Troika:
a ternary hash function

Reference document

Version 1.0.1

December 21, 2018

cybercrypt
cyber-crypt.com

Table of Contents

1 Introduction

1.1 Context

IOTA has developed a novel public distributed ledger technology based on a directed acyclic graph (DAG), called the Tangle. The security of the core components in this system relies on the security of a cryptographic hash function and it is therefore crucial that this hash function fulfills the security requirements to ensure the validity of the transactions on the Tangle.

Troika is a new hash function for IOTA's ternary architecture and platform developed by CYBERCRYPT.

1.2 Design goals for Troika

The main design objective is to construct a crytographically secure hash function $h : \mathbb{F}_3^* \to \mathbb{F}_3^{243}$ mapping arbitrary-length inputs to hash values of 243 trits. It should follow the sponge construction with a rate of 243 and a capacity of 486 trits, yielding a total state of 729 trits. Furthermore, the rate part of the state of Troika is overwritten by the input instead of added to it, in order to enable distributed hashing where only the capacity part of the state (486 trits) need to be sent instead of the entire state (729 trits).

The three main security requirements for a secure hash function are:

Preimage resistance: For a given output y it should be computationally infeasible to find an input x' such that $y = h(x')$.

Second preimage resistance: For a given x and $y = h(x)$ it should be computationally infeasible to find $x' \neq x$ such that $h(x') = y$.

Collision resistance: It should be computationally infeasible to find two distinct inputs x, x' such that $h(x) = h(x')$.

We note that not all applications require all these properties, for instance collision resistance is only a minor issue when using a cryptographic hash function for secure storage of passwords. Likewise, some hash-based signature schemes do not require collision resistance [16].

For the design of Troika, we have the following explicit security requirements:

Preimage resistance: No preimage attack of non-negligible success probability with a complexity of less than 3^{243} queries.

Second preimage resistance: No second preimage attack of non-negligible success probability with a complexity of less than 3^{243} queries.

Collision resistance: No collision attack of non-negligible success probability with a complexity of less than $3^{243/2}$ queries.

2 Specification

This section provides the complete specification of the Troika hash function. It is self-contained in the sense that implementers of the hash function can solely focus on this section.

2.1 Preliminaries

We denote the finite field of 3 elements by $\mathbb{F}_3 = \{0, 1, 2\} = \{0, 1, -1\}$. An element of $\mathbb{F}_3$ is also called a *trit*. The two basic field operations, addition and multiplication in $\mathbb{F}_3$ are defined in Table 2.1. The analysis of Troika will also require a difference operation in $\mathbb{F}_3$ which is defined via subtraction, cf. Table 2.1.

For any prime p and positive integer n, the finite field of $q = p^n$ elements is denoted by $\mathbb{F}_{p^n}$, which is an extension field of degree n over $\mathbb{F}_p$. Seen as an additive group, $\mathbb{F}_{p^n}$ is isomorphic to the vector space $\mathbb{F}_p^n$. An element of $\mathbb{F}_3^3$ will be referred to as a *tryte*. Addition and subtraction in $\mathbb{F}_{3^n}$ are defined as the component wise addition and subtraction in $\mathbb{F}_3$ (respectively). Our design and its analysis do not require extension field multiplication.

We denote the scalar product of two vectors v and w in the vector space $\mathbb{F}_3^n$ by

$$v^T w \stackrel{\text{def}}{=} \sum_{i=1}^{n} v_i w_i, \tag{2.1}$$

where addition and multiplication are taken in $\mathbb{F}_3$.

2.2 Overall design

2.2.1 State layout

The hash function Troika operates on a state $x \in \mathbb{F}_3^{729}$ which is organized as a $9 \times 3 \times 27$ cuboid of trits:

$$x \in \mathbb{F}_3^{9\times 3\times 27}. \tag{2.2}$$

The individual trits of the state are identified as $x_{x,y,z}$ via their x, y, z coordinates where $0 \le x < 9, 0 \le y < 3, 0 \le z < 27$. A row of x is a (row) vector $x_{\cdot,y,z}$

a	b	$a+b$
0	0	0
0	1	1
0	2	2
1	0	1
1	1	2
1	2	0
2	0	2
2	1	0
2	2	1

a	b	$a-b$
0	0	0
0	1	2
0	2	1
1	0	1
1	1	0
1	2	2
2	0	2
2	1	1
2	2	0

a	b	$a \cdot b$
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	2
2	0	0
2	1	2
2	2	1

Table 2.1: Addition, subtraction and multiplication operations in $\mathbb{F}_3$.

of 9 trits, a column a (column) vector $x_{x,\cdot,z}$ of 3 trits, and a lane a row vector $x_{x,y,\cdot}$ of 27 trits. The collection $x_{\cdot,\cdot,z}$ of all 27 trits with a common z-coordindate is called a slice. Analogously, the 243 trits $x_{\cdot,y,\cdot}$ with the same y-coordinate are called a plane, and the 81 trits $x_{x,\cdot,\cdot}$ with the same x-coordinate are called a sheet. This nomenclature follows the terminology introduced by Keccak [5] and is illustrated in Figure 2.1.

Equivalently, such a state x can be seen as a 3×27 cube of columns of 3 trytes, referred to as x:

$$\mathsf{x} \in \left(\left(\mathbb{F}_3^3\right)^3\right)^{3\times 27} \cong \mathbb{F}_3^{9\times 3\times 27}. \tag{2.3}$$

In this representation, each 3 adjacent trits in a row are grouped into one tryte. As for the trit-oriented view, a row, column, or lane of x is a collection $\mathsf{x}_{\cdot,y,z}$, $\mathsf{x}_{\cdot,y,z}$ or $\mathsf{x}_{\cdot,y,z}$ of 3, 3, and 27 trytes, respectively. Slices, planes, and sheets are defined analogously.

The correspondence between a sequence $a_0, \ldots, a_{728}$ of trits and a state x is given by

$$\begin{aligned} x_{x,y,z} &= a_{\alpha(x,y,z)}, \text{ with} \\ \alpha(x,y,z) &= 27z + 9y + x \end{aligned} \tag{2.4}$$

so the state x corresponds to the sequence a slice by slice, row by row.

In particular, a 243-trit sequence $a_0 \ldots, a_{242}$ maps onto the positions $x_{x,y,z}$ by means of equation (2.4), that is, onto the first nine slices of x. This division of rate and capacity part of the state is also illustrated in Figure 2.2.

For the analysis of Troika, we will regularly need to illustrate sparse states with only relatively few nonzero trits. Such states are written as a sequence of

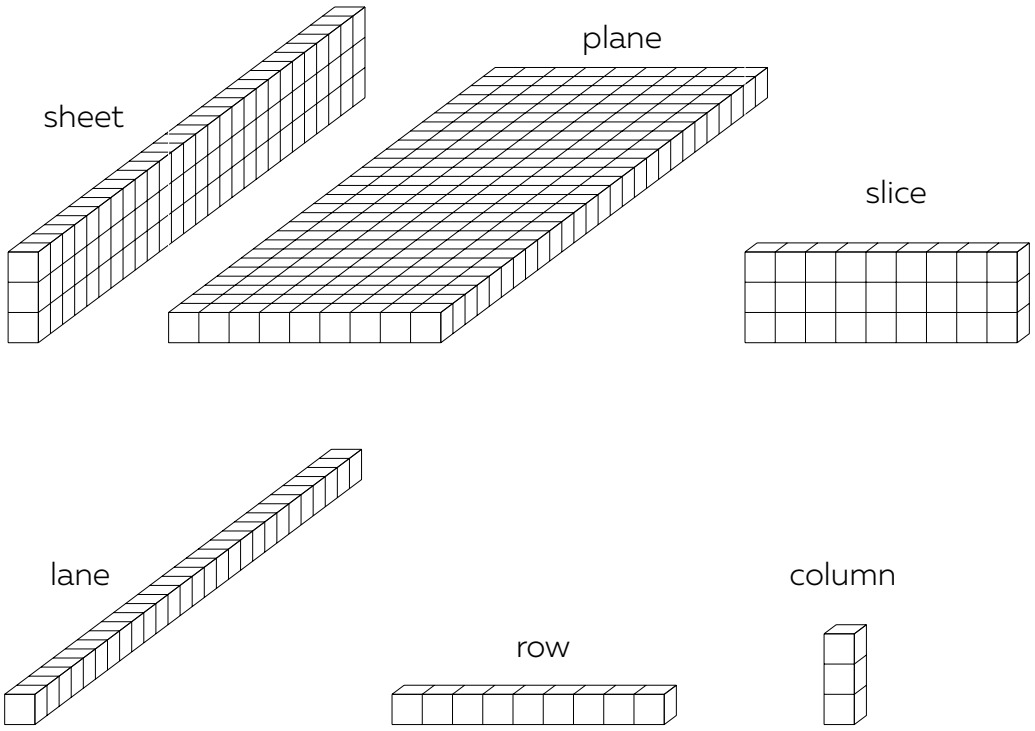

Figure 2.1: Illustration of the nomenclature for different parts of the state of Troika.

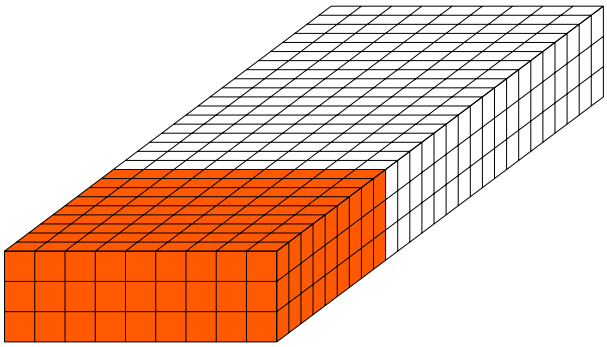

Figure 2.2: Illustration of the rate (orange) and capacity (white) parts of the state of Troika.

slices (along with their z-coordinate) which contain nonzero entries. Furthermore, for each column, its parity (column sum in $\mathbb{F}_3$) will be indicated below a dashed line. As an illustration, the state x with all trits zero except $x_{1,2,3} = 2$ and $x_{2,0,3} = x_{2,1,3} = x_{2,2,3} = 1$ is written as follows:

$$
\begin{array}{c}
z = 3: \\
\begin{array}{ccccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
\end{array}
\tag{2.5}
$$

2.2.2 Sponge construction

The design of Troika follows the sponge construction, cf. Fig. 2.3. Troika uses a state of 729 trits, which is divided into two parts: *capacity* of 486 and *rate* of 243 trits. The state is initially initialized with all zeros. A message $m \in \mathbb{F}_3^*$ is padded and formatted into n blocks $m_0, \ldots, m_{n-1} \in \mathbb{F}_3^{243}$ of $r = 243$ trits each. Each message block $m_0, \ldots, m_{n-1}$ is then assigned to the rate part of the state followed by a chained call to the f function, which is a permutation on $\mathbb{F}_3^{729}$. Specifically, the 243-trit rate part of the state corresponds to the positions $x_{x,y,z}$ by means of equation (2.4), that is, to the first nine slices of x.

The overall operation of Troika is given in Algorithm 1.

Algorithm 1 Troika(m)

1: $(m_0, m_1, \ldots, m_{n-1}) \leftarrow m||10^*$
2: $x \leftarrow 0^{729}$
3: **for** $i = 0, \ldots, n-1$ **do**
4: $\quad x_{x,y,z} \leftarrow (m_i)_{\alpha(x,y,z)}$ for $0 \leq x < 9, 0 \leq y < 3, 0 \leq z < 9$.
5: $\quad x \leftarrow f(x)$
6: **end for**
7: $z_{\alpha(x,y,z)} \leftarrow x_{x,y,z}$ for $0 \leq x < 9, 0 \leq y < 3, 0 \leq z < 9$.
8: **return** z

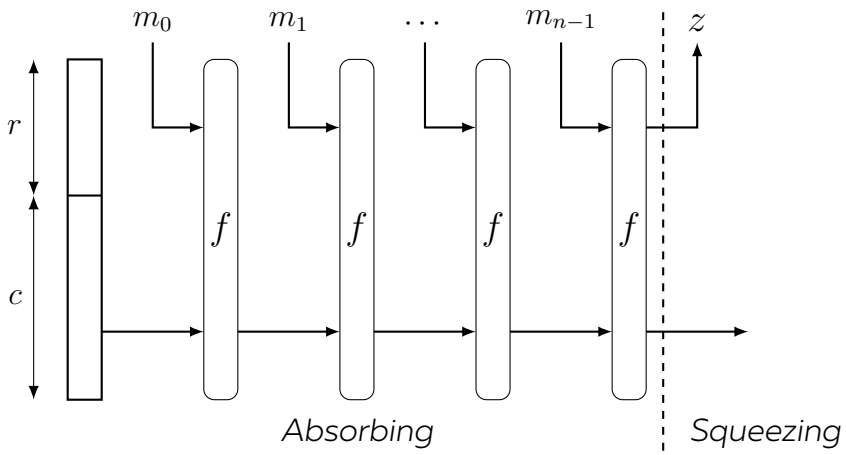

Figure 2.3: Overview of the sponge construction.

2.3 Round function & components

The permutation $f : \mathbb{F}_3^{729} \to \mathbb{F}_3^{729}$ is defined as the R-fold iteration of a round function f^r which is itself composed of several components SubTrytes, AddColumnParity, ShiftRows, ShiftLanes, and AddRoundConstant^r:

$$f : \mathbb{F}_3^{729} \to \mathbb{F}_3^{729} \tag{2.6}$$

$$x \mapsto \left(\bigcirc_{r=1}^{R} f^r\right)(x), \tag{2.7}$$

with

$$f^r = (\text{AddRoundConstant}^r \circ \text{AddColumnParity} \circ \text{ShiftLanes} \circ \text{ShiftRows} \circ \text{SubTrytes})$$

The number of rounds is defined as $R = 24$. An algorithmic description of the f permutation is given in Algorithm 2.

Algorithm 2 Troika's internal permutation $f(x)$

1: **for** $r = 1, \ldots, 24$ **do**
2: $\quad x \leftarrow \text{SubTrytes}(x)$
3: $\quad x \leftarrow \text{ShiftRows}(x)$
4: $\quad x \leftarrow \text{ShiftLanes}(x)$
5: $\quad x \leftarrow \text{AddColumnParity}(x)$
6: $\quad x \leftarrow \text{AddRoundConstant}^r(x)$
7: **end for**
8: **return** x

2.3.1 SubTrytes: S-boxes

The SubTrytes mapping consists of the application of a 3-trit S-box $s : \mathbb{F}_3^3 \to \mathbb{F}_3^3$ to each tryte of the state. Define

$$F(x_0, x_1, x_2) = (x_0, x_1, x_0 \cdot x_1 + x_2)\,, \tag{2.8}$$

and the trit permutations

$$\pi(x_0, x_1, x_2) = (x_1, x_2, x_0)\,, \tag{2.9}$$

$$\rho(x_0, x_1, x_2) = (x_2, x_1, x_0)\,, \tag{2.10}$$

then s is defined via

$$s : \mathbb{F}_3^3 \to \mathbb{F}_3^3$$
$$(x_0, x_1, x_2) \mapsto \rho(F(\pi(F(\pi(F(\pi(F(x_0 - 1, x_1, x_2))))))). \tag{2.11}$$

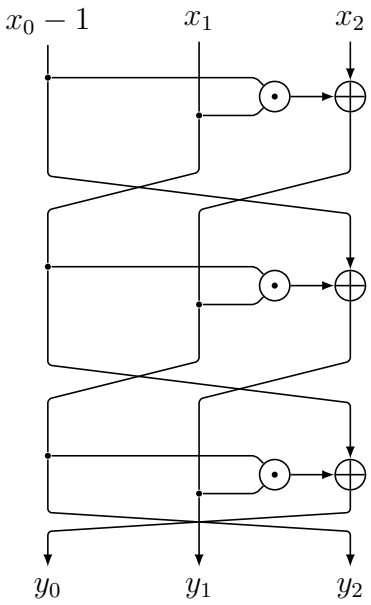

Figure 2.4: Outline of the Feistel network used for constructing the S-box.

This corresponds to a 3-round source-heavy unbalanced 3-trit Feistel network with the Feistel function $x \cdot y$, the cyclic shift π, and the additional affine mapping $(x_0, x_1, x_2) \mapsto (x_0 - 1, x_1, x_2)$ at the input and the trit permutation ρ at the output (see Figure 2.4). A lookup table for s on $\mathbb{F}_3^3$, identifying (a_0, a_1, a_2) with the integer $a = 9a_0 + 3a_1 + a_2$ where $0 \le a < 27$, is given in Table 2.2.

The mapping SubTrytes on the entire state x is then defined as

$$\mathsf{x}_{\mathsf{i,j,k}} \mapsto s(\mathsf{x}_{\mathsf{i,j,k}}) \tag{2.12}$$

for $0 \le i < 9, 0 \le j < 3, 0 \le k < 9$.

2.3.2 ShiftRows: Shifts inside rows

The purpose of ShiftRows is to provide diffusion along the x-axis in each row by shifting entire trytes cyclically to the right:

$$\begin{pmatrix} \mathsf{x}_{\cdot,0,i} \\ \mathsf{x}_{\cdot,1,i} \\ \mathsf{x}_{\cdot,2,i} \end{pmatrix} \mapsto \begin{pmatrix} \mathsf{x}_{\cdot,0,i} \\ \mathsf{x}_{\cdot,1,i} \ggg 1 \\ \mathsf{x}_{\cdot,2,i} \ggg 2 \end{pmatrix} \tag{2.13}$$

for all slices $0 \le i < 27$.

x	$S(x)$	x	$S(x)$	x	$S(x)$
0	6	9	0	18	3
1	25	10	1	19	13
2	17	11	2	20	23
3	5	12	9	21	7
4	15	13	22	22	11
5	10	14	26	23	12
6	4	15	18	24	8
7	20	16	16	25	21
8	24	17	14	26	19

Table 2.2: Lookup table for the tryte S-box.

j	$R_{0,j}, R_{1,j}, \ldots, R_{8,j}$
0	$19, 13, 21, 10, 24, 15, 2, 9, 3$
1	$14, 0, 6, 5, 1, 25, 22, 23, 20$
2	$7, 17, 26, 12, 8, 18, 16, 11, 4$

Table 2.3: Rotation constants for ShiftLanes.

2.3.3 ShiftLanes: Shifts inside lanes

The purpose of ShiftLanes is to provide diffusion along the z-axis in each lane by shifting trits cyclically to the right:

$$\begin{pmatrix} x_{i,0,\cdot} \\ x_{i,1,\cdot} \\ x_{i,2,\cdot} \end{pmatrix} \mapsto \begin{pmatrix} x_{i,0,\cdot} \ggg R_{i,0} \\ x_{i,1,\cdot} \ggg R_{i,1} \\ x_{i,2,\cdot} \ggg R_{i,2} \end{pmatrix} \tag{2.14}$$

for all sheets $0 \leq i < 9$, where $R_{i,j}$ are defined as in Table 2.3.

2.3.4 AddColumnParity: Column parity addition

The mapping θ provides diffusion along columns by adding to each column $x_{x,\cdot,z}$ the parities of the two adjacent columns $x_{x-1,\cdot,z}$ and $x_{x+1,\cdot,z+1}$, where indices are taken modulo their respective dimensions:

$$\begin{pmatrix} x_{i,0,j} \\ x_{i,1,j} \\ x_{i,2,j} \end{pmatrix} \mapsto \begin{pmatrix} x_{i,0,j} \\ x_{i,1,j} \\ x_{i,2,j} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} x_{(i-1),0,j} \\ x_{(i-1),1,j} \\ x_{(i-1),2,j} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} x_{(i+1),0,(j+1)} \\ x_{(i+1),1,(j+1)} \\ x_{(i+1),2,(j+1)} \end{pmatrix} \tag{2.15}$$

for all columns $0 \leq i < 9$ in all slices $0 \leq j < 27$.

2.3.5 AddRoundConstant: Round constants

The role of the component AddRoundConstant is to add round-dependent constants to one plane of the state. It is the only mapping in the round transformation that differs from round to round. The round constants themselves are denoted $\mathrm{RC}^r \in \mathbb{F}_3^{9\cdot 27}$ for $r = 1, \ldots, R$:

$$\mathrm{RC}^r = (L_{243\cdot(r-1)}, \ldots, L_{243\cdot(r-1)+242}), \tag{2.16}$$

and are derived by means of the following linear feedback shift register (LFSR) over $\mathbb{F}_3$:

$$L_{n+11} = L_{n+3} - L_n \qquad (n \geq 0), \tag{2.17}$$

initialized with $(L_0, \ldots, L_{10}) = (2, 2, 2, 2, 1, 2, 0, 1, 0, 1, 1)$. The AddRoundConstant mapping is then defined as

$$x_{\cdot,0,\cdot} \mapsto x_{\cdot,0,\cdot} + \mathrm{RC}^r, \tag{2.18}$$

where the mapping of the 243 trits of RC^r to the trits $x_{x,0,z}$ of the plane $x_{\cdot,0,\cdot}$ is defined as

$$x_{x,0,z} = \mathrm{RC}^r_{9z+x}, \tag{2.19}$$

for $0 \leq x < 9, 0 \leq z < 27$.

2.4 Test vectors

A number of input-output pairs for the Troika hash function are provided in Table 2.4.

Input	Length [trits]	Hash value
0	1	002000202102220201001220111 001110000100020222112211021 100211012021011011012010120 121020201011100221110202212 001222102020210020002012200 211220021202001200101020122 120001011201010211200210021 020000021001202001122002210 221110021110000012122220002
00	2	202002111110120011101221222 120022111012201202121212010 000010202021222010212121210 210201112221111010002100121 110001121212000222121202002 210002220220222210022101211 200111212102201120221102112 020011102000021012211022211 002112200212012211202212111
$10 \cdots 02$	243	120220121212020211012202221 121212221211022112222201212 001221011202211020020000200 100012021222011211111202210 102202211120102211222000002 210202121001221010022001101 021010000021221011220002100 012221020010112001222020112 100211020221121101101102212

Table 2.4: Test vectors for Troika.

3 Design rationale

This section provides the design rationale for the Troika hash function. For each of the design choices made in the individual components of the round transformation, as well as the overall design and the number of rounds, we document our reasoning behind these design decisions.

3.1 Overall design

The overall construction of Troika follows the sponge construction [3]. It is well-analyzed [2, 3] and in particular comes with a proof of indifferentiability from a random oracle [3] when used with a random permutation or random function as the underlying primitive. Per IOTA's requirements, the design of Troika employs a variation of the usual sponge construction in that it uses replacement instead of an addition operation for injecting message blocks into the rate part of the state.

With the intended security level of the design set at 243 trits against preimage and second preimage attacks and $243/2$ trits against collision attacks, a capacity of at least 486 trits is necessary. With an overall state size of 729 trits, this yields a rate of 243 trits.

Furthermore, the design of Troika is done directly over $\mathbb{F}_3$ instead of a ternary adaption of a binary design in order to improve its efficiency.

3.2 Design of the S-box

The SubTrytes mapping is the only non-linear transformation in the Troika permutation and as such is crucial to the security of the overall primitive. Since f is required to be a permutation, the S-box used in the SubTrytes mapping needs to be bijective.

3.2.1 Size and number of S-boxes

With a state size of $729 = 3^6$ trits, the size of the S-box in trits is necessarily a power of 3. Smaller S-boxes are typically easier to implement on

resource-constrained platforms and in hardware, while larger S-boxes offer more nonlinearity, potentially reducing the number of rounds required for a fixed security level. For binary designs, 8-bit S-boxes are usually considered a good general-purpose trade-off between implementation characteristics and cryptanalytic considerations, see for instance the design of the AES [11] or hash functions such as Grøstl [12]. In the context of lightweight cryptography, 4-bit S-boxes are common [8, 24]. Translated to $\mathbb{F}_3$, this suggests 5-trit or 3-trit S-boxes, respectively. With the intended deployment of Troika in a variety of low-resource environments, our design features a 3-trit bijective S-box.

We chose to employ a single S-box in the S-box layer instead of applying many different S-boxes in parallel. While the latter has the potential to slightly increase the security level of the design (mainly by avoiding self-similarity and similar properties), this comes at a significant increase in implementation cost.

3.2.2 Random S-box vs dedicated design

Random S-boxes have on average very good algebraic and cryptographic properties with respect to all major attacks (for a comprehensive study see for instance [23]). Choosing a random S-box can also be seen as an attempt to be as close as possible to the notion of a random permutation for the overall design, avoiding inbuilt structure as much as possible. Random S-boxes however significantly increase the size of hardware implementations and usually do not allow flexible trade-offs between circuit size and latency (as opposed to iterated constructions).

We therefore chose to use a dedicated design for maximum implementation efficiency.

3.2.3 Differential properties

In the context of a hash function design, the arguably single most important cryptographic property of an S-box is its differential uniformity, also called the Δ-parameter of an S-box (see Section 4.2 for a definition). As a rule, Δ should be as low as possible.

For Troika, we therefore focused on finding a highly efficient 3-trit bijective S-box with differential uniformity $\Delta = 3$. The S-box employed by the SubTrytes mapping achieves this. Together with a ternary linear potential of $3^{-2.74}$, it offers good resistance against differential and linear cryptanalysis. The rationale behind its construction is given in Section 3.2.4.

3.2.4 Feistel construction

Since the direct construction of bijective 3-uniform S-boxes that can be implemented in few operations over $\mathbb{F}_3$ seems hard, we opted to explore the iteration of a simple nonlinear but non-bijective function $F : \mathbb{F}_3^3 \to \mathbb{F}_3^3$ in an unbalanced Feistel structure with 3 branches. This has the main advantage that the resulting Feistel construction is then bijective by construction. The mapping π, implementing the Feistel swap between the rounds as given by Equation (2.9) was chosen as the simple $x_i \mapsto x_{(i-1) \bmod 3}$, which is a common choice for generalized Feistel networks, see for instance the ciphers HIGHT [15], LEA [14] and CLEFIA [25].

For this construction to have good cryptographic properties, it is necessary to have at least 3 Feistel rounds (since we have three branches). Among all considered candidates for the F-function, the final choice (see Equation (2.8)) has the lowest number of ternary operations (2 per round) and has differential uniformity 3 when iterated over 3 rounds.

The resulting mapping

$$\begin{aligned} s' : \mathbb{F}_3^3 &\to \mathbb{F}_3^3 \\ (x_0, x_1, x_2) &\mapsto F(\pi(F(\pi(F(x_0, x_1, x_2))))) \end{aligned} \tag{3.1}$$

has optimal differential properties, but 2 fixed points. For a random permutation, the expected number of fixed points is 1 [22]. The introduction of the affine mapping $(x_0, x_1, x_2) \mapsto (x_0 - 1, x_1, x_2)$ and the final trit swap ρ allows to have exactly one fixed point at the minimum additional cost of one gate per S-box. Since both these mappings are affine and linear (respectively), they do not change the differential properties of the resulting mapping, as only columns and rows of the difference distribution table are permuted.

Overall, this Feistel construction allows us to construct a 3-trit bijective S-box with differential uniformity 3 at the cost of just 7 basic ternary operations overall.

3.3 Design of the ShiftRows operation

The purpose of the ShiftRows operation is to provide diffusion along the x-axis. Any selection of three different shift amounts for the three rows would provide equally good diffusion for the rectangular structure of each slice of the state. Our choice of $0, 1$ and 2 as the shifts for rows 0, 1 and 2 (respectively) is arguably the simplest such choice, and furthermore in line with the ShiftRows operation of the AES and many related designs.

Following Definition 4 and the rectangular design of Section 7 of [10], such a choice of shifts can be considered optimal. We note however that since the number of columns is greater than the number of rows, Theorem 2 of [10] is not applicable here, hence even an diffusion-optimal ShiftRows operation cannot guarantee at least $\mathcal{B}^2$ active S-boxes every 4 rounds, where $\mathcal{B}$ is the (differential) branch number of the AddColumnParity mapping, see Section 4.4.2.

The rationale behind choosing shifts over entire trytes instead of trits is to ensure the activation of different S-boxes after the ShiftRows mapping even if S-boxes are only activated by single trits.

3.4 Design of the ShiftLanes operation

Analogous to ShiftRows, the purpose of the ShiftLanes operation is to provide diffusion along the z-axis. The ShiftLanes operation implements this with 27 different shift amounts that are derived from a randomly chosen permutation of $\{0, \ldots, 27\}$ in order to shift each lane of the state differently. Note that this implies that we shift in a different granularity than entire S-boxes. As an alternative, we evaluated the use of the same shift amount for the three trytes corresponding to one S-box, however the final design choice greatly improves the empirical diffusion, since it breaks up lower numbers of active trits into more slices, and consequently, active S-boxes.

3.5 Design of the AddColumnParity operation

The AddColumnParity mapping provides diffusion along columns by adding to each column the parities of two adjacent columns, where one is taken from the same, and the other from an adjacent slices of the state.

This type of diffusion mapping can be described in terms of the Column Parity Mixer framework [27]. A similar mapping is used as the θ mapping in SHA-3 [5]. As a consequence, this type of mappings is well-understood and well-analyzed. We chose the AddColumnParity operation to be a column parity mixer due to its good diffusion properties (it has branch number 4 both on trit and tryte level, see Section 4.4.2) while at the same time offering a very low implementation cost of only 2 additions per trit.

3.6 Design of the round constants

The round constants are supposed to break similarities between the rounds of the Troika permutation. Conventionally, the addition of one or very few trits to the state per round would be regarded sufficient [12] to avoid such properties that could for instance be exploited by slide attacks [7]. However, with the advent of invariant subspace and nonlinear invariant attacks [17, 28], very sparse round constants are seen as potentially problematic.

Ideally, a complete pseudorandom sequence of trits would be added to the entire state in each round. This however comes at a significant implementation cost of 729 additions per round, which is half of the cost of the entire diffusion layer. The design of the AddRoundConstant operation in Troika therefore balances the requirements of dense round constants and efficiency by adding a different sequence of 243 trits to the first plane in each round.

In an implementation of Troika, these constants can either be precomputed or computed on the fly. In order to provide an efficient implementation also for the latter case, we opted for a linear feedback shift register (LFSR) with the minimal implementation cost of only one ternary gate per trit.

The connection polynomial of the LFSR is given by $z^{11} - z^3 + 1$. It is primitive in $\mathbb{F}_3[z]$, therefore the period of this LFSR is maximal at $3^{11} - 1 = 177146$ [18]. In total, We need $243 \cdot R = 5832$ clockings worth of trits for all round constants for $R = 24$ rounds.

The round constants of Troika have been verified to resist invariant subspace, nonlinear invariant, and generalized nonlinear invariant attacks. See Section 4.9 for details.

3.7 Choice of the number of rounds

The choice of the number of rounds for Troika is based on the cryptanalysis results summarized in Section 4. No potentially exploitable cryptanalytic properties covering more than 5 rounds were identified. The diffusion layer only allows to extend such a property by at most 3 rounds in either direction. The main limiting factor are differential attacks, since our analysis can only prove a lower bound of 25 active S-boxes per 4 rounds. This bound implies that for $R = 20$ rounds, one can guarantee that no differential trail with probability higher than $3^{-2 \cdot 25 \cdot 5} = 3^{-250}$ exists.

We emphasize that no actual differential trail with less than 41 active S-boxes for 4 rounds has been identified so far, corresponding to a empirical maximum differential probability of $3^{-2 \cdot 41} = 3^{-82}$ over 4 rounds already.

We consider it highly unlikely that a differential trail over 5 rounds with a

probability higher than $3^{243/2}$ exists, but proving it is beyond the available computational resources.

We therefore follow a conservative approach and determine the number of rounds according to the provable bound, adding 4 extra rounds (which corresponds to full diffusion). This results in $R = 24$, which should offer a comfortable long-term security margin.

4 Security analysis

In this section, we briefly summarize the cryptanalysis results obtained on Troika so far.

4.1 Security of the overall construction

Since the design of Troika follows the sponge construction, the generic bounds proven for the security of cryptographic sponges apply to Troika as well. According to Theorem 2 of [3, 4], when a sponge with capacity c trits and instantiated with a random permutationis is used as a hash function with output length $n = c/2$ trits, the following bounds apply for its collision, second preimage and preimage resistance:

Collisions: Negligible success probability if less than $3^{n/2}$ queries.

Second preimages: Negligible success probability if less than 3^n queries.

Preimages: Negligible success probability if less than 3^n queries.

In the case of Troika, these bounds apply for $n = 243$.

In particular, this means that any attack on the collision, second preimage or preimage resistance of Troika that is faster than the guarantees offered by these security bounds, will necessarily need to exploit properties and weaknesses of the underlying permutation f. Our security evaluation of Troika therefore focuses on finding attacks against the permutation f.

4.2 Ternary differential cryptanalysis

Originally proposed by Biham and Shamir [6], differential cryptanalysis is one of the most influential techniques for the analysis of symmetric primitives. This is particularly the case for hash functions, where the security objectives such as collision resistance find very natural expression in terms of difference propagations. Since the exact definitions can vary, we recast the main notions of differential cryptanalysis over $\mathbb{F}_3^n$ in the following.

Let $a, b \in \mathbb{F}_3^n$. The *difference* of a and b is defined by

$$\Delta(a, b) \stackrel{\text{def}}{=} a - b, \tag{4.1}$$

where the subtraction is the inverse of the usual componentwise addition in $\mathbb{F}_3^n$. The differential input-output behaviour of a given function mapping n trits to q trits is completely characterized by its difference distribution table: Given a function $f : \mathbb{F}_3^n \to \mathbb{F}_3^q$, the *difference distribution table* (DDT) of f is defined as

$$\begin{aligned} D_f(\alpha, \beta) &\stackrel{\text{def}}{=} \#\{x \in \mathbb{F}_3^n \mid \Delta(f(x+\alpha), f(x)) = \beta\} \\ &= \#\{x \in \mathbb{F}_3^n \mid f(x+\alpha) - f(x) = \beta\}. \end{aligned} \tag{4.2}$$

The *differential uniformity* of a mapping $f : \mathbb{F}_3^n \to \mathbb{F}_3^q$ is then defined as

$$\Delta_f = \max_{\alpha \neq 0, \beta} D_f(\alpha, \beta), \tag{4.3}$$

and the related δ*-parameter* of f is given by

$$\delta_f = \max_{\alpha \neq 0, \beta} \Pr[\alpha \xrightarrow{f} \beta] = \Delta_f / 3^n. \tag{4.4}$$

4.3 Differential properties of the S-box

A direct computation of the ternary difference distribution table shows that, as claimed, Troika's S-box has differential uniformity $\Delta_s = 3$, hence its maximum differential probability is $\delta_s = 3^{-2} \approx 2^{-3.17}$.

Comparing this to random 3-trit S-boxes, we found that out of 10^7 such S-boxes, only $\simeq 0.05\%$ had the same differential uniformity 3. No S-box with differential uniformity 2 was observed, and on average, the differential uniformity ranged between 5 and 6. The S-box of Troika therefore compares favourably to random S-boxes.

4.4 Properties of the diffusion layer

We can view the three-dimensional state of $9 \times 3 \times 27$ trits as a one-dimensional state of 729 trits as follows:

$$\bar{x} = \left[x_{0,0,0}, \cdots, x_{8,0,0}, \quad x_{0,1,0}, \cdots, x_{8,1,0}, \quad \cdots, \quad x_{8,2,26}\right]^T. \tag{4.5}$$

4.4.1 AddColumnParity as a column parity mixer

We can express the AddColumnParity mapping in the framework of column parity mixers [27]. A column parity mixer θ is a linear mapping defined by a square *parity-folding matrix* Z which adds the θ*-effect* of this matrix Z on a matrix A to A itself. Adapting [27, Definition 5] to the finite field $\mathbb{F}_p$, with $A \in \mathbb{F}_p^{m\times n}$ and $Z \in \mathbb{F}_p^{n\times n}$, the column parity mixer θ using the parity-folding matrix Z is defined as:

$$\theta(A) = A + \mathbf{1}_m^m AZ, \tag{4.6}$$

where $\mathbf{1}_m^m$ is the $m \times m$ matrix with all elements equal to 1.

By direct inspection of Equation (2.15), we can now express the AddColumnParity mapping as a column parity mixer by deriving the corresponding parity-folding matrix Z as the block matrix

$$Z = Z_{i,j}, \quad Z_{i,j} \in \mathbb{F}_3^{27\times 27}, 0 \le i, j < 27, \tag{4.7}$$

where

$$Z_{i,j} = \begin{cases} Z_1 & \text{for } i = j, \\ Z_1^T & \text{for } i \equiv (j-1) \bmod 27, \\ 0^{27\times 27} & \text{otherwise;} \end{cases} \tag{4.8}$$

which corresponds to

$$Z = \begin{bmatrix} Z_1 & & & & & Z_1^T \\ Z_1^T & Z_1 & & & & \\ & Z_1^T & Z_1 & & & \\ & & & \ddots & & \\ & & & & Z_1^T & Z_1 \end{bmatrix}, \tag{4.9}$$

where the matrix Z_1 is defined as

$$Z_1 = \begin{bmatrix}
0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0
\end{bmatrix}. \tag{4.10}$$

Due to the structure of Z, the matrix Z', corresponding to the parity-folding matrix of the column parity mixer for the inverse of AddColumnParity, is given by

$$Z' = -Z. \tag{4.11}$$

We note here that even though this matrix Z' is the parity-folding matrix of AddColumnParity^{-1}, it is not the inverse of Z since Z is not of full rank and therefore not invertible. The corresponding mapping AddColumnParity^{-1} is described by the transformation matrix $M^{-1} = Z' + I$, which is invertible.

4.4.2 Branch numbers

Initially proposed by Daemen [9], the differential and linear branch numbers of a linear mapping are a universally accepted measure for their diffusion properties, and hence the resistance of cryptographic primitives against differential and linear cryptanalysis [10, 23]. Let $\theta : \mathbb{F}_p^n \to \mathbb{F}_p^n$ a linear mapping described by $x \mapsto Mx$ for the transformation matrix M and denote by $w(x)$ the number of nonzero entries of a vector x. The differential and linear branch numbers $\mathcal{B}_d(\theta)$ and $\mathcal{B}_d(\theta)$ are then defined as

$$\mathcal{B}_d(\theta) = \min_{0 \neq x \in \mathbb{F}_p^n} \{w(x) + w(\theta(x))\} \tag{4.12}$$

$$\mathcal{B}_l(\theta) = \min_{0 \neq x \in \mathbb{F}_p^n} \{w(x) + w(\theta^T(x))\}, \tag{4.13}$$

where θ^T is the mapping characterized by $x \mapsto M^T x$ with the transformation matrix M^T, which is the transpose of M.

For our concrete state layout of $9 \times 3 \times 27$ elements over $\mathbb{F}_3$, we refine the notion of a state weight as

$$w(x) = \#\{(i,j,k) \mid x_{i,j,k} \neq 0 \text{ for } 0 \leq i < 9, 0 \leq j < 3, 0 \leq k < 27\}|, \tag{4.14}$$

and analogously the weight of a state x when viewed as trytes:

$$w(\mathsf{x}) = \#\{(i,j,k) \mid \mathsf{x}_{i,j,k} \neq 0 \text{ for } 0 \leq i < 3, 0 \leq j < 3, 0 \leq k < 27\}. \tag{4.15}$$

We note that the tryte weight $w(\mathsf{x})$ is equal to the number of active S-boxes in x.

For the differential and linear branch numbers of AddColumnParity, we have the following result.

Theorem 4.1. *The differential and linear branch numbers* $\mathcal{B}_d(\text{AddColumnParity})$ *and* $\mathcal{B}_l(\text{AddColumnParity})$ *of the* AddColumnParity *mapping are equal to 4.*

Proof. We first focus on the differential branch number $\mathcal{B}_d(\text{AddColumnParity})$. It is obvious that $\mathcal{B}_d(\text{AddColumnParity}) \leq 4$ since there are states of weight 2 with a zero column parity for all columns, for instance the state s defined as follows:

$$s = \left(\begin{array}{ccccccccc} & & & & z = 0: & & & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right), \tag{4.16}$$

resulting in the following transition through AddColumnParity:

$$\text{AddColumnParity}(s) = s,$$

which with $w(s) = 2$ means that $\mathcal{B}_d(\text{AddColumnParity})$ is upper bounded by $w(s) + w(s) = 2 + 2 = 4$. Since AddColumnParity is an invertible linear mapping, $\mathcal{B}_d(\text{AddColumnParity}) \geq 2$ by definition. It remains to prove that there are no transitions $s \mapsto \text{AddColumnParity}(s)$ such that $w(s) + w(\text{AddColumnParity}(s)) = 3$. This condition can be fulfilled if and only if a state s of weight 1 is mapped to a state of weight 2 by either AddColumnParity or $\text{AddColumnParity}^{-1}$. We consider these two cases in turn.

Case 1: Let s be a state with $w(s) = 1$, i.e. with a single active trit in position $s_{i,j,k}$. Then the column parity is equal to ± 1 for column j and zero everywhere else. By Equation (2.15), there are two columns that are affected by the nonzero parity in column j, specifically columns $j - 1$ in the same slice and $j + 1$ in slice $k + 1$, both of which are zero in s. This implies that $w(\text{AddColumnParity}(s)) = 3 + 3$, therefore $w(s) + w(\text{AddColumnParity}(s)) = 1 + 6 = 7$.

Case 2: Let $\text{AddColumnParity}(s)$ be a state with $w(\text{AddColumnParity}(s)) = 1$, i.e. with a single active trit in position $\text{AddColumnParity}(s)_{i,j,k}$, leading to a single-trit nonzero parity in column j. By Equation (4.11) and the definition of the matrix Z (4.8), we see that 6 columns in $\text{AddColumnParity}^{-1}(\text{AddColumnParity}(s)) = s$ are affected by this nonzero parity. This implies that $w(s) = 6$, therefore we have $w(s) + w(\text{AddColumnParity}(s)) = 6 + 1 = 7$.

In both cases, we always have $w(s) + w(\text{AddColumnParity}(s)) > 3$, therefore $\mathcal{B}_d(\text{AddColumnParity}) = 4$.

For the linear branch number $\mathcal{B}_l(\text{AddColumnParity})$, we note that exactly the same reasoning applies to the transpose of the parity-folding matrix Z. □

The linear branch number $\mathcal{B}_l(\text{AddColumnParity})$ is therefore equal to the differential branch number, and we collectively refer to this number as *the* branch number of AddColumnParity.

Considering the tryte weight instead of the trit weight, the result of Theorem 4.1 imply that there are at least 4 active S-boxes per two rounds:

Corollary 4.2. *The* AddColumnParity *mapping guarantees at least 4 active S-boxes every two rounds of the Troika permutation:*

$$\min_{0 \neq \mathsf{x} \in \mathbb{F}_3^{3\times 3\times 27}} w(\mathsf{x}) + w(\text{AddColumnParity}(\mathsf{x})) = 4. \tag{4.17}$$

Proof. The state defined in Equation (4.16) also has tryte weight $w(\mathbf{s}) = 2$, maps to itself under the AddColumnParity mapping, and therefore shows that there can be at most 4 active S-boxes per two rounds of the Troika permutation. By the same reasoning as in the proof of Theorem 4.1, there need to be at least two active S-boxes per two rounds.

It therefore only remains to show that the case of 3 active S-boxes is impossible. This can happen if and only if a state of tryte weight 1 is mapped to a state of tryte weight 2 by either AddColumnParity or AddColumnParity^{-1}. We distinguish three cases, depending on the number of active trits corresponding to the single active S-box.

Case 1: Let s be a state with $w(\mathbf{s}) = 1$ and $w(s) = 1$, i.e. with a single active trit in a single active S-box. Then the column parity is equal to ± 1 in one column and zero everywhere else. It now follows from Case 1 in the proof of Theorem 4.1 that there are 6 active trits in AddColumnParity(s) across 3 different rows, which means that there are at least 3 active S-boxes in AddColumnParity$(\mathbf{s})$.

Case 2: Let s be a state with $w(\mathbf{s}) = 1$ and $w(s) = 2$, i.e. with two active trits in a single active S-box. It can be shown by exhaustive computer search over all $27 \cdot 3 \cdot 3 \cdot \binom{3}{2} \cdot |\{1, 2\}|^2 = 2916$ possible such states s that for AddColumnParity and AddColumnParity^{-1}, the image of s always has at least 9 active S-boxes.

Case 3: Let s be a state with $w(\mathbf{s}) = 1$ and $w(s) = 3$, i.e. with three active trits in a single active S-box. It can be shown by exhaustive computer search over all $27 \cdot 3 \cdot 3 \cdot \binom{3}{3} \cdot |\{1, 2\}|^3 = 1944$ possible such states s that for AddColumnParity and AddColumnParity^{-1}, the image of s always has at least 12 active S-boxes.

Summarising, in all three possible cases, $w(\mathbf{s}) + w(\text{AddColumnParity}(\mathbf{s})) > 3$, and $w(\mathbf{s}) + w(\text{AddColumnParity}^{-1}(\mathbf{s})) > 3$, which proves the claim. □

4.4.3 Overall diffusion

One way to measure the overall diffusion quality of Troika is to check the *avalanche effect* for each trit of the state. If we alter one of the input trits we would expect that $\approx 2/3$ of the state are changed if the permutation we use is ideal. The size of the permutation of Troika is too large to test for all possible inputs. We therefore use a Monte-Carlo approach by generating a random state S and altering the trit at some position i to obtain the state S_i.

We then compute the number of trits which are different after applying the round function.

We repeat this for all choices of i, repeat the whole process for different choices of S and keep a statistic of $w(\Delta(S, S_i))$, the number of nonzero trits in the difference between S and S_i. We are now interested in finding the number of rounds such that

$$\frac{w(\Delta(S, S_i))}{729} \approx 2/3 \qquad i = 0, \ldots, 728. \tag{4.18}$$

The experiments show that we need ≈ 4 rounds in both forward and backward direction to get this effect (see Table 4.1 and Table 4.2).

Table 4.1: Measuring the avalanche effect of the Troika permutation. The statistics are on the value of $w(\Delta(S, S_i))$ for a sample size of 100 random states.

Rounds	Mean	Std. Deviation	Max	Min	Median
1	15.5	4.7	21	7	14
2	188.0	51.0	279	83	183
3	483.2	18.2	532	434	483
4	485.5	12.4	515	457	487
5	489.1	12.6	516	450	488
6	487.2	13.9	520	453	486
7	484.8	12.4	516	453	486
8	484.4	11.5	512	461	485
9	487.6	12.7	518	454	488

4.5 Differential trails

4.5.1 Trail search

In order to find the differential trails with the lowest count of active S-boxes we use a tool assisted approach based on the publicly available tool CryptoSMT [26]. For this we constructed a ternary SMT model which resembles how differences propagate through the round function of Troika. We can then use this to generate a proof that no differential trail with less than t S-boxes exist and also obtain the differential trail with t S-boxes.

The combination of a relatively large state size of Troika and the fact that the ShiftLanes and AddColumnParity mappings operate on individual trits

Table 4.2: Measuring the avalanche effect of the inverse Troika permutation. The statistics are on the value of $\delta(S, S_i)$ for a sample size of 10 random states.

Rounds	Mean	Std. Deviation	Max	Min	Median
1	15.4	1.8	17	12	16
2	180.3	24.3	205	135	187
3	476.8	12.8	494	460	476
4	481.4	16.0	510	457	479
5	490.6	13.2	514	471	490
6	485.7	20.0	517	458	481
7	483.3	15.3	500	457	486
8	483.7	10.0	497	463	485
9	489.7	11.1	514	479	488

instead of trytes (which is the unit of the S-box) implies that it is computationally infeasible to accurately determine the best differential trails, and hence identify tight bounds for the minimum number of active S-boxes, over any significant number of rounds. This situation for instance also arises in the case of SHA-3, where there is a significant gap between provable bounds and actually known trails [5, 21].

In order to simplify the analysis, we define scaled-down versions of the Troika permutation by reducing the number of slices (lower dimension in the z-axis), and taking the shifts of the ShiftLanes operation modulo the reduced lane size. Denoting the number of slices by $|z|$, we considered the four variants $|z| = 1, 3, 9, 27$ (note that $|z| = 27$ corresponds to the full Troika as specified).

4.5.2 Bounds for differential trails

The results of our search for the bounds for differential trails over different numbers of rounds for f and its variants with a smaller number of slices ($|z| = 1, 3, 9$) are summarized in Table 4.3. We first note that the higher $|z|$, the more informative the bound becomes. In particular, the bound of at least 69 active S-boxes per 24 rounds for a single slice is not tight for the full design, since the bound of 25 active S-boxes per 4 rounds already implies at least $24/4 \cdot 25 = 150$ active S-boxes over 24 rounds.

We find that over 4 rounds, there are at least 25 active S-boxes. This translates into an upper bound of $3^{-2 \cdot 25} = 3^{-50}$ for the maximum probability of a differential trail over 4 rounds. Therefore, differential trails over 20 rounds of Troika have a maximum probability of 3^{-250}, which exceeds the security pa-

Number of rounds r	Number of slices $\lvert z\rvert$			
	1	3	9	27
2	4	4	4	4
3	7	11	11	12
4	10	21	$\boxed{25}$	–
5	11	28	–	–
8	20	–	–	–
16	45	–	–	–
24	69	–	–	–

Table 4.3: Provable minimum numbers of active S-boxes over the Troika permutation over different numbers of rounds r for various numbers of slices $\lvert z\rvert$. Note that $\lvert z\rvert = 27$ corresponds to the full Troika permutation, and that the number of active S-boxes for a certain value of $\lvert z\rvert$ are lower bounds for the number of active S-boxes for higher values of $\lvert z\rvert$. – indicates that no better bound could be obtained within the available computational resources.

rameter of 243 trit security for second preimages and preimages. It is important to note that these extrapolated numbers for higher numbers of rounds are lower bounds on the number of active S-boxes: the actual bound is likely much higher.

We summarize the corresponding upper bounds for the probability of differential trails over various number of rounds in Table 4.4.

4.6 Meet-in-the-middle attacks

To evaluate the strength of Troika with regard to guess-and-determine or meet-in-the-middle attacks, we compute the number of S-boxes that are required to compute S-boxes a number of rounds forward or backward. In the forward direction, computing one trit after the AddColumnParity operation requires the knowledge of seven trits. Hence, computing one S-box can require at least 3 and at most 21 S-boxes of the previous round. Combined with the entire linear layer, we find that the computation of one S-box requires the knowledge of at between 18 and 19 S-boxes of the previous round. After two rounds, computing one S-box requires the knowledge of between 188 and 195 S-boxes. After three rounds, all S-boxes are required. The properties of

Number of rounds r	Probability of r-round differential trail
1	$\leq 3^{-2}$
2	$\leq 3^{-8}$
3	$\leq 3^{-24}$
4	$\leq 3^{-50}$
6	$\leq 3^{-58}$
8	$\leq 3^{-100}$
16	$\leq 3^{-200}$
20	$\leq 3^{-250}$
24	$\leq 3^{-300}$

Table 4.4: Upper bounds for for probability of differential trails over r rounds of Troika. Note that for $r \geq 4$, these bounds are not tight, as no corresponding trail is known.

the internal permutation in the backward direction are similar.

This means that a meet-in-the-middle distinguisher can be extended by at most 3 rounds at the beginning and 3 rounds at the end, suggesting that 7 (out of the 24) rounds of Troika offer sufficient protection against meet-in-the-middle attacks.

4.7 Algebraic attacks

Resistance against algebraic attacks is closely related to the degree of the cryptographic primitive. Since we have the relation $t^3 = t$, the degree of a function $g : \mathbb{F}_3^n \to \mathbb{F}_3$ is lower than $2n$.

According to its definition, we have

$$s(x+1, y, z) = (-x^2y - xy^2z + yz^2 + xz + y, xy^2 + yz + x, xy + z) \tag{4.19}$$

for the Troika S-box. We note that one of its component functions has degree 2 only. However, after the ShiftRows operation, each column has one degree 4 component and then, after the AddRoundConstant operation, each trit has degree 4. The maximal degree of the Troika permutation is $2 \times 729 = 1458$ and thus the number of rounds R should be chosen such that $4^R \geq 1458$ which is equivalent to $R \geq 6$. The same result can be obtained for the inverse.

The results of Section 4.6 imply that the degree of Troika can be controlled for particular inputs or outputs over at most 3 rounds, suggesting that 9 rounds offer sufficient protection against algebraic attacks.

4.8 Ternary linear cryptanalysis

Linear cryptanalysis was originally proposed by Matsui [19,20] for the analysis of the DES, and has since then established itself as one of the primary cryptanalytic tools for the analysis of keyed symmetric primitives. Even though is not directly applicable in the context of unkeyed hash functions [12], we nevertheless analyze the resistance of the S-box layer, the only nonlinear part of the Troika round transformation, against linear cryptanalysis over $\mathbb{F}_3$.

Differential and linear properties of pseudorandom permutation families have been studied for general Abelian groups by Granboulan et al. [13]. The key notion for linear cryptanalysis over $\mathbb{F}_3$ is the notion of a ternary *bias vector* $L_f(\alpha, \beta)$ which is defined as

$$L_f(\alpha, \beta) = \left(\#\{x \in \mathbb{F}_3^n \mid \alpha^T x - \beta^T f(x) = u\} - 3^{n-1}\right)_{u \in \mathbb{F}_3}$$

for a function $f : \mathbb{F}_3^n \to \mathbb{F}_3^n$ and input and output masks $\alpha, \beta \in \mathbb{F}_3^n$. The collection of all bias vectors $L_f(\alpha, \beta)$ over all α, β is then the ternary equivalent of a linear approximation table. As a measure of f with respect to ternary linear cryptanalysis, we use the ternary linear potential TLP_f, which is given by

$$TLP_f \stackrel{\text{def}}{=} 3^{-2n} \cdot \left(\max_{\alpha \in \mathbb{F}_3^n, 0 \neq \beta \in \mathbb{F}_3^n} \left(\max_{u \in \mathbb{F}_3} \left(L_f(\alpha, \beta)_u \right)^2 \right) \right), \tag{4.20}$$

such that complexity of a linear attack is then proportional to $1/TLP$.

For the Troika S-box, we find that the maximum entry of any ternary bias vector has magnitude 6, which means that the TLP of the S-box is equal to $(\frac{6}{27})^2 = \frac{4}{81} \approx 3^{-2.74} \approx 2^{-4.34}$. This implies that Troika with 25 active S-boxes per 4 rounds will have 4-round linear trails with a linear probability of at most $3^{-68.5}$ and 16-round trails with a linear probability of at most 3^{-274}.

4.9 Invariant attacks

Attacks based on linear or nonlinear invariants of the round function of a block cipher have been receiving significant attention in recent years. They exploit self-similarities in the round function, which are normally meant to be avoided by means of round constants. The recently proposed invariant attacks however show that the round constants need to be chosen carefully in order to fulfill their purpose.

All these attacks are key-recovery attacks on block ciphers or authenticated encryption schemes, and not directly applicable to hash functions. We nevertheless investigate the security of the internal permutation of Troika permutation against these attacks.

A framework for using linear invariants for cryptanalysis has first been proposed in 2011 by Leander et al. in form of the invariant subspace attack [17].

Nonlinear invariant attacks are an extension of invariant subspace attacks and were proposed by Todo et al. in 2016 [28]. are based on finding a nonlinear function $g : \mathbb{F}_3^n \to \mathbb{F}_3$ such that

$$\forall x \in \mathbb{F}_3^n . g(f(x+k)) - g(x) = c \tag{4.21}$$

for a fixed constant $c \in \mathbb{F}_3^n$.

A criterion for resistance against invariant subspace and nonlinear invariant attacks has been proposed in [1]. However, a very recent work [29] shows that the criterion for the design of the round constants of [1] is not sufficient in all cases. In particular, [29] introduces the concept of generalized nonlinear invariants, which generalize both invariant subspaces and regular nonlinear invariants. It is based on finding a function $g : \mathbb{F}_3^n \to \mathbb{F}_3$ such that

$$\forall x \in \mathbb{F}_3^n . g(x + a_1) - g(f_k(x) + a_2) = c \tag{4.22}$$

for constants $a_1, a_2, c \in \mathbb{F}_3^n$. This approach allows to absorb some additive influence from the round constants by choice of the constants a_1 and a_2. Furthermore, the authors of [29] demonstrate that not only the round constants, but also the S-box plays a crucial role in the robustness of a design against the generalized nonlinear invariant attack. A new criterion for robust round constants was proposed in [29], and it has been verified that the combination of S-box and round constants of Troika fulfills this criterion.

Bibliography

[1] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving resistance against invariant attacks: How to choose the round constants. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 647–678. Springer, 2017.

[2] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document. Submission to NIST SHA-3 competition (Round 3), 2011. `http://keccak.noekeon.org/`.

[3] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[4] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions, version 0.1, January 2011. `https://keccak.team/files/CSF-0.1.pdf`.

[5] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, version 3.0. Submission to NIST (SHA-3 competition, final round), 2011.

[6] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.

[7] Alex Biryukov and David A. Wagner. Slide attacks. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.

[8] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid

Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[9] Joan Daemen. *Cipher and hash function design strategies based on linear and differential cryptanalysis*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1995.

[10] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. Linear frameworks for block ciphers. *Des. Codes Cryptography*, 22(1):65–87, 2001.

[11] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer-Verlag, 2002.

[12] P. Gauravaram, L.R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S.S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST (SHA-3 competition, final round), 2011.

[13] Louis Granboulan, Éric Levieil, and Gilles Piret. Pseudorandom permutation families over abelian groups. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 57–77. Springer, 2006.

[14] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Donggeon Lee. LEA: A 128-bit block cipher for fast encryption on common processors. In Yongdae Kim, Heejo Lee, and Adrian Perrig, editors, *Information Security Applications - 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers*, volume 8267 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2013.

[15] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A new block cipher suitable for low-resource device. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.

[16] Andreas Hülsing. W-OTS+ - shorter signatures for hash-based signature schemes. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *Progress in Cryptology - AFRICACRYPT 2013, 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24,*

2013. Proceedings, volume 7918 of *Lecture Notes in Computer Science*, pages 173–188. Springer, 2013.

[17] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of printcipher: The invariant subspace attack. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2011.

[18] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, revised edition, 1994.

[19] Mitsuru Matsui. Linear cryptoanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[20] Mitsuru Matsui. The first experimental cryptanalysis of the Data Encryption Standard. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.

[21] Silvia Mella, Joan Daemen, and Gilles Van Assche. New techniques for trail bounds and application to differential trails in keccak. *IACR Transactions on Symmetric Cryptology*, 2017(1):329–357, Mar. 2017.

[22] Kent E. Morrison. Random maps and permutations. Technical report, California Polytechnic State University, San Luis Obispo, 1998.

[23] Vincent Rijmen. *Cryptanalysis and design of iterated block ciphers*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.

[24] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357. Springer, 2011.

[25] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.

[26] Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. `https://github.com/kste/cryptosmt`.

[27] Ko Stoffelen and Joan Daemen. Column parity mixers. *IACR Transactions on Symmetric Cryptology*, 2018(1):126–159, Mar. 2018.

[28] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear invariant attack - practical attack on full scream, iscream, and midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[29] Yongzhuang Wei, Tao Ye, Wenling Wu, and Enes Pasalic. Generalized nonlinear invariant attack and a new design criterion for round constants. *IACR Transactions on Symmetric Cryptology*, 2018(4), to appear 2019.

Changelog

Below we summarize the changes since version 1.0 of this document. Note that the values of the preimage challenges are unchanged.

- Figure 2.2, Algorithm 1, Figure 2.3: removed one slice too many that was marked in orange and one superfluous permutation call from the sponge construction. Table 2.4 updated accordingly. Thanks to Silvia Mella for reporting this.
- Section 2.3.5: fixed typo in the intial value for the round constant LFSR. Thanks to Thomas Pototschnig for reporting this.

Contact

info@cyber-crypt.com
+45 78 74 69 80

Address

cybercrypt A/S
Tuborg Blvd. 12, 3rd floor
2900 Hellerup
Denmark
CVR 37664480