. We found that both spaces are empty. It follows that any 10-round trail has weight at least 220.

We split our search space into two sub-spaces. The first sub-space contains the 10-round trail cores with $\mathrm{w}(a_1)+\mathrm{w}(b_1)+\mathrm{w}(b_2)+\mathrm{w}(b_3) < 88$. This space can be covered by extending all 4-round trail cores with weight below 88 in the forward direction to 10 rounds below weight 220. The second sub-space contains the 10-round trail cores with $\mathrm{w}(a_1)+\mathrm{w}(b_1)+\mathrm{w}(b_2)+\mathrm{w}(b_3) \geq 88$. For such trail cores, $\mathrm{w}(b_4)+\mathrm{w}(b_5)+\mathrm{w}(b_6)+\mathrm{w}(b_7)+\mathrm{w}(b_8)+\mathrm{w}(b_9) < 132$ otherwise their weight would be at least 220. We can thus cover this space by extending all 6-round trail cores with weight below 132 in the backward direction to 10 rounds below weight 220. However, there are no such 6-round trail cores. For this reason and given that we already generated all 4-round trail cores with weight below 88, proving the bound on 10 rounds took only 12 minutes for differential trails, and 2 minutes for linear trails.

6.6 Bounds on 12 rounds

Any 12-round trail core with weight below 264 has $\mathrm{w}(a_1)+\mathrm{w}(b_1)+\mathrm{w}(b_2)+\mathrm{w}(b_3)+\mathrm{w}(b_4)+\mathrm{w}(b_6) < 132$ or $\mathrm{w}(b_7)+\mathrm{w}(b_8)+\mathrm{w}(b_9)+\mathrm{w}(b_{10})+\mathrm{w}(b_{11})+\mathrm{w}(b_{12}) < 132$. Since there are no 6-round trails with weight below 132, we can conclude that there are no 12-round trails with weight below 264.

7 Trail core distributions

In Figure 7, 8, and 9, we report on all canonical trail cores that we found for 2, 3, and 4 rounds, respectively. For 2 rounds, we found a total of 2,983,444,980 differential trail cores and 2,983,073,628 linear trail cores with weight below 44. It is worth noticing that these numbers are close to each other within 0.05%. Similarly, for 3 and 4 rounds we can notice that for each weight, the difference between the number of linear and differential trail cores with that weight is only 1 or 2. This is a consequence of the choice of 3-bit χ over, for instance, 5-bit χ as in Keccak-p.

8 Staircase trail cores

In our search we found four families of (canonical) trail cores, two differential and two linear, that have interesting properties. Due to their weight profiles, we call them *staircase* trail cores. Each family of staircase trail cores has a member for each length. For r rounds, they have weight $\binom{r}{2}8$. Put differently, extending an $r-1$-round staircase trail core by one round will add weight $8r$. For two staircase trail cores this extension can only be done forward and these have weight profile $(8, 16, 24, 32, \ldots)$. We call these *upstairs* trail cores. For the two other staircase trail cores this extension can only be done backward and these have weight profile $(\ldots, 32, 24, 16, 8)$ and we call them *downstairs* trail cores.

We illustrate the differential staircase trail cores in Figure 10 and Figure 11.

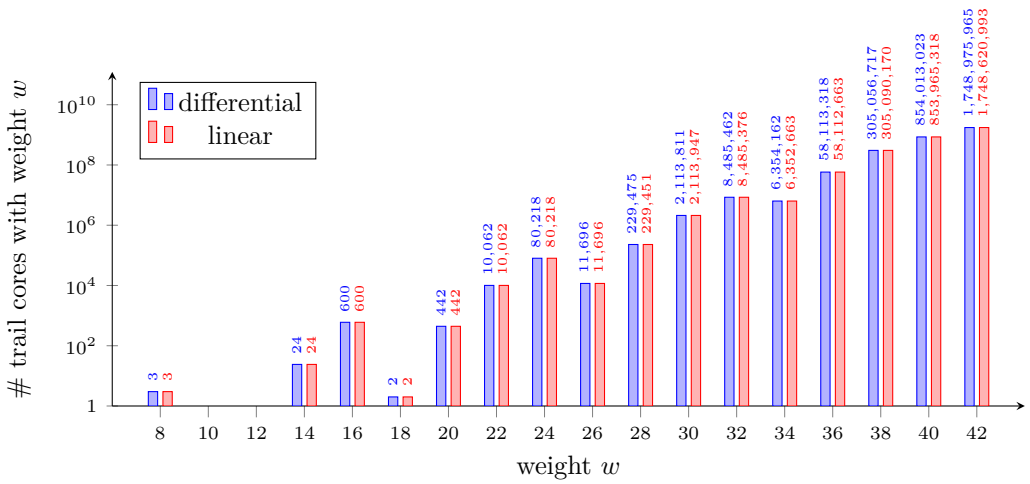


Figure 7: Number of canonical 2-round trail cores with weight w.

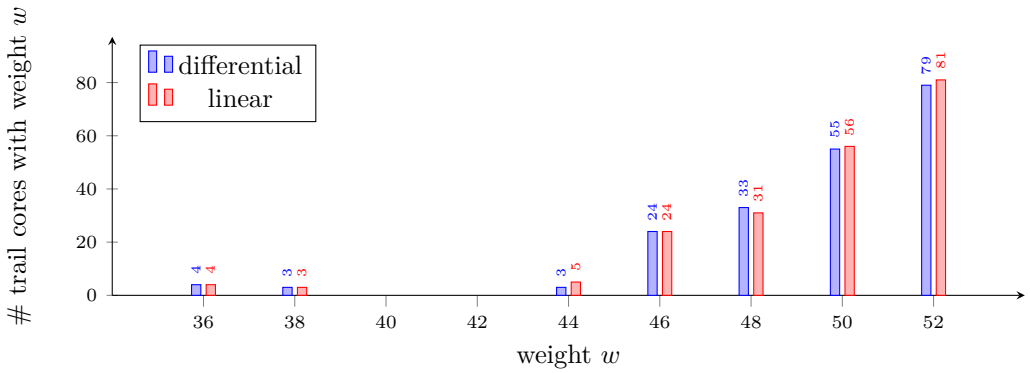


Figure 8: Number of canonical 3-round trail cores with weight w.

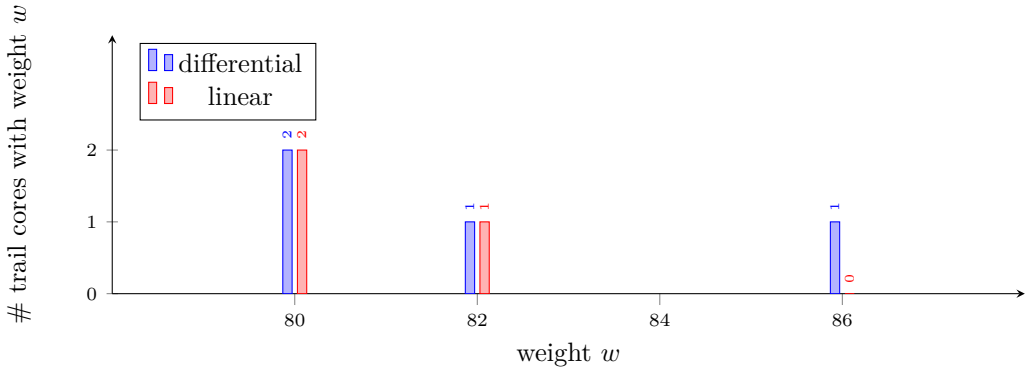


Figure 9: Number of canonical 4-round trail cores with weight w.

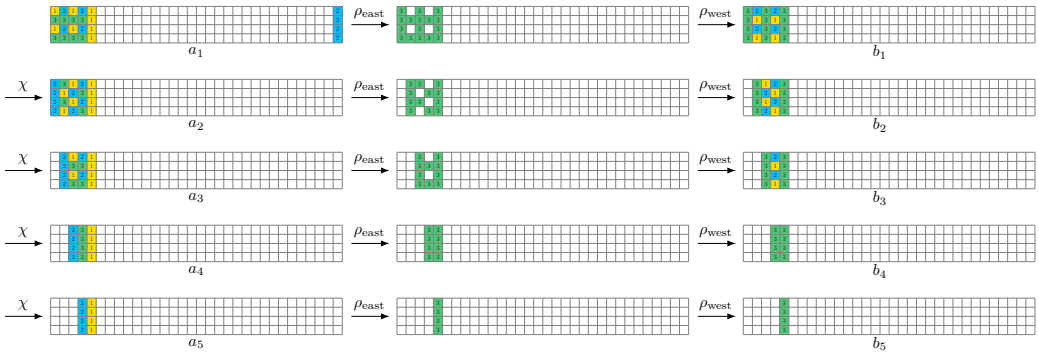


Figure 10: Differential staircase trail that goes downstairs

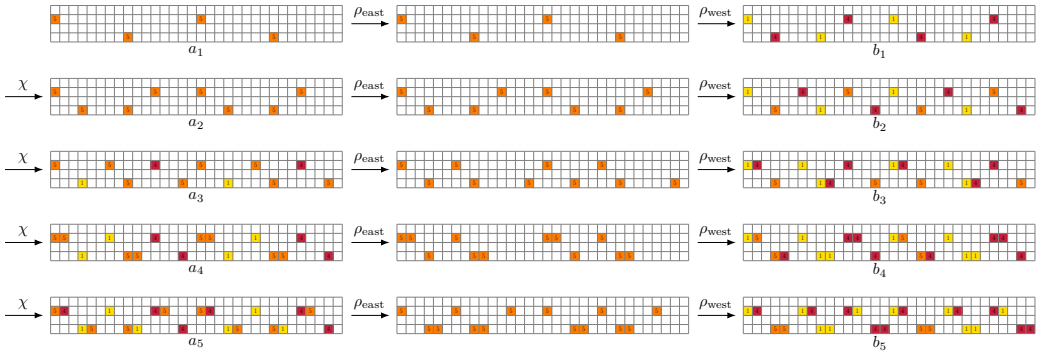


Figure 11: Differential staircase trail that goes upstairs

It can be seen that both depicted staircase trail cores are in the θ-kernel: the states between ρ_{east} and ρ_{west} have always an even number of active bits in each column. For such trails, θ behaves as the identity and we therefore omit it from our figures.

Mask propagation and difference propagation through the χ layer of Xoodoo is governed by the same laws and masks and differences also propagate the same through bit shuffles. As a consequence, the differential trail cores in Figure 10 and Figure 11 can also be interpreted as linear trail cores. So, despite the fact that we have four staircase trail cores, we only have two different staircase trail structures.

We will discuss the downstairs trail core in Figure 10 because it is easiest to understand thanks to its simple geometry. However, the upstairs trail core in Figure 11 is structurally very similar to it.

The states of the trail core are only active in two planes: plane 0 and 1, limiting the differences in the active columns to $1, 2$ and 3, where 3 is in the kernel. The seed of the trail core is in the last round: we see that the state at θ there has 4 active columns in the same slice. ρ_{west} leaves plane 0 where it is and shifts plane 1 one position up in the direction of the x-axis, vertical in the figure. So the pattern is invariant under $\rho_{\text{west}} \cdot {\rho_{\text{east}}}^{-1}$ also leaves plane 0 where it is and shifts plane 1 one position left in the direction of the z-axis, horizontal in the figure. This leads to 8 active columns in a_5, each with a single active bit. This state is compatible with b_4, that is again in the kernel: it consists of two copies of the structure of b_5. This state is invariant under ρ_{west} and θ. ${\rho_{\text{east}}}^{-1}$ shifts plane 1 and this results in a three-slice state: one slice with all bits in plane 0 active, one slice with all bits in plane 1 active and in between one slice with all bits in both planes active. Through χ the two outer slices propagate to in-kernel patterns and the middle one to an alternating pattern of active bits in plane 0 and 1. This value of b_3 propagates to

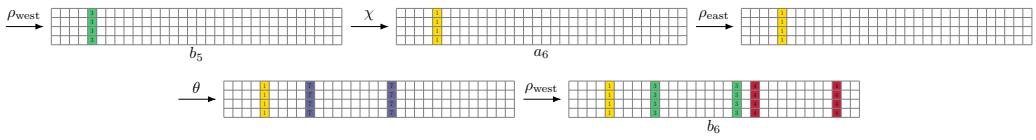


Figure 12: Forward extension of the differential staircase trail that goes downstairs

an in-kernel pattern through ${\rho_{\text{west}}}^{-1}$ as it aligns the alternating pattern in two columns. The effect of ${\rho_{\text{east}}}^{-1}$ on this state results in a state similar to a_4 but now there are two middle slices with alternating structures. Through χ the two outer slices propagate again to in-kernel patterns and the middle ones to alternating patterns of active bits in plane 0 and 1.

In general, the weight profile can be understood as follows. In any round except the last, the state at θ consists of a tablecloth pattern boarded by two fully active slices. ρ_{west} moves plane 1 one position up resulting in a different tablecloth pattern of the same width. ${\rho_{\text{east}}}^{-1}$ moves plane 1 one position to the left resulting in yet another tablecloth pattern that is one slice wider. The tablecloth patterns at b_i are compatible with the ones as a_{i+1} through χ as the compatible pairs are $(1,1),(1,3),(3,1),(2,2),(2,3),(3,2)$. Note that $(3,3)$ is not a compatible pair. So with each round the pattern gets one slice wider, increasing the weight by 8. From b_5 one cannot extend this trail inside the kernel anymore: after χ only a single slice will be active and ρ_{east} will move bits in the same column to columns in different slices. Still, as illustrated in Figure 12, forward extension yields in a relatively light difference with weight 40, resulting in a weight profile $(\ldots, 48, 40, 32, 24, 16, 8, 40)$. This is the weight profile that determines the upper bounds to the minimum trail weight reported in Table 1.

The upstairs trail core in Figure 11 looks very different. It has states that are only active in the cross sections of plane 0 and plane 2 and sheet 0 and 2, its weight profile is increasing and it has a different symmetry. But on the other hand it is also very similar and can be explained in a similar way: it is in the kernel in all rounds and exhibits a regularly expanding pattern over the rounds. As opposed to the downstairs trail, extending this trail from the seed with weight 8 in the opposite direction does not lead to a relatively light difference. This is due to the fact that this extension requires the dense inverse of θ.

9 Conclusions

This work further develops the trail core tree search technique to prove bounds on differential and linear trails. In particular, we presented new methods to make trail extension in Xoodoo more efficient. To this end, we further exploited the propagation properties of χ and θ. Once coupled with the 2-round trail core generation techniques presented in [DHVV18a], these new methods allowed us to scan in practical time larger spaces of trails than before. As a direct consequence, we could improve over known bounds for differential and linear trails. We prove a tight bound of 80 for the weight of 4-round trails, both linear and differential. For 6 and 12 rounds, we proved bounds of 132 and 264, respectively.

Acknowledgments

Joan Daemen is supported by the European Research Council under the ERC advanced grant agreement under grant ERC-2017-ADG Nr. 788980 ESCADA.
Silvia Mella is supported by PROACT (NWA.1215.18.014), a project financed by the Netherlands Organisation for Scientific Research (NWO); by the European Commission

through the ERC Starting Grant 805031 (EPOQUE); by the Cryptography Research Center of the Technology Innovation Institute (TII) under the TII-Radboud project *Evaluation and Implementation of Lightweight Cryptographic Primitives and Protocols*.

References

[BDH+17] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer. Farfalle: parallel permutation-based cryptography. *IACR Trans. Symmetric Cryptol.*, 2017(4):1–38, 2017.

[BDPV11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On alignment in Keccak. In *ECRYPT II Hash Workshop*, 2011.

[BFR22] Christina Boura, Margot Funk, and Yann Rotella. Differential analysis of the ternary hash function troika. In *Selected Areas in Cryptography - SAC 2022*, 2022.

[Dae95] J. Daemen. *Cipher and hash function design strategies based on linear and differential cryptanalysis, PhD Thesis*. K.U.Leuven, 1995.

[DHP+20] J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Trans. Symmetric Cryptol.*, 2020(S1):60–87, 2020.

[DHP+21] J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer. Xoodyak, a lightweight cryptographic scheme. Submission to NIST Lightweight Cryptography Standardization Process (round 3), May 2021. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/xoodyak-spec-final.pdf.

[DHVV18a] J. Daemen, S. Hoffert, G. Van Assche, and R. Van Keer. The design of Xoodoo and Xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

[DHVV18b] J. Daemen, S. Hoffert, G. Van Assche, and R. Van Keer. XooTools software. https://github.com/XoodooTeam/Xoodoo, 2018.

[DMA22] Joan Daemen, Silvia Mella, and Gilles Van Assche. Tighter trail bounds for xoodoo. *IACR Cryptol. ePrint Arch.*, page 1088, 2022.

[DV12] J. Daemen and G. Van Assche. Differential propagation analysis of Keccak. In A. Canteaut, editor, *Fast Software Encryption (FSE) 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 422–441. Springer, 2012.

[FRD23] Jonathan Fuchs, Yann Rotella, and Joan Daemen. On the security of keyed hashing based on public permutations. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 607–627. Springer, 2023.

[HMMD22] Solane El Hirch, Silvia Mella, Alireza Mehrdad, and Joan Daemen. Improved differential and linear trail bounds for ASCON. *IACR Trans. Symmetric Cryptol.*, 2022(4):145–178, 2022.

[MDV17] S. Mella, J. Daemen, and G. Van Assche. New techniques for trail bounds and application to differential trails in Keccak. *IACR Trans. Symmetric Cryptol.*, 2017(1):329–357, 2017.

[MMGD22] Alireza Mehrdad, Silvia Mella, Lorenzo Grassi, and Joan Daemen. Differential trail search in cryptographic primitives with big-circle chi: Application to subterranean. *IACR Trans. Symmetric Cryptol.*, 2022(2):253–288, 2022.

[Nat21] National Institute of Standards and Technology. Status report on the second round of the nist lightweight cryptography standardization process, 2021.

[SD18] K. Stoffelen and J. Daemen. Column parity mixers. *IACR Trans. Symmetric Cryptol.*, 2018(1):126–159, 2018.

[The21] The Keccak Team. Updated bounds on differential and linear trails in Xoodoo. https://keccak.team/2021/updated_bounds_xoodoo.html, 2021.