Tighter trail bounds for Xoodoo

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

[1] Radboud University, Nijmegen, The Netherlands
[2] STMicroelectronics, Diegem, Belgium

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 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 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 ant that allowed us 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.

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

1 Introduction

The Xoodoo cryptographic permutation is the core of Xoodyak and Xoofff [DHVV18a, DHP+19, DHP+20]. Xoodyak is a versatile cryptographic object based on the Cyclist construction [DHVV18b], intended for more 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, an instance of Farfalle [BDH+17, DHVV18b, DHVV18a]. 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, the resulting permutation shall be fast enough and strong enough once plugged in a given construction. We choose the number of rounds 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 must prevent high-probability differential trails or high-correlation linear trails (or equivalently, low-weight 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

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	168
8	148	[DHP+20]	-		176	288
10	188	[DHP+20]	-		220	440
12	222	[DHP+20]	-		264	624

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].

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, based on scanning the space of all trails with weight below a given threshold T_r and for a given number of rounds r. If such space is non-empty, then the trail with the smallest weight defines a tight bound on the weight of all r-round trails. On the contrary, T_r defines a non-tight bound. 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. This is why one can usually prove only non-tight bounds.

In this work, we present a number of methods that allow us to make our search more efficient and, as a consequence, target higher values of T_r in a time that is still practical. Thanks to such methods, we can improve over known bounds for Xoodoo for different numbers of rounds. Notably, we prove tight bounds for the weight of linear and differential trails over 4 rounds. For 6 and 12 rounds, we prove for the first time (non-tight) bounds beyond 128 and 256, respectively. We report our improved bounds in Table 1.

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 use such trails to provide upper bounds, which are reported in Table 1.

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, z) coordinates. The Xoodoo state is illustrated in Figure 1 with its different parts.

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

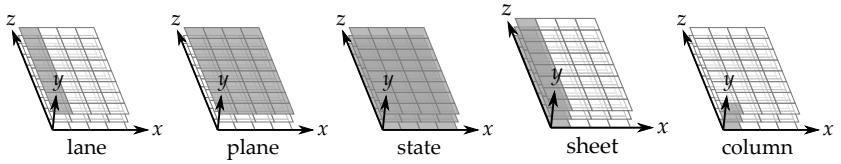

Figure 1: Toy version of the Xoodoo state, with lanes reduced to 8 bits, and different parts of the state highlighted.

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

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.

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 alignment [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 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

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

an involution and its propagation properties hold also for its inverse.

Differences 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. 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 state differential is twice the number of non-zero columns 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 masks.

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 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], trails are 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. Then, we recall the general strategy used in [DV12, MDV17, DHVV18a, MMGD22] to perform trail search that we 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 call λ the *linear layer*. Therefore, the re-phased round function becomes $\chi \circ \iota \circ \lambda$.

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: $(a_0, b_0, a_1, b_1 \ldots, b_{r-1}, a_r)$.

Since λ is linear, $b_i = \lambda(a_i)$ and the restriction weight of the trail only depends on the differentials over χ: $\sum_i \mathrm{w_r}(b_{i-1}, a_i)$. We denote the sequence $\mathrm{w_r}(b_{i-1}, a_i)$ for $i = 1$ to r as the weight profile of the trail.

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 $b_1, \ldots, b_{r-1}$: $\mathrm{w_r}(a_1) + \sum_{1 \le i < r} \mathrm{w_r}(b_i)$ and is thus independent of the value of a_0, b_0, and a_r. As in [DV12], we call *differential trail core* the sequence:

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

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}. 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. In our analysis, we can limit ourselves to bound the weight of trail cores.

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 $C(a_i, a_{i+1})$. The correlation weight of trail is the sum of the correlation weight of its round correlations.

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 correlation (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.

3.3 Trail search strategy

Given the similarities between the study of differential and linear trails, a unified notation is introduced in [DHVV18a]:

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

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 is illustrated in Figure 2.

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.

To generate all r-round trail cores with weight below a given target T_r, the search starts by generating all $\lfloor r/2 \rfloor$-round trail cores with weight below $\lfloor T_r/2 \rfloor$ and continues by extending them in the forward and backward direction to reach the desired number of rounds. This reasoning can be iterated until reaching the shortest trail cores which are 2-round trail cores.

A 2-round trail core is fully specified by a state a and its weight is given by $\mathrm{w}(a) + \mathrm{w}(\lambda^\star(a))$. States a's such that $\mathrm{w}(a) + \mathrm{w}(\lambda^\star(a)) < T_2$ are iteratively generated by using

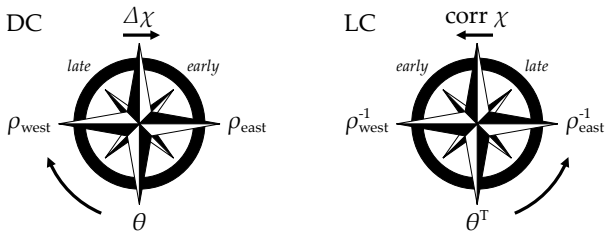

Figure 2: Conventions for differential (DC) and linear (LC) trails in the round function.

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] where all details can be found.

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 trails to extend and the number of active columns in the trail. In fact, for any state b at the input of χ there are $2^{\mathrm{w}(b)}$ compatible states at the output of χ. The more active columns in b, the more compatible states there exist. In the next section we recall how trail extension works for Xoodoo and how we optimized it in order to achieve higher target weights.

4 Optimizing the trail extension

We start with describing how trail extension can be implemented as a tree search. We then explain how to lower bound the weight of the states in this tree. Finally, we describe optimization techniques so as to limit the search.

4.1 Affine space of compatible states

Extending a trail means looking for all the possible states that are compatible through the round function 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 $\mathcal{A}(b) = O + \langle V_1, \ldots, V_w \rangle$, 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 $\mathcal{B}(a)$.

4.2 Extension as a tree-based search

The affine space $\mathcal{A}(b)$ or $\mathcal{B}(a) = O + \langle V_1, \ldots, V_w \rangle$ can be scanned through 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, an order relation among basis vectors can be introduced; 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 $i_{k+1} \in \{i_k + 1, \ldots, w\}$. Alternatively, the parent of node $O + V_{i_1} + \ldots + V_{i_{k-1}} + V_{i_k}$ is $O + V_{i_1} + \ldots + V_{i_{k-1}}$. To give an example, let us assume the affine space has dimension 10 and consider the node $O + V_2 + V_3 + V_6$. The parent of this node is $O + V_2 + V_3$ and its children are $O + V_2 + V_3 + V_6 + V_7$, $O + V_2 + V_3 + V_6 + V_8$, $O + V_2 + V_3 + V_6 + V_9$ and $O + V_2 + V_3 + V_6 + V_{10}$.

4.3 Limiting the search using the far view

Scanning the entire affine space takes 2^w iterations and becomes prohibitively expensive if a or b has a large 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, $\rho_{early}(\mathcal{A}(b))$, $\theta^\star(\rho_{early}(\mathcal{A}(b)))$. This goes up to the input to the next χ with $\mathcal{B}(b) = \lambda^\star(\mathcal{A}(b)) = O^{\text{far}} + \langle V_1^{\text{far}}, V_2^{\text{far}}, \ldots, V_w^{\text{far}} \rangle$ with $O^{\text{far}} = \lambda^\star(O^{\text{near}})$ and $V_i^{\text{far}} = \lambda^\star(V_i^{\text{near}})$ if $\mathcal{A}(b) = O^{\text{near}} + \langle V_1^{\text{near}}, \ldots, V_w^{\text{near}} \rangle$. And, similarly for the backward extension, we can transpose $\mathcal{B}(a)$ to $\rho_{late}^{-1}(\mathcal{B}(a))$, $\theta^{\star-1}(\rho_{late}^{-1}(\mathcal{B}(a)))$, and $\mathcal{A}(a) = \lambda^{\star-1}(\mathcal{B}(a))$.

All the elements of this space 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 $i < j$. Using the far view, we can start pruning the tree when their weight goes above the target.

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* the set of bit positions that does not change when going through its descendants. Note that the stable bits depend only on the index of the last basis element encountered, in this case i_k.

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. Here S_i is the stability mask of a node with last basis element has index i; S_0 is the stable bit mask for the entire basis and S_w is the all-one mask. The expression for S_i is as follows:

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

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 cannot become passive in its descendants. In other words, if a bit is active in $N \wedge S_{i_k}$, it will be active in any descendant of N and thus any active column in $N \wedge S_{i_k}$ will be active in all descendants of N. It follows that the weight of any descendant of N has weight at least $\mathrm{w}(N \wedge S_{i_k})$; this is the score of the subtree under N. Similarly, the score of the root node is $\mathrm{w}(O \wedge S_0)$.

4.5 Triangularization

The stability masks S_0 through S_w depend on the basis $\langle V_1, V_2, \ldots, V_w \rangle$ in the far view. To limit the search as much as possible, intuitively, 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 potentially prune entire subtrees early.

To this end, a technique consists in triangularizing the basis $\langle V_1, V_2, \ldots, V_w \rangle$. With some abuse of notation, let us denote the triangularized basis again as $\langle V_1, V_2, \ldots, V_w \rangle$. Let $p_i = (x_i, y_i, z_i)$ be the coordinate of the first active bit in V_i, where "first" refers to some arbirary ordering, for instance the lexicographic order relation over x, y, z. By construction, all bits strictly smaller than p_{i+1} are passive in all vectors in $V_{i+1}, \ldots, V_w$, which are therefore included in the stability mask S_i.

Actually, the implementation of the trail extension in [DHVV18a] is suboptimal. It computes the score as the weight over all bits strictly smaller than p_{i+1}, instead of the entire stability mask. Just by better formalizing the idea and using the stability mask as defined in (1), we could speed up backward extension by a factor about 100 for the search spaces in [DHVV18a].

4.6 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.

Computing a tight score is more difficult with backward extension than with forward extension. 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 output of $\theta^\star$ 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. Basis elements in the far view can therefore potentially turn a lot of active bits back to passive, and without an adequate treatment the number of stable bits grows slowly.

A simple triangularization can help, but we can do better and exploit the special form of the basis to stabilize more bits at a time. For this purpose, we are going to look at columns before and after $\theta^\star$, and we call this the *θ-view* to avoid confusion with the columns in the near and the far views.

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 columns of $E(x)$ are either completely active or completely passive. So, the basis elements of $\theta^{\star-1}(\rho_{late}^{-1}(\mathcal{B}(a)))$ are one or two active bits plus a number of completely active columns. The idea is therefore to triangularize in a way that takes a whole active column in the θ-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}.

Let us describe the procedure more precisely. First, we let $\mathcal{E} = E(\rho_{late}^{-1}(\mathcal{B}))$ be the affine space representing the θ-effect of $\mathcal{B}$. In other words, $\mathcal{E} = O^E + \langle V_1^E, \ldots, V_w^E \rangle$ with $O^E = E(\rho_{late}^{-1}(O^{\text{near}}))$ and $V_i^E = E(\rho_{late}^{-1}(V_i^{\text{near}}))$ if $\mathcal{B} = O^{\text{near}} + \langle V_1^{\text{near}}, \ldots, V_w^{\text{near}} \rangle$. Then, we triangularize $\mathcal{E}$ and modify the representation of $\mathcal{B}$ accordingly: Every time we add V_i^E to V_j^E in $\mathcal{E}$, we add V_i^{near} to V_j^{near} in $\mathcal{B}$. Finally, we transpose $\mathcal{B}$ to $\mathcal{A} = \lambda^{\star-1}(\mathcal{B})$.

There are two refinements to this method, both related to the order in which we consider the columns (x, z) when choosing a pivot.

First, in the θ-view, i.e., when looking at $\rho_{late}^{-1}(\mathcal{B})$, 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 $\rho_{late}^{-1}(\mathcal{B})$, and the *no-go* columns are the remaining ones. When choosing a pivot in the triangularization of $\mathcal{E}$, 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 is as follows. Assume that (x, z) is a go column and that it is taken as a pivot in the triangularization of $\mathcal{E}$. Because it is a go column, we have $\rho_{late}^{-1}(b)[x, z] = (0, 0, 0)$ for all $b \in \mathcal{B}$. Since it is taken as pivot, it means that $E(\rho_{late}^{-1}(V_i^{\text{near}}))[x, z] = 1$ and $\theta^{\star-1}(\rho_{late}^{-1}(V_i^{\text{near}}))[x, z] = (1, 1, 1)$ for some basis vector

V_i^{near} of $\mathcal{B}$, 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, 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 $E(\rho_{late}^{-1}(V_i^{\text{near}}))[x, z] = 1$ 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.

Second, when choosing a pivot column, we follow a diagonal pattern in the θ-view. Instead, assume that we follow a horizontal pattern, i.e., after pivot column (x, z), we consider pivot column $(x, z + 1)$. 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 $\rho_{early}^{-1} = \rho_{\text{east}}^{-1}$ in the case of DC. 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. To limit this effect, when we look for a pivot column we loop over $(x, z) = (k \bmod 4, (k + \lfloor k/32 \rfloor) \bmod 32)$ for $k \in \mathbb{Z}_{128}$.

5 Improved bounds

In this section, we report on our experiments 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 trail cores found and the execution time for the different steps, are reported in Table 3 for the differential case and in Table 4 for the linear case.

We will integrate our optimizations into the public software XooTools [DHVV18c]. 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 generated the set of all 2-round trail cores with weight $\mathrm{w}(a_1) + \mathrm{w}(b_1) < 44$. We found 2,983,444,980 differential trail cores and 2,983,073,628 linear trail cores. It is worth noticing that these numbers are close to each other within 0.05%. This is a consequence of the choice of 3-bit χ over, for instance, 5-bit χ as in Keccak-p.

We also used all 3-round trail cores below weight 54, that are 201 in the differential case and 204 in the linear case. Such trail cores were produced at the end of 2021 [The21] and can be found in the XooTools repository [DHVV18c].

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 6.

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

Table 3: Details on the generation of r-round 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's 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	-

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.

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

Table 4: Details on the generation of r-round 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's 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	-

reused the 2-round trail cores with $\mathrm{w}(a) + \mathrm{w}(b) < 44$ already generated.

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:**
   (a) 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 $\mathrm{w}(b_3) + \mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 54$.
   (b) 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 $\mathrm{w}(b_4) + \mathrm{w}(b_5) \geq 8$.
   (c) 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 $\mathrm{w}(b_5) \geq 2$.
   (d) 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.
   (a) 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 $\mathrm{w}(b_5) \geq 2$.
   (b) 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 value of a_3, since any other will lead to a trail core with higher weight once extended.
   (c) 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 $\mathrm{w}(a_1) \geq 2$.
   (d) 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.
   (a) 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 $\mathrm{w}(a_1) + \mathrm{w}(b_1) + \mathrm{w}(b_2) \geq 54$.
   (b) 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 $\mathrm{w}(a_3) + \mathrm{w}(b_3) \geq 44$.

(c) 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 $\mathrm{w}(a_1) \geq 2$.

(d) 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.

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.

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.

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.

6 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 3 and Figure 4. We depict the states as seen from the top with columns collapsing to single cells. The difference in a column is indicated by a number: a difference (a_0, a_1, a_2), with indices denoting the plane, is encoded as $a_0 + 2a_1 + 4a_2$. Additionally we gave them colors to make the figures easier to read.

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 3 and Figure 4 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 3 because it is easiest to understand thanks to its geometry. However, the upstairs trail core in Figure 4 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 ρ_{west}. ${\rho_{\text{east}}}^{-1}$ 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 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.

The upstairs trail core in Figure 4 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

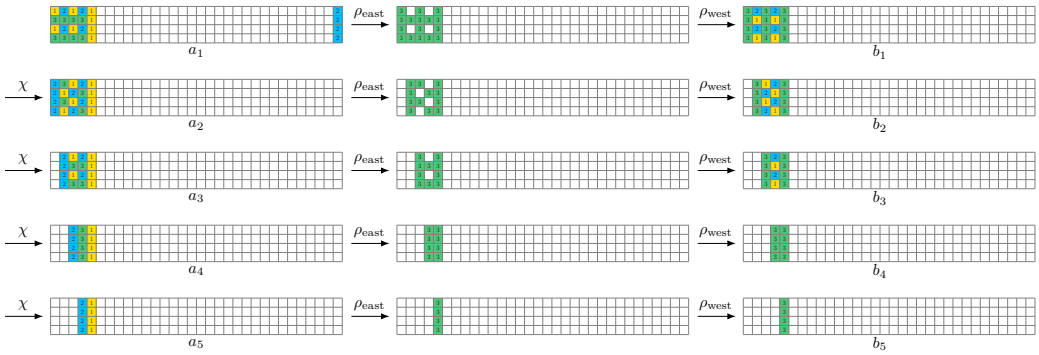

Figure 3: Differential staircase trail that goes downstairs

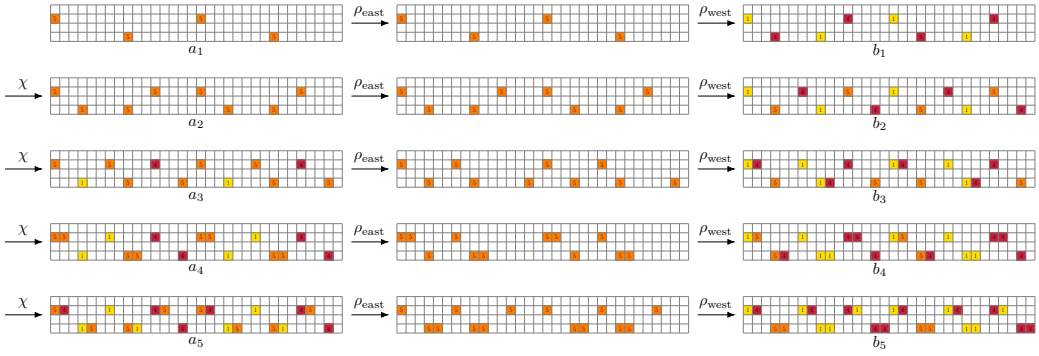

Figure 4: Differential staircase trail that goes upstairs

regularly expanding pattern over the rounds.

7 Conclusions

In this work, we presented some 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 could 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 the Cryptography Research Center of the Technology Innovation Institute (TII), Abu Dhabi (UAE), under the TII-Radboud project with title *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.

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

[DHP+19] 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 2), March 2019. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Xoodyak-spec-round2.pdf.

[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.

[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. Xoodoo cookbook. *IACR Cryptology ePrint Archive*, 2018:767, 2018.

[DHVV18c] J. Daemen, S. Hoffert, G. Van Assche, and R. Van Keer. XooTools software. https://github.com/XoodooTeam/Xoodoo, 2018.

[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.

[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] A. Mehrdad, S. Mella, L. Grassi, and J. 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.