Differential analysis of the ternary hash function Troika

Christina Boura, Margot Funk, Yann Rotella

Differential analysis of the ternary hash function Troika

Christina Boura, Margot Funk, and Yann Rotella

Université Paris-Saclay, UVSQ, CNRS, Laboratoire de mathématiques de Versailles, 78000,Versailles, France
christina.boura@uvsq.fr, margot.funk@uvsq.fr, yann.rotella@uvsq.fr

Abstract. Troika is a sponge-based hash function designed by Kölbl, Tischhauser, Bogdanov and Derbez in 2019. Its specificity is that it is defined over $\mathbb{F}_3$ in order to be used inside IOTA's distributed ledger but could also serve in all settings requiring the generation of ternary randomness. To be used in practice, Troika needs to be proven secure against state-of-the-art cryptanalysis. However, there are today almost no analysis tools for ternary designs. In this article we take a step in this direction by analyzing the propagation of differential trails of Troika and by providing bounds on the weight of its trails. For this, we adapt a well-known framework for trail search designed for Keccak and provide new advanced techniques to handle the search on $\mathbb{F}_3$. Our work demonstrates that providing analysis tools for non-binary designs is a highly non-trivial research direction that needs to be enhanced in order to better understand the real security offered by such non-conventional primitives.

Keywords: differential cryptanalysis, Troika, ternary design

1 Introduction

Almost all symmetric-key cryptographic schemes known today are defined over the binary field $\mathbb{F}_2$. However, recent advances in cryptology imply that ternary symmetric constructions could fit better in some particular contexts. A first notable example is a cryptocurrency and distributed ledger technology called IOTA. This platform is based on a ternary architecture and its security relied, among others, on the security of a ternary hash function. For this purpose, a hash function called Curl-P was designed for IOTA, but was soon after found to have devastating security issues [7]. Curl-P was then replaced by Kerl, another ternary hash function that could be seen as an adaptation of SHA-3 to $\mathbb{F}_3$. This adaptation was however not natural and led to a particularly inefficient design. For this reason, Kölbl, Tischhauser, Bogdanov and Derbez proposed in 2019 Troika [8], a ternary hash function claimed to be secure and efficient at the same time.

Even if the advantage of a ternary construction is not completely clear in the IOTA paradigm, ternary symmetric designs could be very meaningful in other

settings. For example, in a completely different context, Debris-Alazard, Sendrier and Tillich designed in 2019 Wave, a ternary code-based signature scheme [6]. While this scheme is asymmetric, it needs the generation of good quality ternary randomness. Troika would then be a natural candidate for this purpose, as hash functions are regularly used as random number generators.

In the last 30 years, cryptanalysis of traditional binary schemes has been extremely developed. On the other hand, almost nothing is known about the security of ternary symmetric schemes, starting from their resistance against very basic attacks as the differential one. In this article we propose to take a step into this direction by providing an enhanced analysis of the hash function Troika against differential cryptanalysis. The goal of our work is two-fold. First, we aim to provide the first third-party differential analysis against this function by giving concrete upper bounds on the probability of its differential trails. Second, we develop non-trivial techniques for searching differential bounds for ternary designs. We believe that our work can serve as a starting point for the analysis against differential cryptanalysis of ternary designs to come.

Our contributions

Troika is a sponge-based construction whose round function has a design very similar to Keccak. Searching for good differential trails for Keccak is a very complex task due mainly to its large state and its weakly aligned inner components. Dedicated algorithms for generating all trails for Keccak below a given weight, leading to upper bounds on the probability of differential trails for a certain number of rounds were developed in [4,9]. These articles were the starting point of our work. We adapted and extended them for the ternary case and applied them to Troika. This adaptation was not straightforward and we needed to develop advanced techniques to be able to compute on trits (i.e. elements of $\mathbb{F}_3$) in a reasonable time. Most of our improvements concern the tree traversal used to generate out-kernel 2-round trail cores. This applies to the computation of the so-called *runs*, a way to organize the differential patterns of a state. More precisely, we defined an equivalence class notion on the runs that permitted us to treat less trail cores and we improved a specific order relation such that fewer collisions between runs occur, leading to more efficient tree pruning. This permitted us to scan all trail cores up to weight 41 and to prove that there do not exist 6-round trails with weight lower than 82. In comparison, the designers of Troika were only able to provide results on smaller versions of Troika by using a MILP-based approach. Their approach could not apply to the original version of the function because of its large state.

Eventually, we show that adapting the existing tool from Keccak to Troika is a highly nontrivial problem, even if the design strategy of both primitives is similar. This is mainly due to the fact that, contrary to the binary case, active trits can take two values (1 or 2), leading to a huge number of trails to treat. Thus, providing interesting bounds for Troika is not possible without introducing new ideas to prune the search tree efficiently. More generally, our work highlights

that adapting existing tools and methods from $\mathbb{F}_2$ to $\mathbb{F}_3$ is a challenging task and indicates that more research efforts are needed towards this direction.

The rest of the article is organized as follows. In Section 2 we recall classical notions regarding differential trail search on symmetric primitives. Section 3 presents the specifications of Troika while Section 4 describes the general strategy used to scan all trail cores of a certain weight. Our algorithms for generating 2-round trail cores for Troika are described in Section 5. These 2-round trail cores can then be extended to 6-round trail cores with the techniques described in Section 6.2. Section 6.1 is dedicated to the direct search of 3-round trail cores of a particular profile, called *in-kernel* that can be generated relatively easier without the techniques of Section 5. Finally, Section 7 presents some statistics on the trails we managed to scan.

2 Differential cryptanalysis

Differential cryptanalysis is a powerful attack against symmetric primitives that exploits input differences that propagate through the primitive to some output difference with high probability. We recall in this section basic definitions regarding differential properties.

Notation Let $\mathbb{K}$ be a finite field. We want to study a substitution-permutation primitive f from $\mathbb{K}^n$ to $\mathbb{K}^n$. Typically for Troika, $\mathbb{K}$ is the field $\mathbb{F}_3$. We write $+$ for the usual coordinate-wise addition in $\mathbb{K}^n$ and use the operator $-$ in the usual way. We call *state* the value of the variable of $\mathbb{K}^n$ which is progressively updated through the substitution permutation network. We denote by $\mathrm{R}_0, \ldots, \mathrm{R}_{r-1}$ the r round functions that are iteratively applied to the state i.e. $f = \mathrm{R}_{r-1} \circ \cdots \circ \mathrm{R}_1 \circ \mathrm{R}_0$. For $k \in [0, r-1]$, the round function R_k is the composition of a nonlinear map χ, a linear map λ and a round constant addition ι_k, that is to say $\mathrm{R}_k = \iota_k \circ \lambda \circ \chi$. The state is divided for the nonlinear layer into disjoint parts of equal size, called *boxes*. A nonlinear map, called *S-box* and supposed here to be bijective, is applied to each box of the state.

Differentials Let $f\colon \mathbb{K}^n \to \mathbb{K}^n$ be a permutation. A *differential* over f is a couple $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) \in (\mathbb{K}^n)^2$. The difference Δ_{in} is said to be an *input difference* while the difference Δ_{out} is called an *output difference*. The *differential probability* of a differential $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ over f is defined as

$$\mathrm{DP}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) \coloneqq \frac{\#\{x \in \mathbb{K}^n : f(x + \Delta_{\mathrm{in}}) - f(x) = \Delta_{\mathrm{out}}\}}{\#\mathbb{K}^n}.$$

If $\mathrm{DP}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) > 0$, we say that the input difference Δ_{in} is *compatible* with the output difference Δ_{out} through f and call $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ a *valid* differential over f. The *weight of a valid differential* $(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ over f is

$$\mathrm{w}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) \coloneqq -\log_{\#\mathbb{K}}\left(\mathrm{DP}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})\right).$$

Differential trails When the dimension n of the domain of the iterated function f is large, it is in general not possible to compute the exact differential probability of a differential over f. Thus, we try to approximate this probability by studying sequences of differences, called differential trails. In the following, we denote by $f = \mathrm{R}_{r-1} \circ \cdots \circ \mathrm{R}_0$ an iterated function from $\mathbb{K}^n$ to $\mathbb{K}^n$. For $k \in [1, r]$, a *k-round differential trail* over f is a sequence $Q = (q^{(0)}, q^{(1)}, \ldots, q^{(k)}) \in (\mathbb{K}^n)^{k+1}$ where for $0 \leq i \leq k-1$, $(q^{(i)}, q^{(i+1)})$ is a valid differential over R_i. The *differential probability* of Q is defined as

$$\mathrm{DP}_f(Q) \coloneqq \frac{\#\{(x, x') \in (\mathbb{K}^n)^2 : \forall i \in [0, k],\ f[i](x) - f[i](x') = q^{(i)}\}}{\#\mathbb{K}^n},$$

where $f[0]$ is the identity function and for $i \in [1, k]$, $f[i] = \mathrm{R}_{i-1} \circ \ldots \circ \mathrm{R}_0$. If we write $\mathrm{DT}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})$ for the set composed of all trails over f of the form $(\Delta_{\mathrm{in}}, q^{(1)}, \ldots, q^{(k-1)}, \Delta_{\mathrm{out}})$, then $\mathrm{DP}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}}) = \sum_{Q \in \mathrm{DT}_f(\Delta_{\mathrm{in}}, \Delta_{\mathrm{out}})} \mathrm{DP}_f(Q)$.

Computing the exact probability of a differential trail is in general out of reach. Instead, we try to approximate it, looking locally at the differentials that form the trail. The *weight of a differential trail* $Q = (q^{(0)}, q^{(1)}, \ldots, q^{(k)})$ over f, defined as

$$\mathrm{w}_f(Q) \coloneqq \sum_{i=0}^{k-1} \mathrm{w}_{\mathrm{R}_i}(q^{(i)}, q^{(i+1)}),$$

is a value used to approximate the differential probability of the trail. We expect to have $\mathrm{DP}_f(Q) \simeq (\#\mathbb{K})^{-\mathrm{w}_f(Q)}$. The weight of the trail Q is easy to calculate as for all $i \in [0, k-1]$, $\mathrm{w}_{R_i}\left(q^{(i)}, q^{(i+1)}\right) = \mathrm{w}_\chi\left(q^{(i)}, \lambda^{-1}\left(q^{(i+1)}\right)\right)$. This last quantity can be computed as the sum of the weights of the differentials given by the restriction of $\left(q^{(i)}, \lambda^{-1}\left(q^{(i+1)}\right)\right)$ to the domain and co-domain of the S-boxes. To estimate the security of a primitive against differential attacks, it is essential to ensure that there do not exist differential trails with low weight.

Trail cores [4] A k-round differential trail over f will be represented as follows:

$$b_0 \xrightarrow{\chi} a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda} \cdots \xrightarrow{\lambda} b_{k-1} \xrightarrow{\chi} a_k \xrightarrow{\lambda} b_k,$$

where for $i \in [1, k]$, $b_i = \lambda(a_i)$ and $\mathrm{DP}_\chi(b_{i-1}, a_i) > 0$. Since the S-boxes are bijective, if a differential (b, a) is valid over χ, then the nonzero boxes of the difference a are located at the same positions as those of b. These boxes and their corresponding S-boxes are said to be *active*.

The choice of the difference b_0 and the choice of the difference a_k of the k-round trail do not affect the number of active S-boxes. For this reason it is convenient to specify only the central part of the trail, namely $(a_1, \ldots, b_{k-1})$. This defines a set of differential trails

$$\langle a_1, \ldots, b_{k-1} \rangle \coloneqq \{Q = (b'_0, a_1, \ldots, b_{k-1}, a'_k, b'_k) : Q \text{ is a } k\text{-round trail over } f\}$$

called a *k-round trail core* of f. The *weight of a k-round trail core* $\langle a_1, \ldots, b_{k-1} \rangle$, written $\mathrm{w}\langle a_1, \ldots, b_{k-1} \rangle$, is the minimum of the weights of the trails that belong

to the trail core. To make this explicit, the notions of minimum direct and reverse weight of a difference are introduced. The *minimum reverse weight* of a difference $\Delta \in \mathbb{K}^n$ is given by

$$\widetilde{\mathrm{w}}^{\mathrm{rev}}(\Delta) = \min_b \{\mathrm{w}_\chi(b, \Delta)\},$$

where the minimum is taken over the differences $b \in \mathbb{K}^n$ such that (b, Δ) is a valid differential over χ. The *minimum direct weight* of $\Delta \in \mathbb{K}^n$ is

$$\widetilde{\mathrm{w}}^{\mathrm{dir}}(\Delta) = \min_a \{\mathrm{w}_\chi(\Delta, a)\},$$

where the minimum is taken over the differences $a \in \mathbb{K}^n$ such that (Δ, a) is a valid differential over χ. The weight of a 2-round trail core $\langle a, b\rangle$ is then $\mathrm{w}\langle a, b\rangle = \widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b)$. If $k \geq 2$, the weight of a k-round trail core $\langle a_1, \ldots, b_{k-1}\rangle$ is

$$\widetilde{\mathrm{w}}^{\mathrm{rev}}(a_1) + \sum_{i=1}^{k-2} \mathrm{w}_\chi(b_i, a_{i+1}) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b_{k-1}).$$

Following the same approach as in [9], we gradually generate differential trail cores, starting from short trail cores and extending them in the forward and backward direction.

3 Troika description

We denote the finite field with 3 elements by $\mathbb{F}_3$. We call an element of $\mathbb{F}_3 = \{0, 1, 2\}$ a trit and an element of $\mathbb{F}_3^3$ a tryte.

The Troika round function Troika is a hash function from $\mathbb{F}_3^*$ to $\mathbb{F}_3^{243}$. It follows the sponge construction [1] with a rate of 243 trits and a capacity of 486 trits. Therefore, the state is a vector of 729 trits. The permutation f used inside the sponge construction is composed of 24 rounds. Each round follows the Keccak [2] philosophy and is composed of 5 step functions: a nonlinear layer SubTrytes, two shuffling layers ShiftRows and ShiftLanes, a parity-mixer AddColumnParity and a round constant addition $\mathsf{AddRoundConstant}_i$. Taking the same notations as in Keccak-f, we abbreviate them here respectively by χ, ρ_r, ρ_ℓ, θ and ι_i. We also denote by ρ the composition $\rho = \rho_\ell \circ \rho_r$ and λ the whole linear layer. The round function R_i is then given by $\mathrm{R}_i = \iota_i \circ \lambda \circ \chi$, where χ is the nonlinear layer and $\lambda = \theta \circ \rho$ is the linear layer.

Notations and definitions The round function operates on a state of 729 trits, organized as a three-dimensional array of size $9 \times 3 \times 27$ trits. In the following, x will always be an integer between 0 and 8, x_B an integer between 0 and 2, y an integer between 0 and 2 and z an integer between 0 and 26. Moreover, coordinates along the x, y and z axes will always be considered modulo 9, 3 and

27 respectively. We use the letter A to denote a state, that is either seen as an element of $\mathbb{F}_3^{729}$ or as a three-dimensional array. We write $A[x, y, z]$ for the trit of A of coordinates (x, y, z). The box of A of coordinates (x_B, y, z) is the tryte $(A[3x_B + i, y, z])_{i\in[0,\,2]}$. We call a box-column a triplet of boxes that have the same x_B and z coordinates. The different parts of the state of Troika can be visualised in Figure 9 of Appendix B. We denote by $e_{(x,y,z)}$ the canonical state whose components are all zero, except the component of coordinates (x, y, z) that equals 1.

Definition 1. (trit-activity pattern) *The* trit-activity pattern *of a state A is the vector $\overline{A}$ of $\{0, 1\}^{729}$ that indicates the nonzero trits of A, called active trits. It is defined as $\overline{A}[x, y, z] = 0$ if $A[x, y, z] = 0$ and $\overline{A}[x, y, z] = 1$ otherwise.*

Definition 2. (box-activity pattern) *The* box-activity pattern *of a state A is the vector of $\{0, 1\}^{729}$ denoted by $\mathsf{box}(A)$ and given by $\mathsf{box}(A)[x, y, z] = 1$ if the box of A containing the trit of coordinates (x, y, z) is active and $\mathsf{box}(A)[x, y, z] = 0$ otherwise.*

Definition 3. (box weight) *The* box weight [3] *of a state A is the number of boxes of A that are active, namely $\#\{(x_B, y, z) : (A[3x_B + i, y, z])_{i\in[0,\,2]} \neq (0, 0, 0)\}$. It is denoted by $\mathrm{w}_{\mathrm{box}}(A)$.*

The nonlinear layer The map $\mathsf{SubTrytes}$, here denoted by χ, is the parallel application of 9×27 S-boxes of size 3 trits. The minimum reverse and direct weights of a state are easy to compute. Indeed, by computing the differential distribution table of the S-box, one can verify the following properties.

Property 1. For all differences $A \in \mathbb{F}_3^{729}$, $\widetilde{\mathrm{w}}^{\mathrm{rev}}(A) = \widetilde{\mathrm{w}}^{\mathrm{dir}}(A) = 2\,\mathrm{w}_{\mathrm{box}}(A)$.

Property 2. It can be noticed that the following input differences Δ_{in} are compatible through the S-box map with output differences Δ_{out} of the form :

$\Delta_{\mathrm{in}} \in \mathbb{F}_3^3$	$\Delta_{\mathrm{out}} = (t_0, t_1, t_2) \in \mathbb{F}_3^3$	$\Delta_{\mathrm{in}} \in \mathbb{F}_3^3$	$\Delta_{\mathrm{out}} = (t_0, t_1, t_2) \in \mathbb{F}_3^3$
(0, 0, 1)	$t_2 = 1$	(1, 0, 0)	$t_1 \neq 0$
(0, 0, 2)	$t_2 = 2$	(2, 0, 0)	$t_1 \neq 0$

This property is used in Appendix E.3 to extend trail cores in the forward direction.

Remark 1. If a differential (Δ, Δ') is valid through the bijective nonlinear map χ, then the differences Δ and Δ' have the same box-activity pattern.

The shuffling layers The permutations $\mathsf{ShiftRows}$ and $\mathsf{ShiftLanes}$ move the trits of the state along the x and z axes, without modifying their value. We denote by $\tilde{\rho}$ the permutation of the set of coordinates that satisfies $\rho\left(e_{(x,y,z)}\right) = e_{\tilde{\rho}(x,y,z)}$ for all $(x, y, z) \in [0,\, 8] \times [0,\, 2] \times [0,\, 26]$. For a complete description of these layers we refer to [8].

The parity-mixer The map AddColumnParity, written here θ, is a column parity mixer [10]. The *parity plane* of a state A is the bi-dimensional array of size 9×27 that indicates the parity (0, 1 or 2) of each column of A. It is denoted by $P(A)$ and defined as

$$P(A)[x, z] \coloneqq \sum_{y=0}^{2} A[x, y, z] \bmod 3.$$

It is said that the column of A of coordinates (x, z) has parity $P(A)[x, z] \in \mathbb{F}_3$. The θ-*effect plane* of A is the bi-dimensional array of size 9×27, denoted by $E(A)$ and defined as

$$E(A)[x, z] \coloneqq P(A)[x-1, z] + P(A)[x+1, z+1].$$

If $E(A)[x, z]$ is zero, the column of A of coordinates (x, z) is said to be *non affected* by θ. Otherwise, the column is said to be *affected* by $E(A)[x, z] \in \mathbb{F}_3$. The map θ is defined as

$$\theta(A)[x, y, z] \coloneqq A[x, y, z] + E(A)[x, z].$$

The set $K \coloneqq \{A \in \mathbb{F}_3^{729} : \theta(A) = A\}$ is called the *column-parity kernel* (or *kernel* for short). Moreover, the set $\mathbb{F}_3^{729} \setminus K$ is denoted by N. A state belongs to the kernel if and only if all its columns have parity 0. Given a k-round trail core $\langle a_1, b_1, \ldots, a_{k-1}, b_{k-1} \rangle$, its *parity profile* is written $|X_1| \ldots |X_{k-1}|$ where $X_i = K$ if $b_i \in K$ and $X_i = N$ otherwise. The column parity mixer θ does not change the parity of the state : $P(A) = P(\theta(A))$. Indeed, $P(\theta(A))[x, z] = \sum_{y=0}^{2} A[x, y, z] + 3E(A)[x, z] \bmod 3 = P(A)[x, z]$. Therefore, $E(A) = E(\theta(A))$ and the inverse function of θ is given by

$$\theta^{-1}(A)[x, y, z] = A[x, y, z] - E(A)[x, z].$$

z-equivalence classes The four step functions χ, ρ_r, ρ_ℓ and θ are invariant with respect to the translation along the z-axis i.e. they commute with functions that translate the state in the z direction. Differential trails are thus considered up to translation along the z-axis.

4 General strategy for differential trail search [9]

Our objective is to analyze the resistance of Troika against differential cryptanalysis by providing a lower bound on the weight of any trail over a given number of rounds. We explain in this section the general strategy we followed for doing so. As Troika's round function has a similar design to Keccak-f, we naturally adopted the same strategy as the one used in [9]. This strategy permits to lower bound the weight of 6-round trail cores by extending trail cores on 3 rounds.

4.1 Overview of the steps

All 6-round trail cores of weight less than a bound W can be generated by
1. collecting all 3-round trail cores of weight up to $W/2$,
2. extending these 3-round trail cores by 3 rounds in the forward direction and in the backward direction.

Remark 2. In [9], it is sufficient to collect all the 3-round trail cores up to weight $\lfloor W/2 \rfloor$ since the weight is necessarily an integer in $\mathbb{F}_2$.

To cover the space of 3-round trail cores $a_1 \xrightarrow{\lambda} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda} b_2$ up to some weight, the search space is split according to the parity profile of the trail cores. Trail cores of parity profile $|K|K|$ can be generated efficiently (see Section 6.1). For trail cores of parity profile $|K|N|$, $|N|K|$ and $|N|N|$ we use the same steps as in [9], recalled in the following lemma.

Lemma 1. [9] *Let T_3 be the weight up to which we want to collect the 3-round trail cores and T_1 be a parameter that can be adjusted.*
All $|K|N|$-trail cores up to weight T_3 can be generated by
1. *collecting all $|K|$-trail cores $\langle a, b\rangle$ with $\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) \leq T_1$ and extending them forward outside the kernel,*
2. *collecting all $|N|$-trail cores $\langle a, b\rangle$ with $\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b) < T_3 - T_1$ and extending them backward inside the kernel.*

All $|N|K|$-trail cores up to weight T_3 can be generated by
1. *collecting all $|K|$-trail cores $\langle a, b\rangle$ with $\widetilde{\mathrm{w}}^{\mathrm{dir}}(b) \leq T_1$ and extending them backward outside the kernel,*
2. *collecting all $|N|$-trail cores $\langle a, b\rangle$ with $\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b) < T_3 - T_1$ and extending them forward inside the kernel.*

All $|N|N|$-trail cores up to weight T_3 can be generated by
1. *collecting all $|N|$-trail cores $\langle a, b\rangle$ with $2\,\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b) < T_3$ and extending them forward outside the kernel,*
2. *collecting all $|N|$-trail cores $\langle a, b\rangle$ with $\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + 2\,\widetilde{\mathrm{w}}^{\mathrm{dir}}(b) \leq T_3$ and extending them backward inside the kernel.*

To exhaustively generate the 2-round trail cores that respect the constraints of the above lemma, we use a tree traversal, as defined by Mella et al. in [9]. We recall this technique in Section 4.2 and define the specific trees used for Troika in Section 5. The extension phase is detailed in Section 6.2.

4.2 Generating 2-round trail cores as a tree traversal

For simplicity, we explain the method presented in [9] to study 2-round trail cores of weakly aligned primitives in the case of primitives defined on $\mathbb{F}_2^n$. This technique exploits the specific features of the linear layer to take simultaneously into account the weights $\widetilde{\mathrm{w}}^{\mathrm{rev}}(a)$ and $\widetilde{\mathrm{w}}^{\mathrm{dir}}(b)$ of a 2-round trail core $\langle a, b\rangle$.

More precisely, given a bound W and two weightings α and β, a tree traversal is performed to find all trail cores $\langle a, b\rangle$ satisfying $\alpha\,\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \beta\,\widetilde{\mathrm{w}}^{\mathrm{dir}}(b) \leq W$ and

possibly some other restrictions. We denote by $c_{\alpha,\beta}$ the function that associates to a pair $(a, b) \in (\mathbb{F}_2^n)^2$ the quantity $\alpha\, \widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \beta\, \widetilde{\mathrm{w}}^{\mathrm{dir}}(b)$. In the following, after having fixed a pair (α, β), we will call the quantity $c_{\alpha,\beta}(a, b)$ the cost of the trail core $\langle a, b\rangle$.

The idea is to generate trail cores of the form $\langle a, b\rangle$ by adding progressively to a variable $(a, b) \in (\mathbb{F}_2^n)^2$ initialised to $(0, 0)$ some vectors of $(\mathbb{F}_2^n)^2$. These vectors are called units and must be defined according to the distinctive features of the round function. The cost of the trail core $\langle a, b\rangle$ is checked each time a unit is added. This process is stopped when all the trail cores that could be obtained by adding more units would have a cost higher than W.

The tree traversal More formally, the algorithm is a tree traversal. A node of the tree is a list of units, called *unit-list*. A unit-list of the form $L = [u_1, \ldots, u_n]$ is associated to the vector $(a_L, b_L) \coloneqq \sum_{i=1}^{n} u_i$ and to the cost $c_{\alpha,\beta}(a_L, b_L)$. A unit-list L is said to be complete if $b_L = \lambda(a_L)$, that is to say if it is associated to a 2-round trail core. It is not always the case (see the tree defined in Section 5.3). The root of the tree is the empty unit-list. A child of a unit-list L is a unit-list of the form $L \mid\mid [u]$, where u is some particular unit that does not belong to L. To avoid having two unit-lists with the same units in a different order, an order relation on the units is chosen and units are only added in ascending order. This order relation organizes the edges of the tree and must therefore be chosen to ensure good properties of the tree.

During the tree traversal a lower bound on the cost of all of the current node's descendants is calculated. The tree is pruned if this lower bound is higher than W. The goal is to define the units and the tree organization in order to be able to prune the tree efficiently. There should be no descendant of a node L that has a cost much lower than this L's cost. A node should have as few brothers as possible.

Finally, if there exists an equivalence relation on trail cores – like the z-invariance – then it is possible to visit only one representative per equivalence class. The order relation on the units induces an order relation on the unit-lists, given by the lexicographic order. A unit-list is said to be *canonical* if it is the smallest representative of its class. We are interested in the set formed by all canonical unit-lists of cost below W. As the Lemma 1 of [9, Section 3.2] shows that a unit-list that is not canonical cannot have a canonical descendant, the tree is also pruned if a non canonical unit-list is encountered.

Lower bounding the cost To discuss how to give a lower bound on the costs of the descendants of a unit-list, we distinguish as in [5] between two types of active coordinates, the stable ones and the unstable ones. An active coordinate of (a_L, b_L) is said to be a *stable coordinate* of L if it is active for any pair $(a_{L'}, b_{L'})$ where L' is a descendant of L and *unstable* otherwise. By counting only the contribution of the stable coordinates of L we obtain a lower bound on the costs of the descendants of L.

This bound might be improved by studying the behavior of the unstable coordinates of L. For that, we search for some sets of passive or unstable coordinates that necessarily contribute to the cost of the descendants of L. We say that a subset I of the passive or unstable coordinates of (a_L, b_L) is an *activity invariant* of L if for every descendant L' of L there exists a coordinate $i \in I$ such that i is an active coordinate of $(a_{L'}, b_{L'})$. If all the coordinates of an activity invariant belong to boxes that have not already contributed to the bound, then this bound can be increased by adding a lower bound on the contribution of that activity invariant. This process is summarized in Algorithm 1 of Appendix A.

5 Generating 2-round trail cores in Troika

We present in this section the trees used to exhaustively generate the 2-round Troika's trail cores of a given parity profile that also respect a cost constraint.

The cost of a 2-round trail core $\langle a, b\rangle$ depends only on the positions of the active trits, but in some cases we will also need to know the exact difference. For this reason we introduce the notion of *mixed states*.

Definition 4. (Mixed state) *A* mixed state *is a three-dimensional array of size* $9 \times 3 \times 27$ *whose cells can take either the value* 0, 1, 2 *or* $\boxtimes$ *, where the value* $\boxtimes$ *stands for an active trit of unspecified value. It can also be seen as a vector of* $\{0, 1, 2, \boxtimes\}^{729}$.

We say that a difference $A \in \mathbb{F}_3^{729}$ is *compatible* with a parity plane p and a mixed state M if $P(A) = p$ and if for all coordinates (x, y, z), $A[x, y, z] = M[x, y, z]$ if $M[x, y, z] \in \{0, 1, 2\}$ and $A[x, y, z] \neq 0$ otherwise.

Addition of mixed states is defined as the component-wise addition. Only the commutative addition $0 + \boxtimes = \boxtimes$ and the usual addition modulo 3 can occur in our algorithms. The definitions of the trit shuffling ρ, the box weight and the cost functions $c_{\alpha,\beta}$ are extended to mixed states. We denote by $m_{(x,y,z)}$ the mixed state whose components are all zero, except the component of coordinates (x, y, z) that equals $\boxtimes$.

5.1 Generating $|K|$-trail cores

We define the tree used for the generation of trail cores $\langle \rho^{-1}(b), b\rangle$ with $b \in K$ and valid cost. As the difference b of such trail cores belongs to the kernel, its columns have either zero, two or three active coordinates. A node of the tree represents a couple of mixed states $(\rho^{-1}(M), M) \in \{0, \boxtimes\}^{729} \times \{0, \boxtimes\}^{729}$ where M has zero, two or three active coordinates per column. All nodes can be obtained by summing units of the form :

1. $\begin{smallmatrix}\square\\ \boxtimes\\ \boxtimes\end{smallmatrix}_{x,z} := \big(\rho^{-1}(m), m\big)$, where $m = m_{(x,0,z)} + m_{(x,1,z)}$,
2. $\begin{smallmatrix}\boxtimes\\ \square\\ \boxtimes\end{smallmatrix}_{x,z} := \big(\rho^{-1}(m), m\big)$, where $m = m_{(x,0,z)} + m_{(x,2,z)}$,

3. $\begin{smallmatrix}\boxtimes\\\boxtimes\\\square\end{smallmatrix}_{x,z} := \big(\rho^{-1}(m), m\big)$, where $m = m_{(x,1,z)} + m_{(x,2,z)}$,

4. $\begin{smallmatrix}\boxtimes\\\square\end{smallmatrix}_{x,z} := \big(\rho^{-1}(m_{(x,2,z)}), m_{(x,2,z)}\big)$.

Units are characterized by their coordinates (x, z) and their patterns $\begin{smallmatrix}\square\\\boxtimes\\\boxtimes\end{smallmatrix}$, $\begin{smallmatrix}\boxtimes\\\square\\\boxtimes\end{smallmatrix}$, $\begin{smallmatrix}\boxtimes\\\boxtimes\\\square\end{smallmatrix}$ or $\begin{smallmatrix}\boxtimes\\\square\\\square\end{smallmatrix}$. Their ordering is arbitrary. We just need to have $\begin{smallmatrix}\square\\\boxtimes\\\boxtimes\end{smallmatrix}_{x,z} < \begin{smallmatrix}\boxtimes\\\square\\\square\end{smallmatrix}_{x,z}$ for all x and z.

Units have to be added to the unit-list in ascending order while respecting two constraints. A unit of type 4 can only be added to a unit-list that already contains the unit of type 1 of same coordinates (x, z). A unit of types 1, 2 or 3 cannot be added if there is already in the unit-list another unit of same coordinates (x, z). Thereby, the mixed state M of a node $(\rho^{-1}(M), M)$ cannot have a column with only one active coordinate. Moreover, this ensures that all coordinates of a unit-list are stable and consequently that the cost function $c_{\alpha,\beta}$ is monotonic with respect to units addition.

To obtain the differences b of the desired trail cores $\langle \rho^{-1}(b), b\rangle$, it just remains to generate, for each node $(\rho^{-1}(M), M)$ reached during the tree traversal, the differences compatible with the zero parity plane and the mixed state M.

5.2 Reducing the problem of generating $|N|$-trail cores to that of generating parity-bare states at the input of θ

For the generation of trail cores of the form $\langle \rho^{-1}(c), \theta(c)\rangle$ with $c \in N$, our approach is the same as in [9]. Unlike the generation of $|K|$-trail cores, we were not able to define a tree whose nodes have only stable coordinates. The difficulty comes from the column parity mixer : activating trits at the input of θ can deactivate trits at the output of the function and vice versa. The rationale behind the tree's units and the order relation among them is to gradually choose the columns' values on either side of the map θ in an appropriate order that limits the number of unstable coordinates.

As in [9], we reduce the problem of that of finding a good tree containing all trail cores $\langle \rho^{-1}(c), \theta(c)\rangle$ such that c is a *parity-bare* state. That means that the unaffected columns of c have as few active trits as possible for their parity, namely zero for columns of parity zero and one for the other unaffected columns. Once the parity-bare states are generated, we no longer have to care about the θ-effect and can define trees as in Section 5.1 to recover from the parity-bare states all the wanted $|N|$-trail cores. Indeed, any out-kernel state can be obtained by summing a parity-bare state and an in-kernel state that only adds extra active coordinates to the unaffected columns of the initial parity-bare state. We refer to Appendix C for a detailed description of the tree used to add extra active coordinates in the unaffected columns of a parity-bare state.

Example 1. Replacing an unaffected column $\begin{array}{c}\square\\ \square\\ \square\\ \boxed{1}\end{array}$ of a parity-bare state c by one of the columns $\begin{array}{c}\square\\ \square\\ \square\\ \boxed{2}\end{array}$, $\begin{array}{c}\square\\ \square\\ \boxed{2}\end{array}$, $\begin{array}{c}\square\\ \boxed{1}\\ \boxed{2}\end{array}$, $\begin{array}{c}\square\\ \boxed{2}\\ \boxed{1}\end{array}$, $\begin{array}{c}\square\\ \boxed{1}\\ \boxed{1}\end{array}$ does not decrease the cost of the trail core $\langle \rho^{-1}(c), \theta(c)\rangle$.

5.3 Generating parity-bare states at the input of θ

We now present the tree used to organize the trail cores $\langle \rho^{-1}(c), \theta(c)\rangle$ such that c is a parity-bare state. The idea is to choose the columns of the difference c in an order that makes it possible to control the parity plane and the θ-effect plane of c.

To describe more easily the tree's units and the order relation among them, we introduce the notion of *diagonal coordinate* d of a column. The (x, z)-coordinates and (d, z)-coordinates of a column are related by the change of coordinates $(x, z) = (d + 2z \bmod 9, z)$. In the following, the letter d will always be an integer taken modulo 9 that refers to a diagonal coordinate. The coordinates of a column are implicitly given using the (d, z)-coordinates.

General form of units A unit of the tree is an element of $\mathbb{F}_3^{729} \times \mathbb{F}_3^{729}$ whose values at the input and output of θ correspond to the values of a column, affected or not, before and after θ. More formally, a unit is a pair of the form

$$\begin{pmatrix} t_2 \\ t_1 \\ t_0 \end{pmatrix}_{x,z,t} := \left(\rho^{-1}\left(\sum_{i=0}^{2} t_i e_{(x,i,z)} \right), \sum_{i=0}^{2} (t_i + t) e_{(x,i,z)} \right).$$

It is characterized by its (x, z)-coordinates, by a triplet $(t_0, t_1, t_2) \in \mathbb{F}_3^3$ that is the column value at the input of θ and by a trit t that is the θ-effect applied to obtain the column value at the output of θ. To obtain a pair of the form $(\rho^{-1}(c), \theta(c))$ by summing units, the parity-plane and the θ-effect plane formed at the input of θ have to be consistent.

Runs and loops Similar to what is done for Keccak [2] and Xoodoo [5], to link the parity plane and the θ-effect of a state, we group all its columns of non-zero parity in sets called *runs* and *loops*. The following definitions are motivated by the fact that the θ-effect applied to a column of coordinates $(d_0 + 1, z_0)$ depends on the parity of the columns of coordinates (d_0, z_0) and $(d_0, z_0 + 1)$.

Definition 5. *Let A be a state. For an integer $\ell \in [1, 26]$ and some coordinates (d_0, z_0), we consider the set r formed by the columns of A of coordinates $(d_0, z_0), \ldots, (d_0, z_0 + \ell - 1)$. If all the columns of r have non-zero parity and if the two columns of coordinates $(d_0, z_0 - 1)$ and $(d_0, z_0 + \ell)$ have parity zero, we say that r is the* run *of A of coordinates (d_0, z_0) and of length ℓ.*

Definition 6. *If a state admits a set of columns of coordinates* $(d_0, 0)$*, …,* $(d_0, 26)$ *that all have non-zero parity, we say that this set is a* loop.

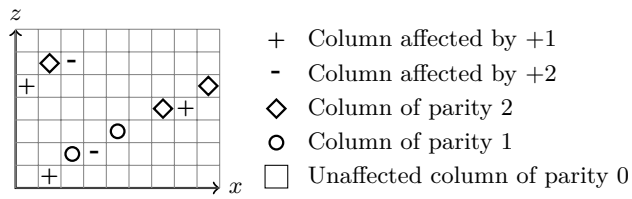

Fig. 1. Parity and θ-effect planes of a run of (d, z)-coordinates $(0, 1)$ and of length 5

The set formed by all the non-zero parity columns of a state can be partitioned into runs and loops. Each run or loop determines a part of the θ-effect plane. More precisely, if a state A admits a run of coordinates (d_0, z_0) and of length ℓ, then this run determines for all $i \in [-1, \ell - 1]$, the value of $E(A)[d_0 + 1, z_0 + i]$. The columns of coordinates $(d_0 + 1, z_0 - 1)$ and $(d_0 + 1, z_0 + \ell - 1)$ are always affected.

From supra-units to units Since a trail core $\langle \rho^{-1}(c), \theta(c) \rangle$ formed from a parity-bare state c that has a loop has a cost too high for our search, we restrict the nodes of the tree to trail cores formed from parity-bare states that have only runs. However, the existence of loops for Troika, is a consequence of the particular θ mapping used in Troika and this effect does not appear in Keccak. The existence of these loops is due to the fact that all the diagonal coordinates form 9 disjoint sets. We use this property later to improve the lower bound on the cost of a trail.

Every parity-bare state with n runs can be decomposed into n parity-bare states with only one run. These latter have only unaffected null columns, unaffected columns with a single active trit and affected columns of parity zero. Such a decomposition is not unique. To make it unique, we adopt the same conventions as in [2] and [5]. The runs of the decomposition must not overlap. Moreover, if the initial state admits affected columns of non-zero parity, these columns have to be decomposed into an unaffected column with a single active trit at $y = 0$, whose value is the parity of the initial column, and an affected column chosen accordingly. We call this rule the non-zero-$y0$ convention. Under these conventions, the existence and the uniqueness of the decomposition can be proved as in [5, Section 7.3.2] by exhibiting a deterministic procedure that removes from a parity-bare state with n runs a parity-bare state with one run to get a parity-bare state with $n - 1$ runs. It removes the columns of a run as well as the columns affected by this run, using the non-zero-$y0$ convention to split the affected columns of non-zero parity and only remove the suitable part of such columns.

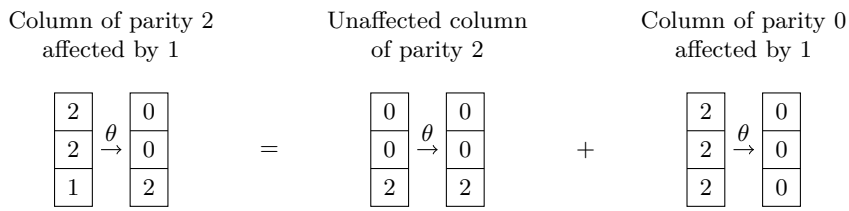

Fig. 2. Decomposition of an affected column of non-zero parity using the non-zero-$y0$ convention

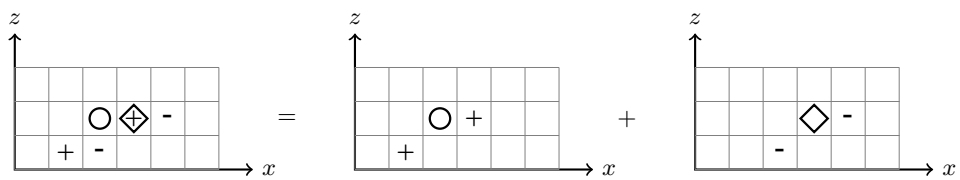

Fig. 3. Parity and θ-effect planes of a parity-bare state with 2 runs and its decomposition into parity-bare states with 1 run. The affected column of non-zero parity has to be split according to the non-zero-$y0$ convention. The symbols are explained in Figure 1

All the sought trail cores can be expressed as a sum of elements of $\mathbb{F}_3^{729} \times \mathbb{F}_3^{729}$ of the form $\big(\rho^{-1}(A), \theta(A)\big)$ where A is a parity bare state with only one run. We call these elements *supra-units*. Since there are too many possible supra-units, they cannot directly be used as units. A finer subdivision is therefore used. It relies on the units introduced before, that enable to form the supra-units one by one. With this refinement, many supra-units have a common ancestor, which is important for the tree pruning.

We define a partial order relation on the supra-units. For that, we characterise a supra-unit $\big(\rho^{-1}(A), \theta(A)\big)$ with the coordinates (d, z) and the length ℓ of the run of A. The partial order is given by the lexicographic order on $[d, z, \ell]$ and supra-units are put in the unit-list in ascending order.

Two types of units are used. Those that are said to be affected correspond to affected columns of parity zero at the input of θ ; those that are said to be unaffected correspond to unaffected columns with only one active trit at the input of θ. Each supra-unit of coordinates (d, z) and length ℓ is the sum of :

- ℓ unaffected units with (d, z)-coordinates (d, z), ..., $(d, z + \ell - 1)$,
- 2 affected units of coordinates $(d + 1, z - 1)$ and $(d + 1, z + \ell - 1)$,
- from 0 to $\ell - 1$ other affected units that could be in positions $(d + 1, z)$, ..., $(d + 1, z + \ell - 2)$.

These units are indexed from 0 to 2ℓ and added to the unit-list in ascending order of indexes. The i-th unit has coordinates $(d + 1, z - 1 + \frac{i}{2})$ if i is even and $(d, z + \lfloor \frac{i}{2} \rfloor)$ if i is odd.

Importance of the ordering of supra-units The ordering of units resembles the one of [9], except that here, the partial order on supra-units allows us to detect more active coordinates that are stable for a unit-list. This results in an improvement of the bound computed with Algorithm 1 (given in Appendix A). We were able to increase the 3-round weight target from 33 to 41.

The only way for active coordinates of a unit-list to turn passive is when an unaffected unit is added to a unit-list that already contains an affected unit of same coordinates, or vice versa. By the non-zero-$y0$ convention, the only overlaps allowed are of the form :

$$\begin{pmatrix} t_2 \\ t_1 \\ t_0 \end{pmatrix}_{x,z,t} + \begin{pmatrix} 0 \\ 0 \\ t'_0 \end{pmatrix}_{x,z,0}, \text{ where } t_0 + t_1 + t_2 = 0,\ t \in \{1,2\} \text{ and } t'_0 \in \{1,2\}.$$

Let L be a unit-list and $(a_L, b_L) = \sum_{u \in L} u$. If a unit-list has only one of the two terms of the above sum, then the trit of a_L of coordinates $\rho^{-1}(x,0,z)$ and the trit of b_L of coordinates $(x,0,z)$ may be unstable. In the particular case of Troika, this criterion can be refined by locating columns that can no longer be concerned by an overlap and thus have only stable coordinates. The main remark is that overlaps only involve affected units of a supra-unit of diagonal coordinate d and unaffected units of a supra-unit of diagonal coordinate $d+1$. The ordering of supra-units implies that unaffected units that belong to a supra-unit of diagonal coordinate $d > 0$ are not concerned anymore by overlaps since supra-units of d-coordinate $d-1$ can no longer be added to the unit-list. Similarly, if we write d_{last} for the d-coordinate of the supra-unit that is being formed or that just has been completed, then all the affected units belonging to a supra-unit of coordinate $d < d_{\text{last}} - 1$ cannot be involved in a new overlap.

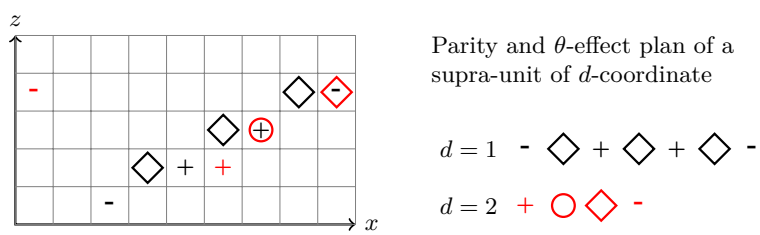

Fig. 4. Columns overlapping

Lower bounding the cost We use Algorithm 1 to compute a lower bound on the costs of node's descendants. The criterion used to distinguish between stable and unstable coordinates of a unit-list L is explained above. It remains to describe the activity invariants of L. Suppose that the difference $\rho(a_L)$ has an affected column of coordinates (x,z). For every descendant L' of L, this column

remains affected and thus $a_{L'}[\rho^{-1}(x,0,z)] \neq b_{L'}[x,0,z]$. In particular, at least one of these two trits is active. Suppose now that the difference $\rho(a_L)$ has an unaffected column with only one active trit of coordinates $(x,0,z)$. Whether or not this column becomes affected for a descendant L' of L, there will always be an active trit among the set $\{a_{L'}[\rho^{-1}(x,0,z)],\ b_{L'}[x,0,z]\}$. Moreover, since the columns of coordinates (x,z) at the input and output of θ will always have a non-zero parity, the same is true for the sets $\{a_{L'}[\rho^{-1}(x,i,z)] : i \in [0,\,2]\}$ and $\{b_{L'}[x,i,z] : i \in [0,\,2]\}$.

An improvement for the tree traversal The fact that we are working over $\mathbb{F}_3$ does not facilitate the tree traversal. If we entirely specify the value of the differences for all the active trits, there will be too many possibilities to treat. But, if we do not specify the value, the behaviour of the column parity mixer θ is too complicated to understand. To deal with this situation we introduce an equivalence relation on parity-bare states used to reduce the search space during the tree traversal.

Definition 7. (Components of a parity-bare state) *Let A be a parity-bare state with n runs and $A = \sum_{i=1}^{n} A_i$ its decomposition into parity-bare states with one run defined in Section 5.3. We form a graph $\mathcal{G}_A$ whose vertices are numbered from 1 to n. The i-th and j-th vertices are connected by an edge if and only if the parity-bare state $A_i + A_j$ has an affected column of non-zero parity. If this graph admits k connected components $C_1, \ldots, C_k$, then we call the elements of the set $\{\sum_{j \in C_i} A_j : i \in [1,\,k]\}$ the* components *of A.*

Definition 8. (Scalar-equivalence) *Let A^1 and A^2 be two parity-bare states that have k components $C_1^1, \ldots, C_k^1$ and $C_1^2, \ldots, C_k^2$. The parity-bare states A^1 and A^2 are* scalar-equivalent *if there exists some scalars $\lambda_1, \ldots, \lambda_k \in \{1, 2\}$ and a permutation σ of the set $\{1, \ldots, k\}$ such that $C_i^1 = \lambda_i C_{\sigma(i)}^2$ for all $i \in [1,\,k]$.*

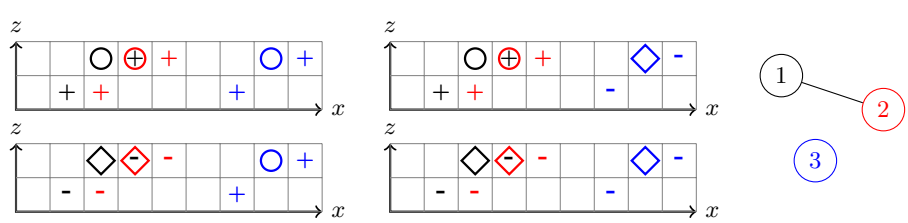

Fig. 5. Parity planes and θ-effect planes of four parity-bare states that are scalar-equivalent and their associated graph.

We say that two complete unit-lists L and L' are scalar-equivalent if the differences $\rho(a_L)$ and $\rho(a_{L'})$ are scalar-equivalent. Two complete scalar-equivalent

unit-lists describe trail cores that have the same trit-activity pattern. We organize the tree to have only one representative per equivalence class. At the end of the tree traversal, we generate from each representative the elements of its class.

6 Dealing with $|K|K|$ profile and extensions

6.1 Generating 3-round trail cores of parity profile $|K|K|$

The space of 3-round trail cores with parity profile $|K|K|$ can be directly scanned up to some limit weight. We search for trail cores of the form $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2)\rangle$ with $b_1 \in K$ and $\rho(a_2) \in K$.

The first step is to generate a subset $\overline{A_2} \subset \mathbb{F}_2^{729}$ that contains all trit activity patterns that are good candidates for $\overline{a_2}$. For that, we derive necessary conditions on $\overline{a_2}$ from the fact that the differences b_1 and $\rho(a_2)$ must be in the kernel. More precisely, if $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2)\rangle$ is a trail core of parity profile $|K|K|$, then a column of $\rho(a_2)$ cannot have exactly one active trit. Moreover, as $b_1 \in K$, a box-column of b_1 cannot have exactly one active box. Due to the fact that the differences b_1 and a_2 have the same box activity pattern, a box-column of a_2 cannot have exactly one active box. Thereby, we deduce the two following necessary conditions on $\overline{\rho(a_2)}$ and on $\overline{a_2}$:

- If $\overline{\rho(a_2)}[x, y, z] = 1$, then $\overline{\rho(a_2)}[x, y + 1, z] = 1$ or $\overline{\rho(a_2)}[x, y + 2, z] = 1$;
- If $\overline{a_2}[x, y, z] = 1$, then it exists $i \in \{0, 1, 2\}$ and $j \in \{1, 2\}$ such that $\overline{a_2}[3x_B + i, y + j, z] = 1$, where $x_B = \lfloor \frac{x}{3} \rfloor$.

To take into account the weight constraint, we compute only from the trit activity pattern $\overline{a_2}$ of a difference a_2, a lower bound on the weight of trail cores of the form $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2)\rangle$, denoted by $\widetilde{\mathrm{w}}_{\min}^{\text{3-round}}(\overline{a_2})$. The set $\overline{A_2}$ is formed by all the trit activity patterns $\overline{a_2}$ that respect the necessary conditions and that lead to a bound $\widetilde{\mathrm{w}}_{\min}^{\text{3-round}}(\overline{a_2})$ below the limit weight. We refer to Appendix D for a complete description of the method to form the subset $\overline{A_2}$.

The second step is to generate all $(|K|K|, W)$-trail cores, by extending backward inside the kernel (if possible) the 2-round trail cores $\langle a_2, \rho(a_2)\rangle$ that satisfy $\overline{a_2} \in \overline{A_2}$ and $\rho(a_2) \in K$. The details of the extension algorithm are given in Appendix E.1.

6.2 Extension of trail cores

Once all trail-cores of a certain length have been obtained, these must be extended. Extensions are treated differently depending on their direction (backward or forward) and their type (inside or outside the kernel).

For the forward extension, the authors of [9] exploited the fact that the Keccak S-box is quadratic, a property that does not hold for Troika. To extend a trail core outside the kernel in the forward and backward direction or inside the kernel in the backward direction we use a step-by-step approach to control the

extensions' weight. Our approach shares some similarities with the method of [9] but is necessarily different because the inner components of the two functions are not the same.

Because of the page restriction all our methods used for extending the obtained trail cores are given in Appendix E.

7 Results and conclusion

We have covered the space of all 3-round trail cores up to weight 41 and showed that there exists no differential trail over six rounds with weight below 82. It results in a better bound than the one obtained with the MILP-based approach. Adapting the framework of [9] for the field $\mathbb{F}_3$ was harder than expected. Dealing with trits rather than bits complicates the analysis and we were not able to take a cost target for the search as high as for Keccak.

Our experimental results have been obtained on a Intel Core i5 processor running at 2.4Hz and a single core was used for each experiment. Our code is publicly available[1]. We assume that we cannot do considerably better as the number of trails to scan would rapidly be a limiting factor. This is why we did not try to run the code longer even though it may be possible to get a slightly better bound.

For the generation of 3-round trail cores of parity profile $|K|K|$, we were able to cover the space up to weight 65. This search takes advantage of the small size of the S-boxes, which gives restrictive conditions on the box-activity pattern of an in-kernel difference. Figure 6 reports the number of found $|K|K|$ trail cores per weight.

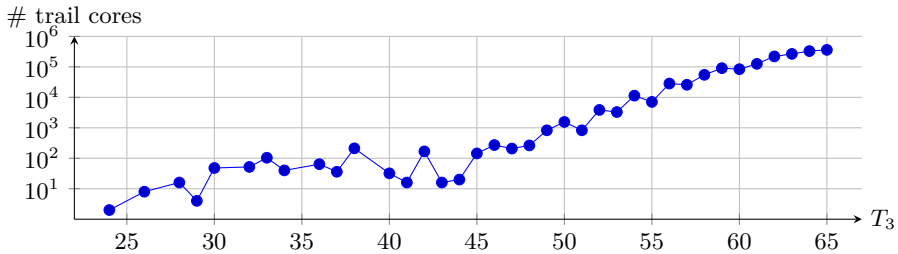

Fig. 6. Number of $|K|K|$ trail cores of weight W such that $\lceil W \rceil = T_3$

For the generation of 3-round trail cores with parity profile $|K|N|$, $|N|K|$ and $|N|N|$, increasing the target weight drastically increases the number of 2-round trail cores to investigate and extend into 3-round trail cores. Figure 7 reports the number of 2-round trail cores that have to be collected in order to apply Lemma 1 with $T_3 = 41$ and $T_1 = 11$.

[1] https://github.com/MargotFunk/troikaDifferentialCryptanalysis

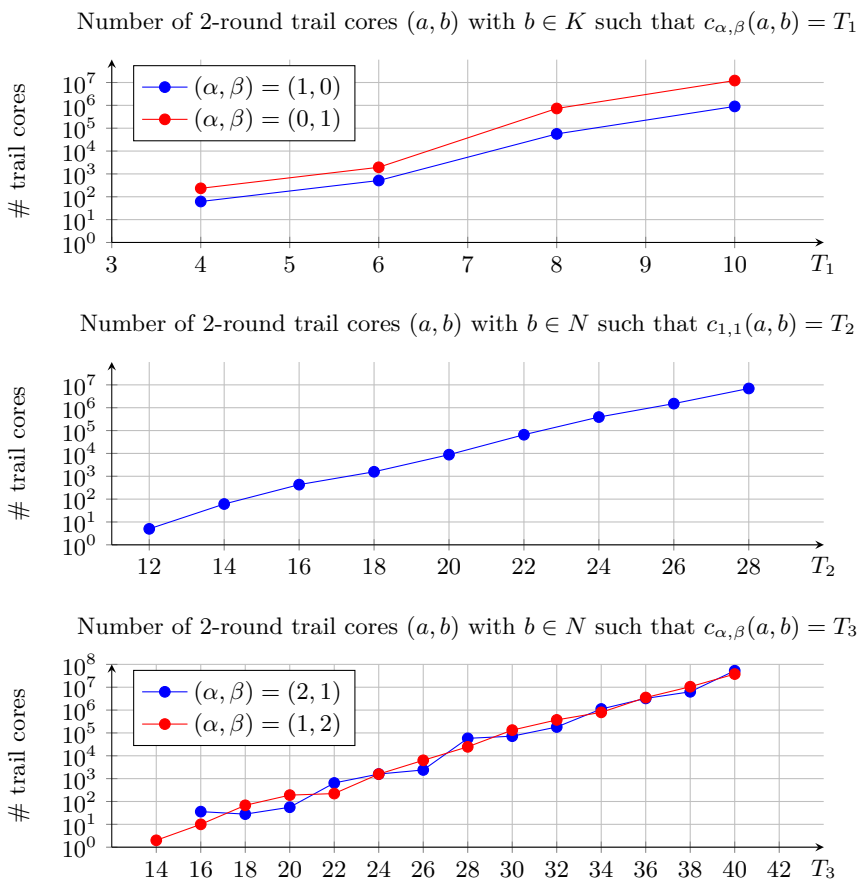

Fig. 7. Number of 2-round trail cores for different cost functions. Trail cores are considered in the three graphs up to z-invariance. In the second graph, trail cores are also considered up to scalar-equivalence.

The execution time for the generation of all 3-round trail cores for the four different parity profiles is given below.

Parity profile	Direction	Time	Parity profile	Direction	Time
$\|K\|K\|$	backward	22m40s	$\|K\|N\|$	forward backward	5m7s 5h19m
$\|N\|N\|$	forward backward	9h16m 17h7m	$\|N\|K\|$	forward backward	6h32m 26m10s

Finally, Figure 8 gives the weight distribution of the 3-round trail cores up to weight $T_3 = 41$ depending on the parity profile of the trail cores.

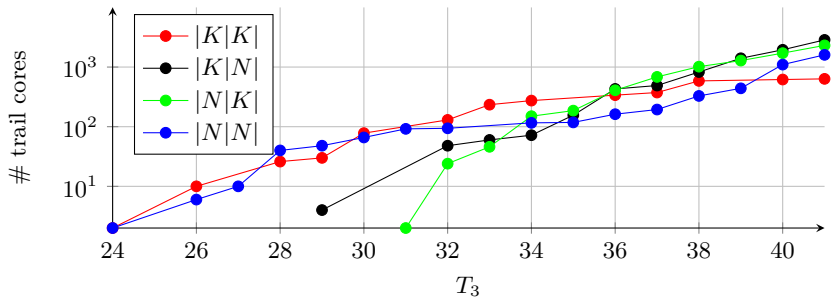

Fig. 8. Number of 3-round trail cores of weight W such that $\lceil W \rceil \leq T_3$ for different parity profiles.

References

1. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: On the indifferentiability of the sponge construction. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer (2008)
2. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak SHA-3 submission (January 2011), http://sponge.noekeon.org/, submission to NIST (Round 3)
3. Bordes, N., Daemen, J., Kuijsters, D., Assche, G.V.: Thinking outside the Superbox. In: CRYPTO, Part III. LNCS, vol. 12827, pp. 337–367. Springer (2021). https://doi.org/10.1007/978-3-030-84252-9_12, https://doi.org/10.1007/978-3-030-84252-9_12
4. Daemen, J., Assche, G.V.: Differential propagation analysis of Keccak. In: FSE 2012. LNCS, vol. 7549, pp. 422–441. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_24, https://doi.org/10.1007/978-3-642-34047-5_24
5. Daemen, J., Hoffert, S., Assche, G.V., Keer, R.V.: The design of Xoodoo and Xoofff. IACR Trans. Symmetric Cryptol. **2018**(4), 1–38 (2018). https://doi.org/10.13154/tosc.v2018.i4.1-38, https://doi.org/10.13154/tosc.v2018.i4.1-38
6. Debris-Alazard, T., Sendrier, N., Tillich, J.: Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In: ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 21–51. Springer (2019)
7. Heilman, E., Narula, N., Tanzer, G., Lovejoy, J., Colavita, M., Virza, M., Dryja, T.: Cryptanalysis of Curl-P and other attacks on the IOTA cryptocurrency. IACR Trans. Symmetric Cryptol. **2020**(3), 367–391 (2020), https://doi.org/10.13154/tosc.v2020.i3.367-391
8. Kölbl, S., Tischhauser, E., Derbez, P., Bogdanov, A.: Troika: a ternary cryptographic hash function. Designs, Codes and Cryptography **88**, 91–117 (Jan 2019). https://doi.org/10.1007/s10623-019-00673-2
9. Mella, S., Daemen, J., Assche, G.V.: New techniques for trail bounds and application to differential trails in keccak. IACR Trans. Symmetric Cryptol. **2017**(1), 329–357 (2017). https://doi.org/10.13154/tosc.v2017.i1.329-357, https://doi.org/10.13154/tosc.v2017.i1.329-357
10. Stoffelen, K., Daemen, J.: Column parity mixers. IACR Transactions on Symmetric Cryptology **2018**(1), 126–159 (Mar 2018), https://tosc.iacr.org/index.php/ToSC/article/view/847

A Algorithm to give a lower bound on the costs of a unit-list and its descendants

We recall here the algorithm used in [5] to give a lower bound on the costs of a unit-list and its descendants. To index the coordinates of an element of $\mathbb{F}_2^n \times \mathbb{F}_2^n$, we distinguish between the set of coordinates of the first component, denoted by C_a, and the set of coordinates of the second component, denoted by C_b. Let L be a unit-list and $(a_L, b_L) \coloneqq \sum_{u \in L} u$. We denote by $A_L = S_L \cup U_L$ the set of active coordinates of (a_L, b_L), where S_L is the set of stable coordinates of L and U_L is the set of unstable coordinates of L.

Algorithm 1: Bound on the costs of the descendants of a unit-list

input : A unit-list L
output: A lower bound on the elements of the set
$\{\alpha\, \widetilde{\mathrm{w}}^{\mathrm{rev}}(a_{L'}) + \beta\, \widetilde{\mathrm{w}}^{\mathrm{dir}}(b_{L'}) : L' \text{ is a descendant of } L\}$

$\mathrm{w}_a \leftarrow$ the smallest minimum reverse weight of all differences with one active box
$\mathrm{w}_b \leftarrow$ the smallest minimum direct weight of all differences with one active box

`/* variable used to avoid counting more than once an active box */`
$(\Delta_a, \Delta_b) \leftarrow (0, 0) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$
Activate all coordinates of (Δ_a, Δ_b) that are stable coordinates of L.

`/* contribution of stable coordinates */`
$\mathsf{bound} \leftarrow \alpha\, \mathrm{w}_a\, \mathrm{w}_{\mathrm{box}}(\Delta_a) + \beta\, \mathrm{w}_b\, \mathrm{w}_{\mathrm{box}}(\Delta_b)$
`/* contribution of unstable coordinates */`
forall $u \in U_L$ **do**
  if *there exists a small activity invariant I_u of L containing u* **then**
    forall $i \in I_u$ **do** Note the contribution c_i of i

$$c_i \leftarrow \begin{cases} 0 & \text{if } i \in C_a \text{ and } i \text{ belongs to an active box of } \Delta_a \\ 0 & \text{if } i \in C_b \text{ and } i \text{ belongs to an active box of } \Delta_b \\ \alpha\, \mathrm{w}_a & \text{if } i \in C_a \text{ and } i \text{ belongs to a passive box of } \Delta_a \\ \beta\, \mathrm{w}_b & \text{if } i \in C_b \text{ and } i \text{ belongs to a passive box of } \Delta_b \end{cases}$$

    `/* Take the minimum of the contributions */`
    $c \leftarrow \min\{c_i : i \in I_u\}$
    if $c > 0$ **then**
      $\mathsf{bound} \leftarrow \mathsf{bound} + c$
      Activate all the coordinates of (Δ_a, Δ_b) that belong to I_u.

return bound

B Different parts of the state of Troika

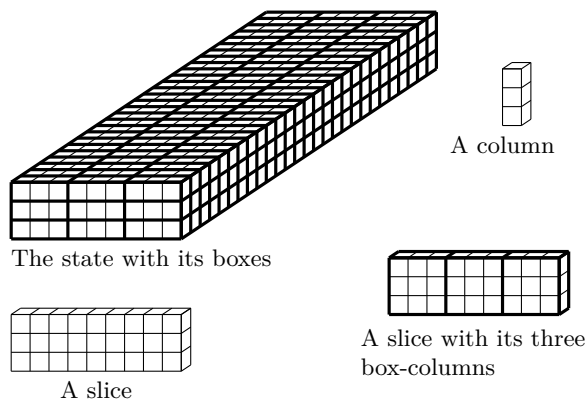

Fig. 9. Different parts of the state of Troika

C Tree to complete the columns of a parity-bare state

Let c be a parity-bare state associated to a trail core $\langle \rho^{-1}(c), \theta(c) \rangle$ of valid cost. We arrange in a tree all the trail cores $\langle \rho^{-1}(c^*), \theta(c^*) \rangle$ that correspond to a difference c^* that has the same parity plane and affected columns as c and possibly extra active coordinates in its unaffected columns.

We form, from the initial parity-bare state c, the mixed state M_c defined as

$$M_c[x,y,z] = \begin{cases} c[x,y,z] & \text{if } E(c)[x,z] \neq 0 \\ \boxtimes & \text{if } E(c)[x,z] = 0 \text{ and } c[x,y,z] \neq 0 \\ 0 & \text{otherwise} \end{cases} .$$

For simplicity we describe the columns of the mixed state M_c using the same attributes as those used to qualify the parity-bare state c from which they come from (affected or unaffected,parity 0, 1 or 2). The unaffected columns of M_c of parity 0 are passive while those of parity 1 or 2 have only one active coordinate.

The unit $(\rho^{-1}(M_c), M_c)$ is the first unit of the unit-list. Other units are added to complete the unaffected columns of M_c as represented in Figure 10. They are of types 1, 2, 3, 4 (see Section 5.1) or

5. $\begin{smallmatrix}\square\\ \boxtimes\\ \square\end{smallmatrix}_{x,z} \coloneqq \big(\rho^{-1}(m_{(x,1,z)}), m_{(x,1,z)}\big)$.

Finally, we generate the trail cores $\langle \rho^{-1}(c^*), \theta(c^*) \rangle$ from the differences c^* that are compatible with the parity plane $P(c)$ and with the descendants of M_c reached during the tree traversal.

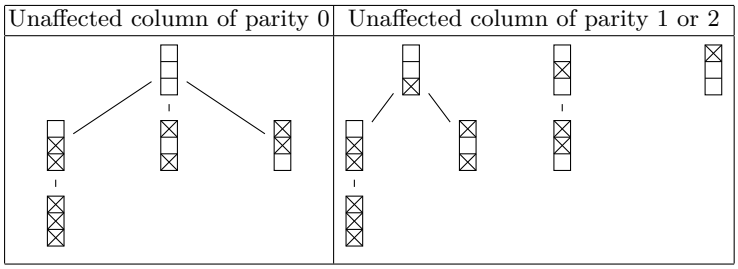

Fig. 10. How to complete the unaffected columns of the mixed state M_c

D First step of the $|K|K|$-trail cores generation

We describe here the algorithm used to generate all the trit activity patterns $\overline{a_2}$ of the wanted $|K|K|$-trail cores $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2)\rangle$.

We introduce a graph $\mathcal{G}$ to organize the necessary conditions on $\overline{\rho(a_2)}$ and $\overline{a_2}$. We denote by $v_{(x,y,z)}$ the pair of coordinates $\big((x,y,z), \tilde{\rho}(x,y,z)\big)$. The set V defined as $V \coloneqq \{v_{(x,y,z)} : (x,y,z) \in [0,\,8] \times [0,\,2] \times [0,\,26]\}$ is the set of vertices of the graph $\mathcal{G}$. A vertex of $\mathcal{G}$ represents a trit seen both from the point of view of $\overline{a_2}$ and $\overline{\rho(a_2)}$. The graph $\mathcal{G}$ has two types of edges, one to visualize the necessary conditions on $\overline{a_2}$, the other to deal with the conditions on $\overline{\rho(a_2)}$. A vertex $v_{(x,y,z)} \in V$ has 6 neighbors by constraints on $\overline{a_2}$, given by the set $N_{a_2}\left(v_{(x,y,z)}\right) \coloneqq \{v_{(3x_B+i,y+j,z)} : x_B = \lfloor \frac{x}{3} \rfloor,\ i \in \{0,1,2\},\ j \in \{1,2\}\}$. It also has 2 neighbors by constraints on $\overline{\rho(a_2)}$, given by the set $N_{\rho(a_2)}\left(v_{(x,y,z)}\right) \coloneqq \{v_{\tilde{\rho}^{-1}(\tilde{\rho}(x,y,z)+(0,i,0))} : i \in \{1,2\}\}$.

We consider that each vertex of the graph $\mathcal{G}$ can be either active or passive. We say that a subset of V that indicates the active vertices of the graph $\mathcal{G}$ is an activity pattern. An activity pattern $S \subset V$ of the graph $\mathcal{G}$ is valid if it corresponds to a pair $(\overline{a_2}, \overline{\rho(a_2)})$ that respects all the necessary conditions on $\overline{a_2}$ and $\overline{\rho(a_2)}$, namely if for all $v \in S$, $N_{a_2}(v) \cap S \neq \emptyset$ and $N_{\rho(a_2)}(v) \cap S \neq \emptyset$. Two activity patterns S and S' are equivalent if there exists $z \in [0,\,26]$ such that $S' = \{v + \big((0,0,z),(0,0,z)\big) : v \in S\}$. Our goal is to generate all equivalence classes of valid activity patterns of the graph $\mathcal{G}$ that do not activate too many S-boxes.

This selection of activity patterns of the graph $\mathcal{G}$ is done with a traversal of a tree $\mathcal{T}$. A node of this tree describes an activity pattern of $\mathcal{G}$ by listing all the vertices of $\mathcal{G}$ that belong to this activity pattern.

We introduce some notations to describe the tree $\mathcal{T}$. A node $L = [v_1, \ldots, v_n]$ of $\mathcal{T}$ defines an activity pattern $S^{(L)} \coloneqq \{v_1, \ldots, v_n\}$ of the graph $\mathcal{G}$ and a pair of trit activity patterns denoted by $\left(\overline{a_2}^{(L)}, \overline{\rho(a_2)}^{(L)}\right)$. If L and L' are two lists, we write $L \mid\mid L'$ the concatenation of L and L'. We use the following order relation on V : for all $v_{(x,y,z)},\ v_{(x',y',z')} \in V$, $v_{(x,y,z)} < v_{(x',y',z')}$ if and only if $(x,y,z) <_{\text{lex}} (x',y',z')$.

The root of $\mathcal{T}$ is the empty list. Children of a node of $\mathcal{T}$ are given by Algorithm 2, in order to have in the tree all the desired activity patterns.

Algorithm 2: Children of a node of the tree $\mathcal{T}$

input : A node L of the tree $\mathcal{T}$
output: The set composed of all children of L

if $L = [\ ]$ **then**
  return $\{\ [v_{(x,y,0)}]\ \}_{(x,y)\in[0,8]\times[0,2]}$;
if $L = [v_1, \ldots, v_n]$ **then**
  $i \leftarrow \max\left\{\{1\} \cup \{1 \leq i \leq n : \bigcup_{j=1}^{i-1}\{v_j\} \text{ is valid}\}\right\}$
  if $N_{a_2}(v_n)) \cap S^{(L)} = \emptyset$ **then**
    $N \leftarrow N_{a_2}(v_n)$
  else if $N_{\rho(a_2)}(v_n) \cap S^{(L)} = \emptyset$ **then**
    $N \leftarrow N_{\rho(a_2)}(v_n)$
  else if $N_{a_2}(v_i) \cap S^{(L)} = \emptyset$ **then**
    $N \leftarrow N_{a_2}(v_i)$
  else if $N_{\rho(a_2)}(v_i) \cap S^{(L)} = \emptyset$ **then**
    $N \leftarrow N_{\rho(a_2)}(v_i)$
  else $S^{(L)}$ is valid
    $N \leftarrow \{v : v \in V \setminus S^{(L)}\}$
  if $N \cap \{v \in V : v_i < v\} = \emptyset$ **then**
    return $\emptyset$
  else
    return $\{L \mid\mid [v] : v \in N \cap \{v \in V : v_i < v\}\}$

D.1 Proof that all desired activity patterns are in the tree

We want to show that if $S \subset V$ is a valid activity pattern of the graph $\mathcal{G}$, then it exists a node L of the tree $\mathcal{T}$ such that $S^{(L)}$ is equivalent to S.

Let n be the cardinality of the valid activity pattern $S \subset V$ and let $v_{(x_{\min}, y_{\min}, z_{\min})}$ be the smallest element of S. We define the set $S_0 \coloneqq \{v_{(x,y,z-z_{\min})} : v_{(x,y,z)} \in S\}$. We can numbered the elements of S_0 from v_1 to v_n so that for all $k \in [1, n]$, the list L_k defined as $L_k \coloneqq [v_1, \ldots, v_k]$ satisfies the propositions:

- $P_1(k)$: L_k is a node of $\mathcal{T}$.
- $P_2(k)$: For all $v \in S_0 \setminus \{v_1, \ldots, v_k\}$, we have that $v_{i(L_k)} < v$, where $i(L_k) \coloneqq \max\left\{\{1\} \cup \{1 \leq i \leq k : \bigcup_{j=1}^{i-1}\{v_j\} \text{ is valid}\}\right\}$.

Let v_1 be the smallest element of S_0, that is to say $v_1 = v_{(x_{\min}, y_{\min}, 0)}$. The elements $v_2, \ldots, v_n$ are chosen one by one with the following procedure.

1. If $\{v_1, \ldots, v_{k-1}\}$ is a valid activity pattern of $\mathcal{G}$, we define v_k to be the minimum of the set $S_0 \setminus \{v_1, \ldots, v_{k-1}\}$. As $P_2(k-1)$ is satisfied, $v_{i(L_{k-1})} < v_k$ and thus L_k is a node of $\mathcal{T}$. Moreover, as $\{v_1, \ldots, v_{k-1}\}$ is valid, we have that $i(L_k) = k$ and the proposition $P_2(k)$ is true.

2. If $N_{a_2}(v_{k-1}) \bigcap \{v_1, \ldots, v_{k-1}\} = \emptyset$, as S_0 is valid, we have that the set $N_{a_2}(v_{k-1}) \bigcap (S_0 \setminus \{v_1, \ldots, v_{k-1}\}\})$ is not empty. Let v_k be an element of $N_{a_2}(v_{k-1}) \bigcap (S_0 \setminus \{v_1, \ldots, v_{k-1}\}\}$. By the proposition $P_2(k-1)$, we know that $v_{i(L_{k-1})} < v_k$ and therefore L_k is a node of $\mathcal{T}$. As $\{v_1, \ldots, v_{k-1}\}$ is not valid, we have that $i(L_{k-1}) = i(L_k)$. As $P_2(k-1)$ holds, the proposition $P_2(k)$ is also true.
3. Similarly if $N_{\rho(a_2)}(v_{k-1}) \bigcap \{v_1, \ldots, v_{k-1}\} = \emptyset$, we choose for v_k an element of $N_{\rho(a_2)}(v_{k-1}) \bigcap (S_0 \setminus \{v_1, \ldots, v_{k-1}\}\}$.
4. Similarly if $N_{a_2}\left(v_{i(L_{k-1})}\right) \bigcap \{v_1, \ldots, v_{k-1}\} = \emptyset$, we choose for v_k an element of $N_{a_2}\left(v_{i(L_{k-1})}\right) \bigcap (S_0 \setminus \{v_1, \ldots, v_{k-1}\}\}$.
5. Similarly if $N_{\rho(a_2)}\left(v_{i(L_{k-1})}\right) \bigcap \{v_1, \ldots, v_{k-1}\} = \emptyset$, we choose for v_k an element of $N_{\rho(a_2)}\left(v_{i(L_{k-1})}\right) \bigcap (S_0 \setminus \{v_1, \ldots, v_{k-1}\}\}$.

D.2 Control the weight during the tree traversal

Among all the valid activity patterns of the graph $\mathcal{G}$, we only need those that may lead to a trail core $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2) \rangle$ of weight below some limit weight W. Given a node L of $\mathcal{T}$, we use Algorithm 3 to compute a lower bound on the weights of trail cores of the form $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2) \rangle$ with $\overline{a_2} = \overline{a_2}^{(L)}$ denoted by $\widetilde{\mathrm{w}}_{\min}^{\text{3-round}}(L)$.

Algorithm 3: $\widetilde{\mathrm{w}}_{\min}^{\text{3-round}}$

input : A node $L = [v_1, \ldots, v_n]$ of the tree $\mathcal{T}$
output: $\widetilde{\mathrm{w}}_{\min}^{\text{3-round}}(L)$ such that
$\widetilde{\mathrm{w}}_{\min}^{\text{3-round}}(L) \leq \min\{\mathrm{w}\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2) \rangle : \overline{a_2} = \overline{a_2}^{(L)}\}$

```
Δ ← {0}^729
min(w_box(Δ)) ← 0
for i = 1 to n do
    // see if min(w_box(Δ)) can be incremented
    ((x, y, z), (ρ̃(x, y, z))) ← v_i
    x_B ← ⌊x/3⌋
    newActiveBox ← true
    for j = 0 to 2 do
        if the box of Δ containing the trit of coordinate ρ̃^{-1}(3x_B + j, y, z)
          is already active then
            newActiveBox ← false
    if newActiveBox = true then
        min(w_box(Δ)) ← 1 + min(w_box(Δ))
        for j = 0 to 2 do
            Δ[ρ̃^{-1}(3x_B + j, y, z)] ← 1
w̃^{3-round}_min(L) ← 2 (min(w_box(Δ)) + w_box(a̅_2^{(L)}) + w_box(ρ(a_2)̅^{(L)}))
return w̃^{3-round}_min(L)
```

The weight of a 3-round trail core $\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2)\rangle$ is

$$\begin{aligned} \mathrm{w}\langle \rho^{-1}(b_1), b_1, a_2, \rho(a_2)\rangle = & \quad 2\,\mathrm{w}_{\mathrm{box}}\left(\rho^{-1}(b_1)\right) + \mathrm{w}_{\chi}(b_1, a_2) + 2\,\mathrm{w}_{\mathrm{box}}\left(\rho(a_2)\right) \\ \geq & \quad 2\left(\mathrm{w}_{\mathrm{box}}\left(\rho^{-1}(b_1)\right) + \mathrm{w}_{\mathrm{box}}(a_2) + \mathrm{w}_{\mathrm{box}}\left(\rho(a_2)\right)\right). \end{aligned}$$

The variable $\min\left(\mathrm{w}_{\mathrm{box}}(\Delta)\right)$ of Algorithm 3 stores a lower bound on the weight $\mathrm{w}_{\mathrm{box}}\left(\rho^{-1}(b_1)\right)$. The variables Δ and $\min\left(\mathrm{w}_{\mathrm{box}}(\Delta)\right)$ are only modified when adding 3 active trits to the variable Δ also activates 3 new active boxes of Δ. One of these 3 boxes is at the same position as an active box of $\rho^{-1}(b_1)$.

If a node L' of $\mathcal{T}$ is a descendant of another node L, the inequality $\widetilde{\mathrm{w}}^{\text{3-round}}_{\min}(L) \leq \widetilde{\mathrm{w}}^{\text{3-round}}_{\min}(L')$ holds. Consequently, we can prune the tree as soon as a node L such that $\widetilde{\mathrm{w}}^{\text{3-round}}_{\min}(L) > W$ is reached.

E Extension of trail cores

In this section we present in detail our algorithms for trail core extensions.

Finding all backward extensions of a trail core $C = \langle a_1, b_1, \ldots, a_{k-1}, b_{k-1}\rangle$ of weight below some bound W' means finding all the pairs $(a, b) \in \left(\mathbb{F}_3^{729}\right)^2$ that define a trail core $\langle a, b, a_1, \ldots, b_{k-1}\rangle$ of weight below W'. This is equivalent to finding the set $\mathrm{Ext}^{\mathrm{back}}(\Delta, W) := \{(a, b) : b = \lambda(a),\ \mathrm{DP}_{\chi}(b, \Delta) > 0,\ \widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \mathrm{w}_{\chi}(b, \Delta) \leq W\}$ with $\Delta = a_1$ and $W = W' - \mathrm{w}\langle C\rangle + \widetilde{\mathrm{w}}^{\mathrm{rev}}(a_1)$. Similarly, extending forward a trail core under some weight constraint consists in generating a set of the form $\mathrm{Ext}^{\mathrm{for}}(\Delta, W) := \{(a, b) : b = \lambda(a),\ \mathrm{DP}_{\chi}(\Delta, a) > 0,\ \mathrm{w}_{\chi}(\Delta, a) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b) \leq W\}$. We say that an element of $\mathrm{Ext}^{\mathrm{back}}(\Delta, W)$ (respectively $\mathrm{Ext}^{\mathrm{for}}(\Delta, W)$) is a backward (respectively forward) extension of weight $\widetilde{\mathrm{w}}^{\mathrm{rev}}(a) + \mathrm{w}_{\chi}(b, \Delta)$ (respectively $\mathrm{w}_{\chi}(\Delta, a) + \widetilde{\mathrm{w}}^{\mathrm{dir}}(b)$).

To explain the algorithms, we introduce the subspaces $\mathsf{Box\text{-}col}(x_B, z) = \langle e_{(3x_B+i,y,z)}\rangle_{(i,y)\in[0,2]\times[0,2]}$ and $\mathsf{Slice}(z) = \langle e_{(x',y',z)}\rangle_{(x',y')\in[0,8]\times[0,2]}$. Given a state A, we define the states $\mathsf{Slice}(A, z) := \sum_{(x',y')\in[0,8]\times[0,2]} e_{(x',y',z)} A[x', y', z]$ and $\mathsf{Box\text{-}col}(A, x_B, z) := \sum_{(i,y')\in[0,2]\times[0,2]} e_{(3x_B+i,y',z)} A[3x_B + i, y', z]$.

The algorithms used to extend a trail core outside the kernel in the forward and backward direction or inside the kernel in the backward direction follow the same principle to control the extensions' weight. Since the functions ρ and θ of the linear layer $\lambda = \theta \circ \rho$ are permutations, we can fix the trits of a pair (a, b) where $b = \lambda(a)$, by choosing either the value of a, $\rho(a)$ or b. The idea is to look for the pairs (a, b) that belong to some set $\mathrm{Ext}^{\mathrm{for}}(\Delta, \mathrm{W})$ or $\mathrm{Ext}^{\mathrm{back}}(\Delta, \mathrm{W})$ by choosing the value of a, $\rho(a)$ or b step by step (e.g. slice by slice or box-column by box-column). These algorithms use a particular direct sum of $\mathbb{F}_3^{729}$ – namely $\bigoplus_z \mathsf{Slice}(z)$ or $\bigoplus_{x_B,z} \mathsf{Box\text{-}col}(x_B, z)$ – denoted here with the general notation $\bigoplus_{i=1}^{k} S_i$. A variable ℓ, that represents either the value of a, $\rho(a)$ or b, is said to be valid if it corresponds to an extension that we are looking for. The algorithms are based on the fact that there exist some interesting sets $L_1 \subset S_1, \ldots, L_k \subset S_k$ such that every valid ℓ can be decomposed as $\ell = \sum_{i=1}^{k} \ell_i$, where $\ell_i \in L_i$ for all $i \in [1, k]$. The set $\{\sum_{i=1}^{k} \ell_i : (\ell_1, \ldots, \ell_k) \in L_1 \times \ldots \times L_k\}$ is seen as the set of the

leaves of a tree, where the first level corresponds to the choice of $\ell_1 \in L_1$, the second level deals with the choice of $\ell_2 \in L_2$ etc. It is generally not necessary to reach the last level of the tree to realize that the path leads to leaves that have a weight too large. Extensions are generated with a tree traversal, during which we compute a lower bound on the weights of the leaves that can be reached from the current vertex. The tree is pruned as soon as this lower bound, called the cost of the vertex, is too high.

The algorithm that deals with forward extensions inside the kernel takes advantage of the very strong requirement that an extension $(a, b) \in \mathrm{Ext}^{\mathrm{for}}(\Delta, W)$ must satisfy to have at the same time the difference a with the same box-activity pattern as Δ and the difference b inside the kernel. The χ-compatibility and the weight constraint are only checked subsequently.

E.1 Backward extensions inside the kernel

We denote by ℓ the variable used to gradually generate the difference b of a pair $(a, b) \in \mathrm{Ext}^{\mathrm{back}}(\Delta, W) \bigcap \mathbb{F}_3^{729} \times K$. The idea is to choose the value of ℓ box-column by box-column.

For all coordinates $(x_B, z) \in [0, 2] \times [0, 26]$, we pre-compute the set $X_{x_B,z}$ formed by all the box-column vectors $\ell_{x_B,z} \in K \cap \mathsf{Box\text{-}col}(x_B, z)$ such that $(\ell_{x_B,z}, \mathsf{Box\text{-}col}(\Delta, x_B, z))$ is a valid differential over χ. In particular, if a box-column of Δ of coordinates (x_B, z) is passive, $L_{x_B,z} = \{0\}$. If for one coordinate (x_B, z) the set $L_{x_B,z}$ is empty, then it do not exist a backward extension in the kernel. That happens for instance if the difference Δ has a box-column with exactly one active box. If none of the sets $L_{x_B,z}$ is empty, the sets $L_{x_B,z}$ define a tree as explained above. A leaf ℓ of this tree verifies $\ell \in K$ and $\mathrm{DP}_\chi(\ell, \Delta) > 0$. Only the leaves of weight $\widetilde{\mathrm{w}}^{\mathrm{rev}}(\rho^{-1}(\ell)) + \mathrm{w}_\chi(\ell, \Delta)$ below W must be reached.

Let $\ell = \sum_{x_B,z} \ell_{x_B,z}$, where $(\ell_{0,0}, \ldots, \ell_{2,26}) \in L_{0,0} \times \ldots \times L_{2,26}$, be a leaf of the tree. During the tree traversal, for each coordinate (x_B, z), we know the exact contribution of $\ell_{x_B,z}$ to the weight $\mathrm{w}_\chi(\ell, \Delta)$, that is $\mathrm{w}_\chi(\ell_{x_B,z}, \mathsf{Box\text{-}col}(\Delta, x_B, z))$. However, without the knowledge of the whole vector $(\ell_{0,0}, \ldots, \ell_{2,26})$, we cannot know the exact contribution of $\rho^{-1}(\ell_{x_B,z})$ to the weight $\widetilde{\mathrm{w}}^{\mathrm{rev}}(\rho^{-1}(\ell)) = 2\,\mathrm{w}_{\mathrm{box}}(\rho^{-1}(\ell))$. Indeed, two differences $\rho^{-1}(\ell_{x_B,z})$ and $\rho^{-1}(\ell_{x'_B,z'})$ can have an active trit in the same box. Therefore, it is not possible to assign to $\rho^{-1}(\ell_{x_B,z})$ a precise number of box activations. We consider roughly that a vector $\rho^{-1}(\ell_{x_B,z})$ activates as many boxes as its number of active trits. Given an element $\ell_{x_B,z}$ of $L_{x_B,z}$, we define its contribution to the total weight $\widetilde{\mathrm{w}}^{\mathrm{rev}}(\rho^{-1}(\ell)) + \mathrm{w}_\chi(\ell, \Delta)$ to be $c(\ell_{x_B,z}) := 2wt(\ell_{x_B,z}) + \mathrm{w}_\chi(\ell_{x_B,z}, \mathsf{Box\text{-}col}(\Delta, x_B, z))$, where wt stands for the Hamming weight. This overestimation needs to be corrected in order to have a lower bound on the weight of the extension $(\rho^{-1}(\ell), \ell)$. Let $\delta = \rho^{-1}(\mathsf{box}(\Delta))$. As the set of active coordinates of δ contains all the active coordinates of $\rho^{-1}(\ell)$

$$wt(\rho^{-1}(\ell)) - \mathrm{w}_{\mathrm{box}}(\rho^{-1}(\ell)) \leq wt(\delta) - \mathrm{w}_{\mathrm{box}}(\delta).$$

This is equivalent to

$$\sum_{x_B,z} c(\ell_{x_B,z}) - 2\,(wt(\delta) - \mathrm{w}_{\mathrm{box}}(\delta)) \leq 2\,\mathrm{w}_{\mathrm{box}}(\rho^{-1}(\ell)) + \mathrm{w}_\chi(\ell, \Delta).$$

For a level $k \in [1, 81]$ of the tree, a node ℓ_k of this level has the form $\sum_{i=1}^{k} \ell_{x_{B_i},z_i}$ where $\ell_{x_{B_i},z_i} \in L_{x_{B_i},z_i}$ for all $i \in [1, k]$. The cost of ℓ_k, that is a lower bound on the weights of the leaves reachable from ℓ_k, is given by

$$\sum_{i=1}^{k} c(\ell_{B_i,z_i}) + \sum_{i=k+1}^{81} \min\{c(d) : d \in L_{x_{B_i},z_i}\} - 2\,(wt(\delta) - \mathrm{w}_{\mathrm{box}}(\delta))\,.$$

E.2 Extensions outside the kernel

When looking for extensions outside the kernel, one has to deal with the column parity mixer θ as it does no longer act as the identity map. The main remark is that the map θ can be computed slice by slice. Depending on the direction of the extension, the extension's trits are either chosen at the input of θ or at the output.

Extensions in the backward direction The trits of an extension (a, b) of $\mathrm{Ext}^{\mathrm{back}}(\Delta, W)$ are chosen at the output of the map θ. In this subsection, we also denote by ℓ the variable whose trits are fixed slice by slice to obtain the final pair $(\rho^{-1} \circ \theta^{-1}(\ell), \ell)$ of weight $\widetilde{\mathrm{w}}^{\mathrm{rev}}\left(\rho^{-1} \circ \theta^{-1}(\ell)\right) + \mathrm{w}_\chi(\ell, \Delta)$. As the map θ^{-1} satisfies

$$\theta^{-1}(\ell)[x, y, z] = \ell[x, y, z] - P(\ell)[x - 1, z] - P(\ell)[x + 1, z + 1]$$

the vector $\mathsf{Slice}(\theta^{-1}(\ell), z)$ is determined by the values of the vectors $\mathsf{Slice}(\ell, z)$ and $\mathsf{Slice}(\ell, z + 1)$. We therefore choose the slices of ℓ in descending order of z-coordinate to know simultaneously the values of the slices of $\theta^{-1}(\ell)$. Let $z_0 \in [0, 26]$ be the coordinate of the first slice of ℓ that we choose and define for $i \in [1, 26]$, $z_i = z_0 - i$. For $i \in [0, 26]$, we introduce the set L_i formed by all the vectors $\ell_i \in \mathsf{Slice}(z_i)$ that are compatible through χ with the output difference $\mathsf{Slice}(\Delta, z_i)$. The set L_i is associated to the $(i+1)$-th level of a tree. The cost of a vertex of this tree is computed using Algorithm 4 which introduces an auxiliary variable to compute the slices of $\theta^{-1}(\ell)$. When a leaf ℓ is reached during the tree traversal, only the weight of the extension $(\rho^{-1} \circ \theta^{-1}(\ell), \ell)$ remains to be checked.

Extensions in the forward direction For an extension in the forward direction, trits are chosen at the input of the map θ, using again a variable that is chosen slice by slice and denoted by ℓ. Pairs of the form $\left(\rho^{-1}(\ell), \theta(\ell)\right)$, that might constitute a forward extension, are generated by fixing the slices of the variable ℓ in descending order of z-coordinate for the same reason as in the case of a backward extension. The Property 2 of the nonlinear layer provides necessary conditions that the difference $\rho^{-1}(\ell)$ must fulfill to be compatible through

Algorithm 4: Cost of a node of the tree used for backward extensions

input : A vertex $v = \sum_{i=0}^{k} \ell_i$ of the $k+1$-th level of the tree, $k \in [0, 25]$
The auxiliary variable aux_p associated to the parent $p = \sum_{i=0}^{k-1} \ell_i$ of v
output: The cost c of v and the auxiliary variable aux_v associated to v

if $k = 0$ **then**
  $\mathsf{aux}_v \leftarrow 0$
  $\mathsf{c} \leftarrow \mathrm{w}_\chi(\ell_0, \mathsf{Slice}(\Delta, z_0)) + 2\sum_{i=1}^{26} \mathrm{w}_{\mathrm{box}}(\mathsf{Slice}(\Delta, z_i))$
if $1 \leq k \leq 25$ **then**
  $\mathsf{aux}_v \leftarrow \mathsf{aux}_p$
  forall $(x, y) \in [0, 8] \times [0, 2]$ **do**
    $\mathsf{aux}_v[x, y, z_k] \leftarrow v[x, y, z_k] - P(v)[x-1, z_k] - P(v)[x+1, z_k+1]$
  $\mathsf{c} \leftarrow \widetilde{\mathrm{w}}^{\mathrm{rev}}(\rho^{-1}(\mathsf{aux}_v)) + \sum_{i=0}^{k} \mathrm{w}_\chi(\ell_i, \mathsf{Slice}(\Delta, z_i)) + 2\sum_{i=k+1}^{26} \mathrm{w}_{\mathrm{box}}(\mathsf{Slice}(\Delta, z_i))$
return $(\mathsf{c}, \mathsf{aux}_v)$

χ with the input difference Δ. More precisely, it indicates the activity or passivity of some trits of $\rho^{-1}(\ell)$ and specifies the value of some other trits. As the map ρ only permutes trits, conditions on the slices of ℓ directly follow. We write $z_0 \in [0, 26]$ for the coordinate of the first slice of ℓ that is chosen and $z_i = z_0 - i$ for $i \in [1, 26]$. For $i \in [0, 26]$, the $i+1$-th level of the tree is given by the set that contains all the vectors of $\mathsf{Slice}(z_i)$ satisfying the constraints stated above. During the tree traversal, it is necessary to ensure that the path will lead to a leaf ℓ such that $\rho^{-1}(\ell)$ has the same box-activity pattern as the difference Δ and of weight $\mathrm{w}_\chi(\Delta, \rho^{-1}(\ell)) + \widetilde{\mathrm{w}}^{\mathrm{rev}}(\theta(\ell))$ not too high. An algorithm similar to Algorithm 4 is used to compute a lower bound on the weights of the χ-compatible leaves that descend from a particular vertex. A leaf ℓ of this tree does not necessarily satisfy $\mathrm{DP}_\chi(\Delta, \rho^{-1}(\ell)) > 0$. This condition needs to be checked each time a leaf is reached.

E.3 Forward in-kernel extension

We present the method used to generate the set $A_\Delta \subset \mathbb{F}_3^{729}$ formed by all differences a satisfying $\mathrm{DP}_\chi(\Delta, a) > 0$ and $\rho(a) \in K$.

Filtering phase A filtering phase is first applied to identify cases of impossible extensions. During that step, some coordinates of the differences $a \in A_\Delta$ that are necessarily active and some that are necessarily passive are identified.

For all $a \in A_\Delta$, the box activity pattern of a is equal to that of Δ. Moreover, since the difference $\rho(a)$ belongs to the kernel, it cannot have a column with exactly one active trit. This is why, using Algorithm 5, it may be possible to conclude from the box activity pattern of the difference Δ that the set A_Δ is

Algorithm 5: First filter

input : The difference Δ

output: false if it can be deduced that A_Δ is empty; Otherwise, a difference $\ddot{a}$ such that the implication $\ddot{a}[x,y,z] = 0 \implies a[x,y,z] = 0$ holds for all $a \in A_\Delta$ and $(x,y,z) \in [0,\,8] \times [0,\,2] \times [0,\,26]$

```
ä ← box(Δ)
ρ(ä) ← ρ(box(Δ))
forall (x, z) ∈ [0, 8] × [0, 26] do
    if the column of ρ(ä) has exactly one active coordinate (x, y, z) then
        ä[x, y, z] ← 0
        ρ(ä)[ρ̃^{-1}(x, y, z)] ← 0
    if the box of ä containing the trit of coordinates ρ̃^{-1}(x, y, z) is not longer
      active then
        return false
return ä
```

empty. If the difference Δ passes this filter, the algorithm outputs a variable $\ddot{a}$ that indicates trits of the differences $a \in A_\Delta$ that are necessarily passive.

A second filter is applied if the difference Δ passes the first one. If the difference Δ has boxes whose values belong to the set $\{(0,0,1),(0,0,2),(1,0,0),(2,0,0)\}$, it is possible to locate, using Property 2 (see Section 3), some trits of the differences $a \in A_\Delta$ that are necessarily active. If there are after this step trits that are simultaneously necessarily active and passive, the set A_Δ is empty.

Finally, from the difference $\ddot{a}$ returned by Algorithm 5, it may be possible to deduce other trits of a and $\rho(a)$ that must be active. If a box of $\ddot{a}$ has a unique active coordinate, we conclude that this is an active coordinate of a. Indeed, the box of a containing that coordinate must be active, and this trit is the only one of the box that can be active. Moreover, if a column of $\rho(\ddot{a})$ has exactly 2 active trits and one of them is necessarily active, then the second is also necessarily active.

Extension phase If the difference Δ passes the filtering phase, an extension phase is performed. It consists in scanning all pairs of the form $(a, \rho(a))$ with $\rho(a) \in K$ that also respect the constraints deduced during the filtering phase. We test for each pair if the output difference a is compatible with the input difference Δ through χ.