Tighter trail bounds for Xoodoo

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

[1]Radboud University [2]STMicroelectronics

Radboud University

life.augmented

Overview

- XOODOO is the cryptographic permutation used in XOODYAK and XOOFFF

Overview

- XOODOO is the cryptographic permutation used in XOODYAK and XOOFFF
- Lower bounds for the weight of trails in XOODOO were previously proven
- Using XooTools: a C++ tool based on the **Trail core tree search** strategy, that
  - scans the space of all r-round **trail cores** with weight below a given target
  - using a **tree**-based approach

Overview

- XOODOO is the cryptographic permutation used in XOODYAK and XOOFFF
- Lower bounds for the weight of trails in XOODOO were previously proven
- Using XooTools: a C++ tool based on the **Trail core tree search** strategy, that
  - scans the space of all r-round **trail cores** with weight below a given target
  - using a **tree**-based approach
- In this work, we
  - improve XooTools to get tighter lower bounds, and
  - present upper bounds for more than 3 rounds

Overview

- Xoodoo is the cryptographic permutation used in Xoodyak and Xoofff
- Lower bounds for the weight of trails in Xoodoo were previously proven
- Using XooTools: a C++ tool based on the **Trail core tree search** strategy, that
  - scans the space of all r-round **trail cores** with weight below a given target
  - using a **tree**-based approach
- In this work, we
  - improve XooTools to get tighter lower bounds, and
  - present upper bounds for more than 3 rounds
- In this presentation, we talk about differential trails

Outline

Xoodoo

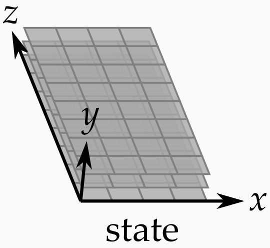
Trail cores and extension

Optimizing trail core extension

New bounds

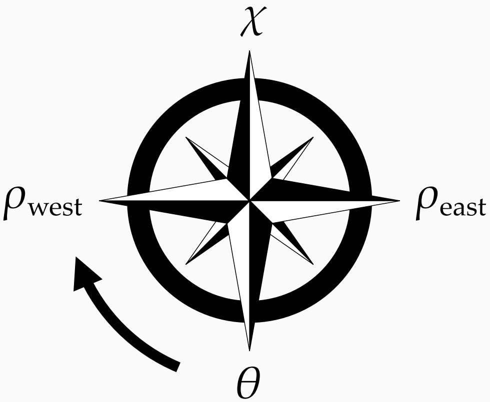
Outline

Xoodoo state



- State: 3 horizontal planes each consisting of 4 lanes

Xoodoo permutation



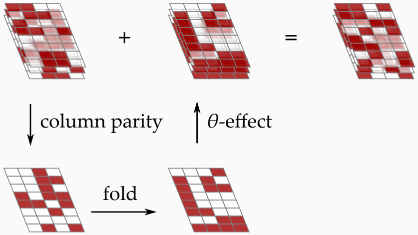
- Iterated: n_{r} rounds that differ only by round constant

Xoodoo round function

Xoodoo round function

θ :

$$P \leftarrow A_0 + A_1 + A_2$$
$$E \leftarrow P \lll (1,5) + P \lll (1,14)$$
$$A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\}$$



- Column parity mixer, good average diffusion

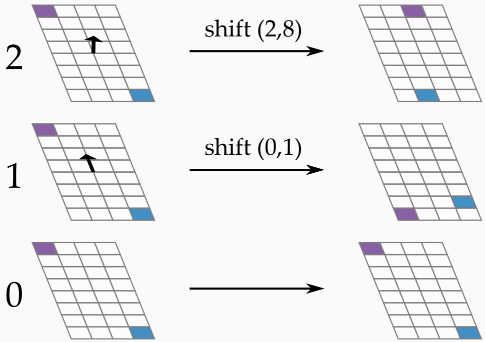
Xoodoo round function

θ :

$$P \leftarrow A_0 + A_1 + A_2$$
$$E \leftarrow P \lll (1,5) + P \lll (1,14)$$
$$A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\}$$

ρ_{west} :

$$A_1 \leftarrow A_1 \lll (1,0)$$
$$A_2 \leftarrow A_2 \lll (0,11)$$



- Plane shift

Xoodoo round function

θ :

$$P \leftarrow A_0 + A_1 + A_2$$
$$E \leftarrow P \lll (1,5) + P \lll (1,14)$$
$$A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\}$$

ρ_{west} :

$$A_1 \leftarrow A_1 \lll (1,0)$$
$$A_2 \leftarrow A_2 \lll (0,11)$$

ι :

$$A_{0,0} \leftarrow A_{0,0} + C_i$$

round i	c_i in hex
-11	0x00000058
-10	0x00000038
-9	0x000003C0
-8	0x000000D0
-7	0x00000120
-6	0x00000014
-5	0x00000060
-4	0x0000002C
-3	0x00000380
-2	0x000000F0
-1	0x000001A0
0	0x00000012

- Round constant addition

Xoodoo round function

θ :

$$P \leftarrow A_0 + A_1 + A_2$$
$$E \leftarrow P \lll (1,5) + P \lll (1,14)$$
$$A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\}$$

ρ_{west} :

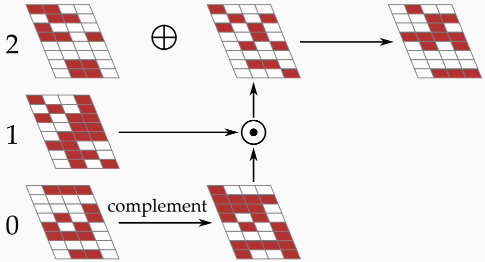
$$A_1 \leftarrow A_1 \lll (1,0)$$
$$A_2 \leftarrow A_2 \lll (0,11)$$

ι :

$$A_{0,0} \leftarrow A_{0,0} + C_i$$

χ :

$$B_0 \leftarrow \overline{A_1} \cdot A_2$$
$$B_1 \leftarrow \overline{A_2} \cdot A_0$$
$$B_2 \leftarrow \overline{A_0} \cdot A_1$$
$$A_y \leftarrow A_y + B_y \text{ for } y \in \{0,1,2\}$$



- χ as in KECCAK-p, operating on 3-bit columns
- Involution and same propagation differentially and linearly

Xoodoo round function

θ :

$$P \leftarrow A_0 + A_1 + A_2$$
$$E \leftarrow P \lll (1,5) + P \lll (1,14)$$
$$A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\}$$

ρ_{west} :

$$A_1 \leftarrow A_1 \lll (1,0)$$
$$A_2 \leftarrow A_2 \lll (0,11)$$

ι :

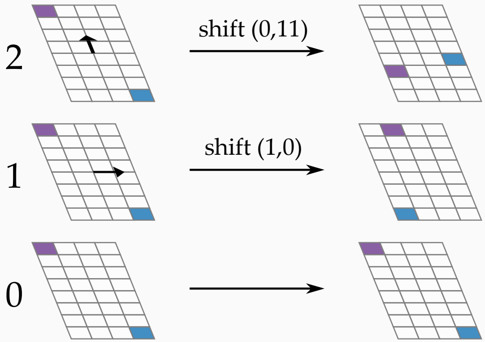
$$A_{0,0} \leftarrow A_{0,0} + C_i$$

χ :

$$B_0 \leftarrow \overline{A_1} \cdot A_2$$
$$B_1 \leftarrow \overline{A_2} \cdot A_0$$
$$B_2 \leftarrow \overline{A_0} \cdot A_1$$
$$A_y \leftarrow A_y + B_y \text{ for } y \in \{0,1,2\}$$

ρ_{east} :

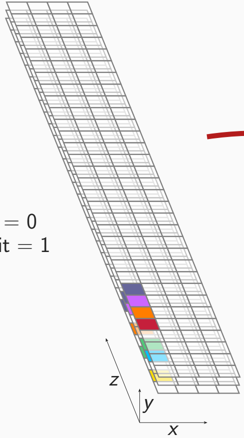
$$A_1 \leftarrow A_1 \lll (0,1)$$
$$A_2 \leftarrow A_2 \lll (2,8)$$

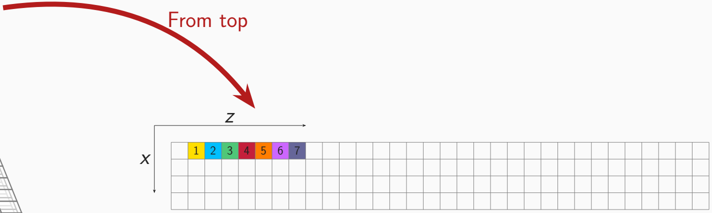


- Plane shift

Xoodoo state representation in this work

White cell: bit $= 0$
Colored cell: bit $= 1$





White cell: column $= 0$ (passive)
Colored cell: column $\neq 0$ (active)

Outline

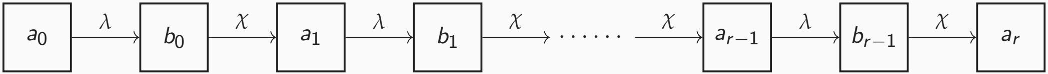
Xoodoo

Trail cores and extension

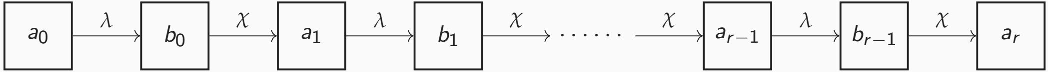
Optimizing trail core extension

New bounds

Differential trails in Xoodoo



- $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$



- $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$
- $\mathrm{w}_\chi(b_{i-1}, a_i) = -\log \mathrm{DP}_\chi(b_{i-1}, a_i)$

Differential trails in Xoodoo

$$a_0 \xrightarrow{\lambda} b_0 \xrightarrow{\chi} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} \cdots\cdots \xrightarrow{\chi} a_{r-1} \xrightarrow{\lambda} b_{r-1} \xrightarrow{\chi} a_r$$

- $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$
- $\mathrm{w}_\chi(b_{i-1}, a_i) = -\log \mathrm{DP}_\chi(b_{i-1}, a_i)$
- Weight of Q

$$\mathrm{w}(Q) = \mathrm{w}_\chi(b_0, a_1) + \mathrm{w}_\chi(b_1, a_2) + \cdots + \mathrm{w}_\chi(b_{r-1}, a_r)$$

Differential trails in Xoodoo

$$a_0 \xrightarrow{\lambda} b_0 \xrightarrow{\chi} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} \cdots\cdots \xrightarrow{\chi} a_{r-1} \xrightarrow{\lambda} b_{r-1} \xrightarrow{\chi} a_r$$

- $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$
- $\mathrm{w}_\chi(b_{i-1}, a_i) = -\log \mathrm{DP}_\chi(b_{i-1}, a_i)$
- Weight of Q

$$\mathrm{w}(Q) = \mathrm{w}_\chi(b_0, a_1) + \mathrm{w}_\chi(b_1, a_2) + \cdots + \mathrm{w}_\chi(b_{r-1}, a_r)$$

- For all valid differentials over χ_3: $\mathrm{DP}_{\chi_3} = \frac{1}{4}$ and $\mathrm{w}_{\chi_3} = 2$

$$\implies \mathrm{w}(Q) = 2 \cdot \#\text{ active S-boxes }(Q) = 2 \cdot (n_c(b_0) + n_c(b_1) + \cdots + n_c(b_{r-1}))$$

Differential trails in Xoodoo

$$a_0 \xrightarrow{\lambda} b_0 \xrightarrow{\chi} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} \cdots\cdots \xrightarrow{\chi} a_{r-1} \xrightarrow{\lambda} b_{r-1} \xrightarrow{\chi} a_r$$

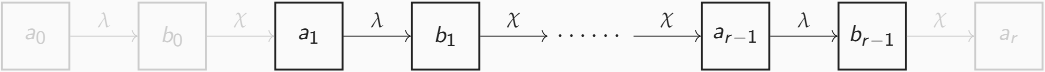
- $\lambda = \rho_{\text{west}} \circ \theta \circ \rho_{\text{east}}$
- $\mathrm{w}_\chi(b_{i-1}, a_i) = -\log \mathrm{DP}_\chi(b_{i-1}, a_i)$
- Weight of Q

$$\mathrm{w}(Q) = \mathrm{w}_\chi(b_0, a_1) + \mathrm{w}_\chi(b_1, a_2) + \cdots + \mathrm{w}_\chi(b_{r-1}, a_r)$$

- For all valid differentials over χ_3: $\mathrm{DP}_{\chi_3} = \frac{1}{4}$ and $\mathrm{w}_{\chi_3} = 2$

$$\begin{aligned} \Longrightarrow \mathrm{w}(Q) = 2 \cdot \#\text{ active S-boxes }(Q) &= 2 \cdot (n_c(b_0) + n_c(b_1) + \cdots + n_c(b_{r-1})) \\ &= 2 \cdot (n_c(a_1) + n_c(b_1) + \cdots + n_c(b_{r-1})) \end{aligned}$$

Trail cores



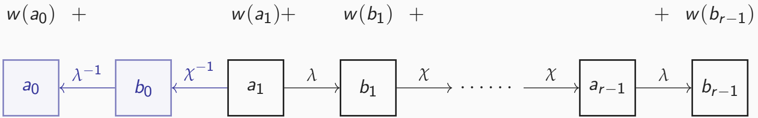
- Trail core: equivalence class of trails with $(a_1, b_1, \cdots, b_{r-1})$ in common and same weight

$$2 \cdot (n_c(a_1) + n_c(b_1) + \cdots + n_c(b_{r-1}))$$

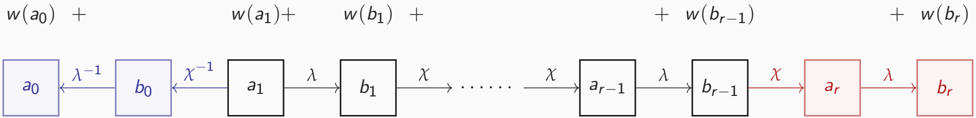
- We can restrict the search to trail cores $\implies$ avoid two non-linear layers
- Start from 2-round trail cores and **extend**

$w(a_1) + \quad w(b_1) \quad + \qquad\qquad + \quad w(b_{r-1})$

Extension over χ

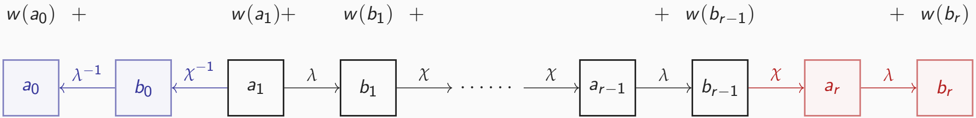


- Trail cores can be extended in the backward



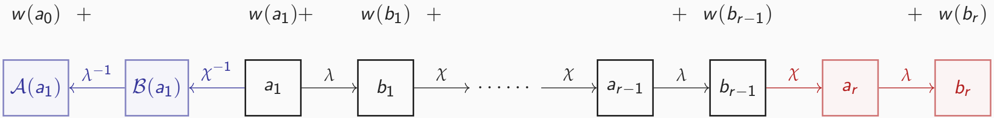
- Trail cores can be extended in the backward and forward direction

Extension over χ



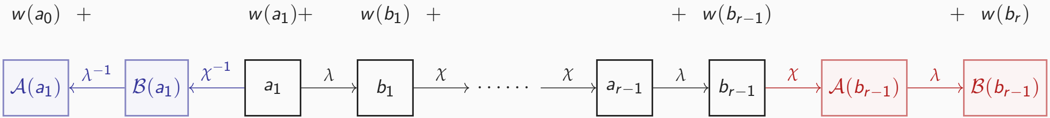
- Trail cores can be extended in the backward and forward direction
- χ has degree 2

Extension over χ



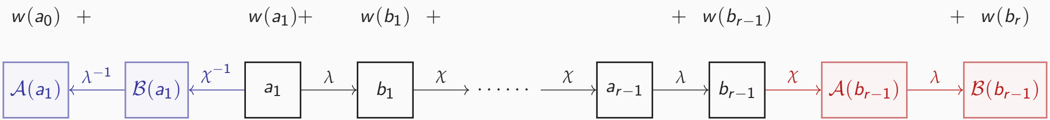
- Trail cores can be extended in the backward and forward direction
- χ has degree 2
  - Valid b_0's form an affine space $\mathcal{B}(a_1)$ of dim $= 2 \cdot n_c(a_1)$

Extension over χ



- Trail cores can be extended in the backward and forward direction
- χ has degree 2
  - Valid b_0's form an affine space $\mathcal{B}(a_1)$ of dim $= 2 \cdot n_c(a_1)$
  - Valid a_r's form an affine space $\mathcal{A}(b_{r-1})$ of dim $= 2 \cdot n_c(b_{r-1})$

Extension over χ



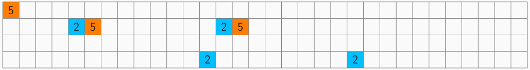
- Trail cores can be extended in the backward and forward direction
- χ has degree 2
  - Valid b_0's form an affine space $\mathcal{B}(a_1)$ of dim $= 2 \cdot n_c(a_1)$
  - Valid a_r's form an affine space $\mathcal{A}(b_{r-1})$ of dim $= 2 \cdot n_c(b_{r-1})$

Δ	o	v_1	v_2
100	100	001	010
010	010	100	001
110	010	110	001
001	001	010	100
101	100	101	010
011	001	011	100
111	001	011	101

over χ_3

Example

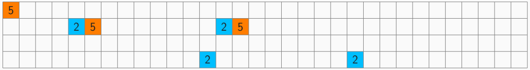
- Difference a_1 that we want to extend in the backward direction.



a_1

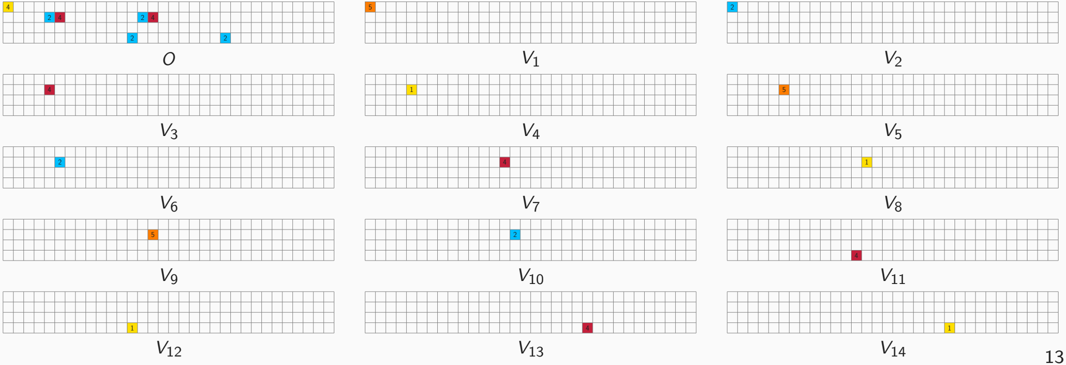
Example

- Difference a_1 that we want to extend in the backward direction.

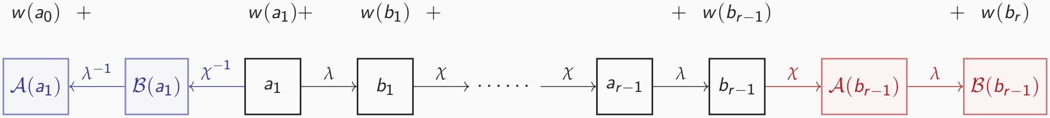


a_1

- $\mathcal{B}(a_1) = O + \langle V_1, V_2, \ldots, V_{14} \rangle$:



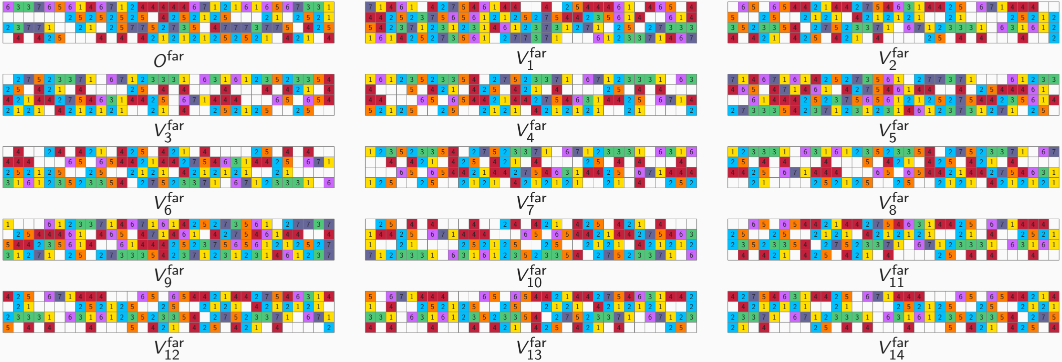
Extension using the far view



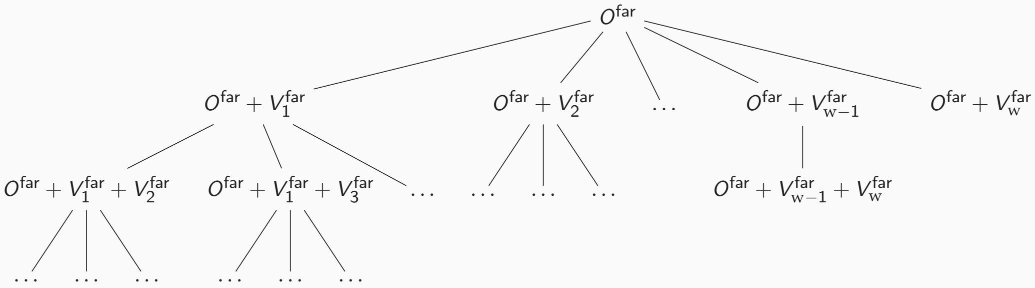
- Since λ is linear, we can apply it to the offset and basis vectors
- Example with backward extension:

$$\begin{aligned}\mathcal{A}(a_1) &= O^{\text{far}} + \left\langle V_1^{\text{far}}, V_2^{\text{far}}, \ldots, V_{\mathrm{w}}^{\text{far}} \right\rangle \\ &= \lambda^{-1}(O) + \left\langle \lambda^{-1}(V_1), \lambda^{-1}(V_2), \ldots, \lambda^{-1}(V_{\mathrm{w}}) \right\rangle\end{aligned}$$

The resulting representation of $\mathcal{A}(a_1)$ is:



Extension as a tree-based search



- $\mathcal{A}(a_1) = O^{\text{far}} + \left\langle V_1^{\text{far}}, \dots, V_{\text{w}}^{\text{far}} \right\rangle$
- The root of the tree is the offset O^{far}
- To avoid duplicates, order relation among basis vectors: $V_i^{\text{far}} \prec V_j^{\text{far}}$ if and only if $i < j$

Lower bounding the weight of nodes

- Addition of V_i^{far} can turn active columns to passive columns
- $\implies$ a node can have weight smaller than the weight of its parent

Lower bounding the weight of nodes

- Addition of V_i^{far} can turn active columns to passive columns
- $\implies$ a node can have weight smaller than the weight of its parent
- **Stable bits**: bits of a node that have same value in all node's descendants
- **Stability mask**: a state value S_i where a bit has a value 1 if it is stable and 0 otherwise

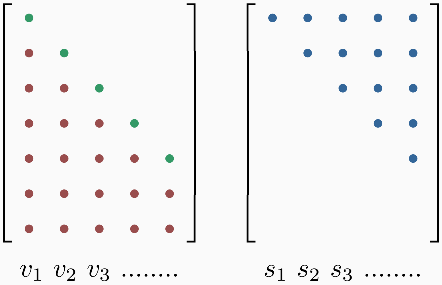
Lower bounding the weight of nodes

- Addition of V_i^{far} can turn active columns to passive columns
- $\implies$ a node can have weight smaller than the weight of its parent
- **Stable bits**: bits of a node that have same value in all node's descendants
- **Stability mask**: a state value S_i where a bit has a value 1 if it is stable and 0 otherwise
- $\mathrm{w}(N \wedge S_i)$ lower bounds the weight of all descendants of N
- When $\mathrm{w}(N \wedge S_i) > T$ we can safely prune

Lower bounding the weight of nodes

- Addition of V_i^{far} can turn active columns to passive columns
- $\implies$ a node can have weight smaller than the weight of its parent
- **Stable bits**: bits of a node that have same value in all node's descendants
- **Stability mask**: a state value S_i where a bit has a value 1 if it is stable and 0 otherwise
- $\mathrm{w}(N \wedge S_i)$ lower bounds the weight of all descendants of N
- When $\mathrm{w}(N \wedge S_i) > T$ we can safely prune
- We would like that the number of stable bits in S_i grows quickly with i
- **How can we define good stability masks?**

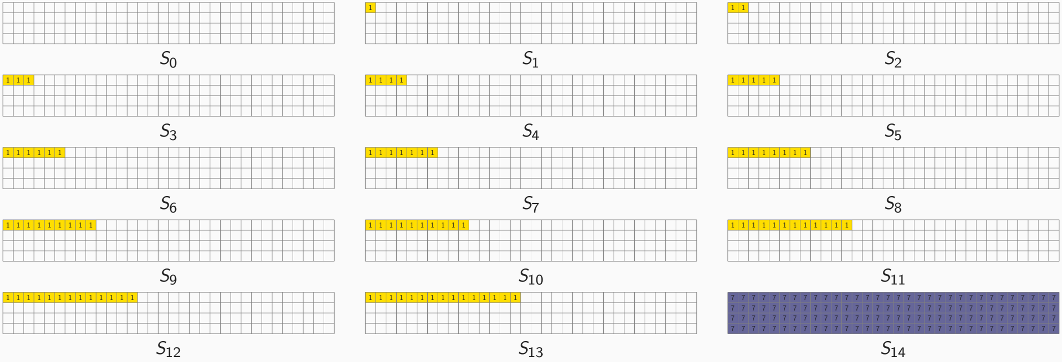
Original definition of stability masks [DHVV18a]



- Stability masks $S_0, S_1, \ldots, S_\mathrm{w}$ depend on the basis $\left\{V_1^{\text{far}}, V_2^{\text{far}}, \ldots, V_\mathrm{w}^{\text{far}}\right\}$
- *Triangularize* the basis $\left\{V_1^{\text{far}}, V_2^{\text{far}}, \ldots, V_\mathrm{w}^{\text{far}}\right\}$
- Using lexicographic order relation of the bit positions $p = (x, y, z)$
- **Pivot bit** p_i: the smallest active bit in V_i^{far} (it is passive in all V_j^{far} with $j > i$)
- **Stability mask** S_i: all bits in positions $\leq p_i$

Example continued

These lead to the following stability masks:

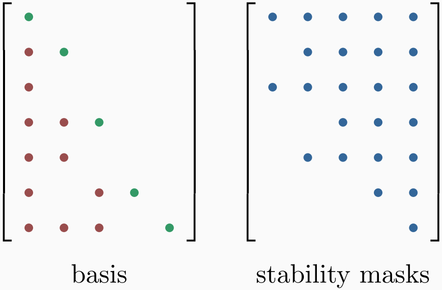


The number of stable bits in each mask S_i:

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

Outline

Redefining stability masks

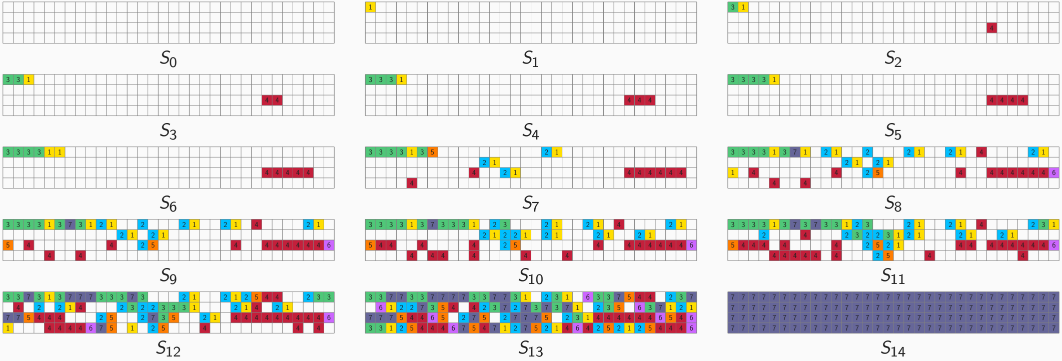


- For a node $N = O + \ldots + V_i^{\text{far}}$
- A bit that is 0 in all V_{i+1}^{far} to $V_{\mathrm{w}}^{\text{far}}$ is stable
- We redefine stability masks as:

$$S_i = \bigwedge_{j>i} \overline{V_j^{\text{far}}} \, . \tag{1}$$

Example continued

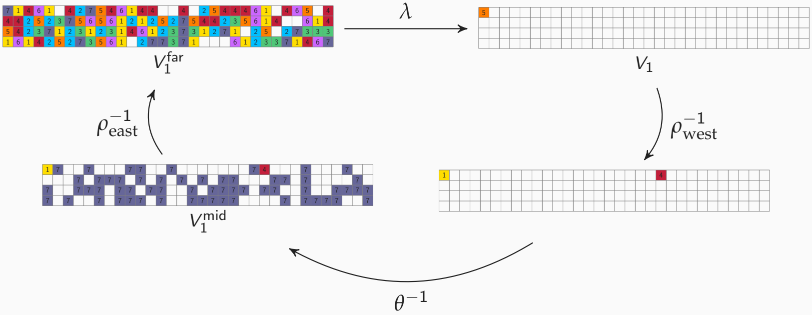
By applying (1) we obtain the following stability masks:



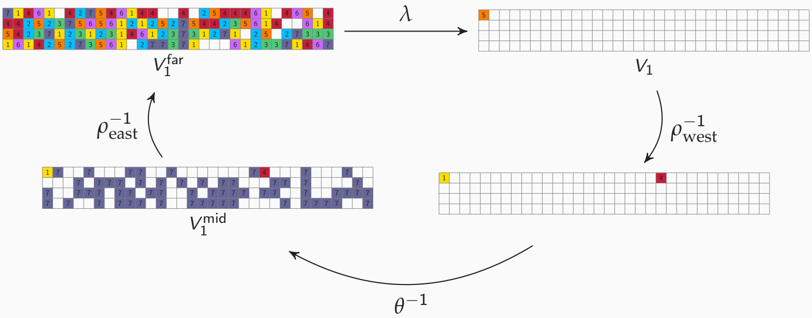
The number of stable bits grows more quickly with i:

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

Triangularization at the mid view

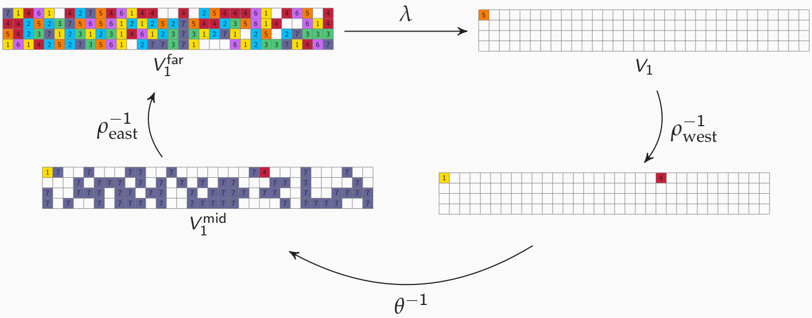


Triangularization at the mid view



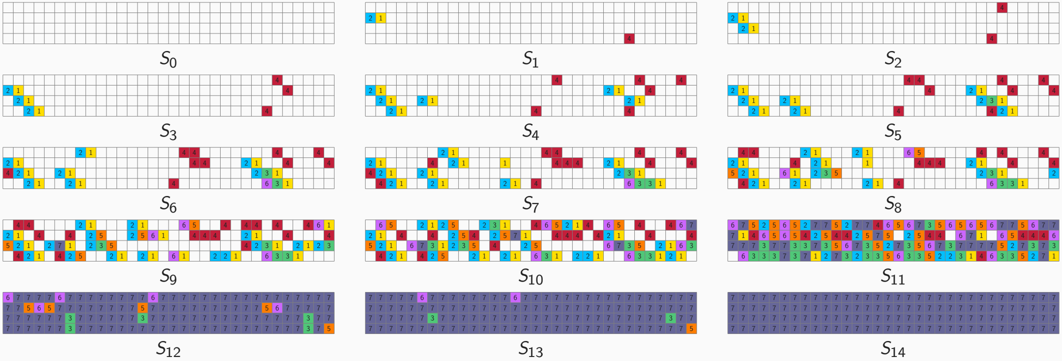
- A whole active column in the mid view as a pivot to stabilize three bits in three different columns in the far view

Triangularization at the mid view



- A whole active column in the mid view as a pivot to stabilize three bits in three different columns in the far view
- Further improvements
  - Prioritize *go-columns*
  - Following a diagonal order

Combining all optimizations, we obtain the following stability masks:



The number of stable bits increases by at least 3 with each i:

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

Outline

Xoodoo

Trail cores and extension

Optimizing trail core extension

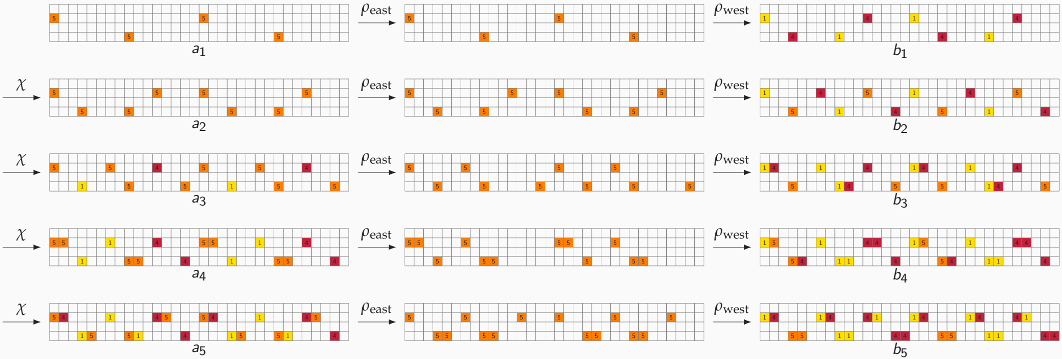
New bounds

Bounds for differential and linear trails

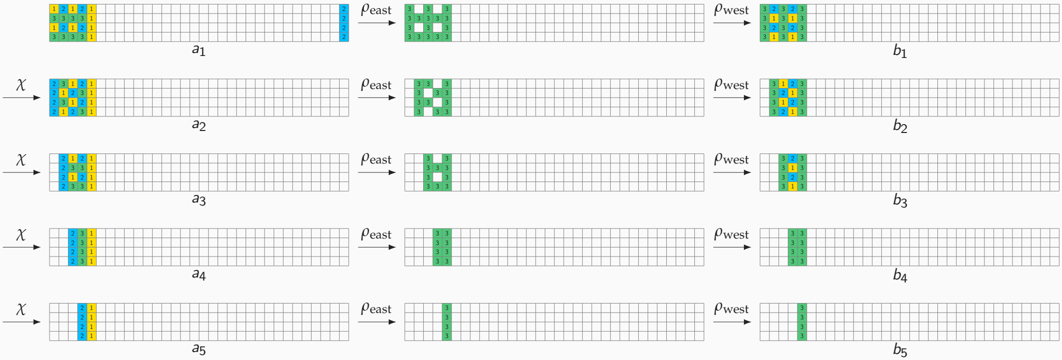
# rounds	Previous works				This work	
	lower bound		best known		lower bound	best known
1	2	[DHVV18a]	2	[DHVV18a]	-	-
2	8	[DHVV18a]	8	[DHVV18a]	-	-
3	36	[DHVV18a]	36	[DHVV18a]	-	-
4	74	[DHP+20]	-		**80**	**80**
5	94	[DHP+20]	-		98	120
6	108	[The21]	-		**132**	160
8	148	[DHP+20]	-		176	264
10	188	[DHP+20]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

Upstairs trail cores

Trails with weight $8 + 16 + 24 + 32 + 40 + 48 \ldots$



Trails with weight $\ldots + 48 + 40 + 32 + 24 + 16 + 8$



Conclusions

- We introduced optimizations to improve trail core tree search in Xoodoo
- We proved tighter lower bounds for the weight of differential and linear trails
  - tight bound for 4 rounds
  - beyond 128 for 6 rounds and 256 for 12 rounds
- We proved upper bounds using staircase trail cores

Thank you for your attention!

Backup: weight profiles for best trails

1:	2																								
2:	4	+	4			=	8																		
3:	12	+	12	+	12	=	36																		
4:															32	+	24	+	16	+	8			=	80
5:															32	+	24	+	16	+	8	+	40	=	120
6:													40	+	32	+	24	+	16	+	8	+	40	=	160
7:											48	+	40	+	32	+	24	+	16	+	8	+	40	=	208
8:									56	+	48	+	40	+	32	+	24	+	16	+	8	+	40	=	264
9:							64	+	56	+	48	+	40	+	32	+	24	+	16	+	8	+	40	=	328
10:					72	+	64	+	56	+	48	+	40	+	32	+	24	+	16	+	8	+	40	=	400
11:			80	+	72	+	64	+	56	+	48	+	40	+	32	+	24	+	16	+	8	+	40	=	480
12:	88	+	80	+	72	+	64	+	56	+	48	+	40	+	32	+	24	+	16	+	8	+	40	=	564