Opening the Blackbox: Collision Attacks on Round-Reduced Tip5, Tip4, Tip4' and Monolith

Fukang Liu[1], Katharina Koschatko[2], Lorenzo Grassi[3,4], Hailun Yan[5], Shiyao Chen[6], Subhadeep Banik[7] and Willi Meier[8]

[1] Institute of Science Tokyo, Tokyo, Japan,
liu.f.ad@m.titech.ac.jp
[2] Graz University of Technology, Graz, Austria
katharina.koschatko@tugraz.at
[3] Ponos Technology, Zug, Switzerland,
[4] Ruhr University Bochum, Bochum, Germany
lorenzo.grassi@rub.de
[5] University of Chinese Academy of Sciences, Beijing, China
hailun.yan@ucas.ac.cn
[6] Digital Trust Centre, Nanyang Technological University, Singapore, Singapore
shiyao.chen@ntu.edu.sg
[7] Universita della Svizzera Italiana, Lugano, Switzerland
subhadeep.banik@usi.ch
[8] University of Applied Sciences and Arts Northwestern Switzerland, Windisch, Switzerland
willimeier48@gmail.com

Abstract. A new design strategy for ZK-friendly hash functions has emerged since the proposal of Reinforced Concrete at CCS 2022, which is based on the hybrid use of two types of nonlinear transforms: the composition of some small-scale lookup tables (e.g., 7-bit or 8-bit permutations) and simple power maps over $\mathbb{F}_p$. Following such a design strategy, some new ZK-friendly hash functions have been recently proposed, e.g., Tip5, Tip4, Tip4', and the Monolith family. All these hash functions have a small number of rounds, i.e., 5 rounds for Tip5, Tip4, and Tip4', and 6 rounds for Monolith (recently published at ToSC 2024/3). Using the composition of some small-scale lookup tables to build a large-scale permutation over $\mathbb{F}_p$ – which we call S-box – is a main feature in such designs, which can somehow enhance the resistance against the Gröbner basis attack because this large-scale permutation will correspond to a complex and high-degree polynomial representation over $\mathbb{F}_p$.

As the first technical contribution, we propose a novel and efficient algorithm to study the differential property of this S-box and to find a conforming input pair for a randomly given input and output difference. For comparison, a trivial method based on the use of the differential distribution table (DDT) for solving this problem will require time complexity $\mathcal{O}(p^2)$.

For the second contribution, we also propose new frameworks to devise efficient collision attacks on such hash functions. Based on the differential properties of these S-boxes and the new attack frameworks, we propose the first collision attacks on 3-round Tip5, Tip4, and Tip4', as well as 2-round Monolith-31 and Monolith-64, where the 2-round attacks on Monolith are practical. In the semi-free-start (SFS) collision attack setting, we achieve practical SFS collision attacks on 3-round Tip5, Tip4, and Tip4'. Moreover, the SFS collision attacks can reach up to 4-round Tip4 and 3-round Monolith-64. As far as we know, this is the first third-party cryptanalysis of these hash functions, which improves the initial analysis given by the designers.

Keywords: Tip5/Tip4/Tip4' · Monolith · (Semi-Free Start) Collisions

1 Introduction

Zero-knowledge (ZK) proof systems and other use cases have seen a rise in popularity in the last couple of years. Recursive SNARKs and folding schemes (incrementally verifiable computation, or IVC [Val08]) provide a possibility to increase the performance in this setting and more and more programs contain hash functions as subroutines. However, in IVC applications, *arithmetization-oriented* hash functions are preferred instead of standard hash functions like SHA-3 or Blake3. This is because the size of hash functions as an arithmetic circuit over a prime field becomes more important than the "native" software performance (e.g., on an x86 architecture). Motivated by such a new requirement, a dozen ZK-friendly primitives have been designed and published in recent years, including MiMC [AGR+16], Marvellous [AD18, AAB+20], Poseidon [GKR+21], Griffin [GHR+23], Neptune [GOPS22], Reinforced Concrete [GKL+22], Anemoi [BBC+23], Tip5 [SLS+23], Tip4/Tip4' [Sal23], Monolith [GKL+24], XHash8/XHash12 [ABK+23], and Arion [RST23].

Although many symmetric primitives with different new features have been proposed for ZK proof systems, some of them do not stand the test of time and soon got broken due to the usage of less-studied components. Concrete examples include the recent FreeLunch attack [BBL+24] against Griffin, Anemoi, and Arion, several algebraic and statistical attacks against Poseidon [BBLP22, KR21, GRS21, BCD+20], algebraic cryptanalysis of the MARVELlous family member Friday [ACG+19], the so-called six worlds of Gröbner basis cryptanalysis of Anemoi [KLR24], and resultant-based algebraic attacks [YZY+24] on Rescue-Prime, Anemoi, and Jarvis. Moreover, there are also several other attacks against symmetric-key primitives for advanced protocols like secure multiparty computation (MPC) and homomorphic encryption (HE) [LAW+23, ZWY+23, LMØM23, GBJR23, GAH+23, GKRS22, LSW+22, CHWW22, LSMI21, BBVY21, Din21, EGL+20, LKSM24]. For these reasons, it is quite important to study the properties of the special S-box used in Tip5, Tip4 and in Monolith, considering the fact that these ciphers have a small number of rounds.

Tip5, Tip4, Tip4' and Monolith. Among these ZK-friendly hash functions, Tip5, Tip4, Tip4' and the Monolith family follow a new design strategy first used in Reinforced Concrete, which is to construct a large-scale permutation over a finite field $\mathbb{F}_p$ with the composition of several small-scale lookup tables. For convenience, such a special large-scale permutation over $\mathbb{F}_p$ is called S-box in this paper. Different from the common way to use simple power maps to construct a nonlinear permutation over $\mathbb{F}_p$, the polynomial representation of this S-box over $\mathbb{F}_p$ will be a complex and high-degree polynomial over $\mathbb{F}_p$. Hence, using this S-box can somehow enhance the resistance against the algebraic attack, e.g., the Gröbner basis attack. Indeed, due to the fact that the algebraic polynomial description of such S-boxes is of extremely high degree and dense (besides being unknown in many cases), the cost of such algebraic attacks becomes prohibitive.

Designers' analysis. The best attacks found by designers consist of setting up a system of equations that describes the considered schemes. In order to remove the influence of the S-boxes (which are treated as a black box), the attacker exploits the available degrees of freedom in order to impose the inputs – and so the outputs – of such S-boxes to be equal to some fixed, known values. (Remember that no secret is involved since these schemes are hash functions; hence, this strategy is always possible.) This is possible since the number of such S-boxes per round is (much) smaller than the state size. Still, it is easy to understand that this attack strategy can only cover a limited number of rounds, which obviously depends on the available degrees of freedom.

Our goal. For these ZK-friendly hash functions, both preimage resistance and collision resistance are of crucial importance. The reason for this is related to the fact that Merkle

Trees are used in blockchain systems in order to efficiently summarize all transactions in a block. Indeed, this structure allows the users to verify whether a particular transaction is included in a block without downloading the entire blockchain. Focusing on root collisions in Merkle Trees, they could potentially undermine the security mechanisms of blockchain networks [Kil92, BSBHR18], leading to vulnerabilities in data integrity and authentication processes. As far as we know, there is no successful attack yet on these hash functions by exploiting the details of this type of S-box. Hence, it is interesting and important to look into this S-box, and study whether there are some properties to be exploited for efficient attacks. This is exactly the goal of this paper for the particular case of collision attacks.

Our Contributions

A secure hash function is required to satisfy collision resistance. Therefore, in this work, we focus on collision attacks on Tip5, Tip4, Tip4' and the Monolith family, and hence we will study the differential properties of the special S-box used in these ciphers. Our contribution is briefly summarized below.

Differential property of Tip5/Monolith S-boxes. We propose a novel and efficient algorithm to determine whether a randomly given input-output difference pair $(\Delta w, \Delta z) \in \mathbb{F}_p^2$ is valid for this special S-box $z = S(w)$ where $z, w \in \mathbb{F}_p$. Moreover, with this algorithm, retrieving all possible conforming input pairs for a valid input-output difference pair is also efficient. As already stated, the polynomial representation of this S-box over $\mathbb{F}_p$ is complex and of high degree, and hence the common method to solve these problems like building the differential distribution table (DDT) or directly solving the univariate differential equation $S(\Delta w + w) - S(w) = \Delta z$ become impractical for a large finite field.

Efficient collision attack frameworks. We propose two attack frameworks to devise efficient (SFS) collision attacks on round-reduced Tip5, Tip4, Tip4', and Monolith, which can take full advantage of our algorithm for the S-box. With these techniques, we can remove the influence of the high-degree polynomial representation of the S-box by consuming as few degrees of freedom as possible, and meanwhile, it is still possible to set up a system of low-degree equations to generate (SFS) collisions, where the number of variables is at least the same as the number of equations. Specifically, by means of our algorithm for the S-box, we are able to efficiently find non-zero input-output difference pairs of the S-box satisfying certain conditions to generate (SFS) collisions instead of always merely forcing the input and output differences of the S-box to be 0 in order to skip it. In this way, we significantly improve the straightforward collision attacks on these ciphers independent of the S-box.

Our results. As the first third-party cryptanalysis, we can mount:

- collision attacks on 3-round Tip5, Tip4, and Tip4',
- practical collision attacks on 2-round Monolith-64 and Monolith-31,
- SFS collision attacks on 4-round Tip4 (besides practical SFS collisions for 3-round Tip5, Tip4 and Tip4'),
- SFS collision attacks on 3-round Monolith-64.

We have verified these attacks as far as we can, and the estimated time complexity of the attacks is as expected. In particular, we provide practical SFS collisions for 3-round Tip5, Tip4, and Tip4', and colliding message pairs for 2-round Monolith-31 and Monolith-64. All the source codes are available at https://github.com/IAIK/ca-tip5family-monolith.

Table 1: Summary of our attacks on Tip5, Tip4, Tip4', Monolith-31 and Monolith-64. We recall that the Tip5 family and Monolith family are instantiated by 5 and 6 rounds, respectively, and they target the 128-bit security (with the only exception of Monolith-31, which targets 124-bit security). The memory complexity is always smaller than the time complexity (we recall that the linear algebra constant ω is bounded by $2 \leq \omega \leq 3$).

Attack type	Target	Rounds	(Estimated) Complexity	$\omega = 2.37$	Reference
Collision	Tip5	3	$\max\{2^{54.3\omega}, 2^{121.1}\}$	$2^{128.7}$	Sect. 3.2
			$2^{101.1} + \max\{2^{31.7\omega+16}, 2^{99.3}\}$	$2^{101.1}$	Sect. 6.1
			$\max\{2^{8+38.4\omega}, 2^{107.7}\}$	$2^{107.7}$	Sect. 6.1
	Tip4	3	$\max\{2^{47.1\omega}, 2^{104.1}\}$	$2^{111.5}$	Sect. 3.2
			$\max\{2^{25.0\omega}, 2^{66.8}\}$	$2^{66.8}$	Sect. 5.3
	Tip4'	3	$\max\{2^{47.1\omega}, 2^{104.1}\}$	$2^{111.5}$	Sect. 3.2
			$\max\{2^{8+25.8\omega}, 2^{74.8}\}$	$2^{74.8}$	Sect. 6.1
SFS Collision	Tip5	3	$\max\{2^{17.4\omega}, 2^{41.1}\}$ (practical)	$2^{41.2}$	Sect. 5.4
	Tip4		$\max\{2^{13.7\omega}, 2^{33.2}\}$ (practical)	$2^{32.5}$	
	Tip4'		$\max\{2^{13.7\omega}, 2^{33.2}\}$ (practical)	$2^{32.5}$	
	Tip4	4	$\max\{2^{8+46.6\omega}, 2^{101.1}\}$	$2^{118.5}$	Sect. 5.5
Collision	Monolith-64	2	$\max\{2^{17.0\omega}, 2^{37.8}\}$ (practical)	$2^{40.3}$	Sect. 3.2
			2^{64}	2^{64}	Sect. 7.2
	Monolith-31	2	$\max\{2^{41.6\omega}, 2^{76}\}$	$2^{98.6}$	Sect. 3.2
			$\max\{2^{14.6\omega}, 2^{27}\}$ (practical)	$2^{34.6}$	Sect. 7.2
SFS Collision	Monolith-64	3	$\max\{2^{49.2\omega}, 2^{88}\}$	$2^{116.7}$	Sect. 7.3

Our results are summarized in Table 1. As far as we know, they are the most successful attacks against these ciphers. In particular, Tip5's designers claim that 3 rounds are sufficient for guaranteeing security (see [SLS+23, Table 4]), and that "*the round count $N = 5$ was set to provide a roughly 50% security margin*" (cited from [SLS+23, Sect. 5.11]). Similar considerations hold for Tip4, and Tip4'. Since the Tip5 family is instantiated by 5 rounds only, *our attacks have significantly reduced the security margins of the Tip5 family.*

2 Preliminaries

2.1 Notation

Throughout this paper, p is a prime number, and $(\mathbb{F}_p, +, \cdot)$ is the Galois field with the set of integers modulo p.[1] In addition, we use the following notation.

- For a matrix M of size $n \times n$ and a vector $x = (x_0, \ldots, x_{n-1})^T$, $(M(x))_i$ represents the i-th entry of $M \times x$ where $0 \leq i < n$;
- $\mathtt{R} = 2^{64} \bmod 2^{64} - 2^{32} + 1$ and $\mathtt{R}^{-1} = (2^{64})^{-1} \bmod 2^{64} - 2^{32} + 1$ are constants;
- $\mathrm{Diag}_{\mathtt{R}^{-1},n,s}$ denotes a diagonal matrix $\mathrm{Diag}(\underbrace{\mathtt{R}^{-1}, \ldots, \mathtt{R}^{-1}}_{s}, \underbrace{1, \ldots, 1}_{n-s})$, and $\mathrm{Diag}_{\mathtt{R},n,s}$ denotes a diagonal matrix $\mathrm{Diag}(\underbrace{\mathtt{R}, \ldots, \mathtt{R}}_{s}, \underbrace{1, \ldots, 1}_{n-s})$;

[1]We emphasize that, in some cases, we will also consider a different modulo for these arithmetic operations. In such cases, we will make it clear.

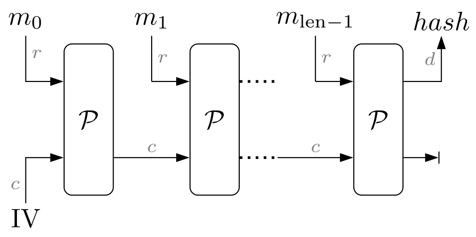


Figure 1: Sponge construction used for the Tip5 family.

- $\neg x$ denotes the logical complement of the binary string x;
- $\lll$ denotes the circular left shift operation for a binary string.

2.2 Sponge Construction

The sponge construction [BDPA08] can be used to model or implement many cryptographic primitives, including cryptographic hashes. It works on an internal state of n field elements, which is divided into two parts: the outer part of r field elements (r is called the rate), and the inner part of c field elements ($c = n - r$ is called the capacity). The sponge construction used for the Tip5 family is illustrated in Figure 1, which is instantiated with a permutation $\mathcal{P} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ and a state of n field elements.

In the absorbing phase, r field elements are read from the input and overwrite the first r elements of the state (or added to the first r elements), interleaved with applications of the permutation $\mathcal{P}$. In the squeezing phase, the first d field elements of the state are returned as output blocks.

The sponge constructions used in Tip5 family and Monolith family The hash function of Tip5 family comes in two modes of operation, depending on whether the input is fixed-length or variable-length. When the input is fixed-length, i.e., the length is always exactly r, no padding is required, and there is only one absorption where IV is initialized with all 1. When it is variable-length, it is padded by appending a 1 followed by the minimal number of 0's necessary to make the padded input length a multiple of r, and IV is initialized to all 0. For Monolith, it uses the SAFE framework [KBM23], and we refer the details to [KBM23] for the padding rule.

To avoid considering the details of padding, in our attacks on Monolith family, we only consider the collision in the inner part after the permutation. Once an inner collision is obtained, a full-state collision can be trivially generated and a collision in the hash value is found, which is a common method to find collisions for the sponge construction. Such a strategy will also be used in the attaks on Tip5 family.

Security claims for Tip5 and Monolith families. Preimage and collision attacks are well-known attacks on a hash function. For both Tip5 and Monolith family, the designers claim 128 bits of security for these 2 attacks (with only the exception of Monolith-31 that targets 124-bit security). Although the resistance of SFS collision attack is not claimed, finding a SFS collision should be as hard as finding a collision for a secure hash function. Note that in the SFS collision attack, the attacker can even control the value of IV, but the difference of IV has to be 0.

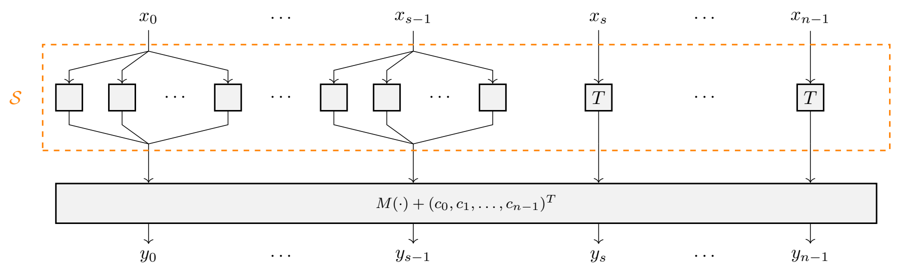


Figure 2: One round of the permutations of Tip5, Tip4 and Tip4', where $x_i, y_i \in \mathbb{F}_p$.

2.3 Permutations of the Tip5 Family

We consider an equivalent description of the permutations of Tip5 [SLS+23], Tip4 [Sal23] and Tip4' [Sal23] in this part. They are SPN-based permutations defined over $\mathbb{F}_p^n$, where $p = 2^{64} - 2^{32} + 1$. As illustrated in Figure 2, the i-th round function is defined as

$$\mathcal{R}^{(i)} : \mathbb{F}_p^n \to \mathbb{F}_p^n, \quad x \mapsto c^{(i)} + M_i(\mathcal{S}(x))$$

for $1 \le i \le N$, where

- $c^{(i)} = (c_0^{(i)}, \ldots, c_{n-1}^{(i)}) \in \mathbb{F}_p^n$ denotes the i-th round constant;
- $M_i = \mathrm{Diag}_{\mathtt{R},n,s} \times M_o \times \mathrm{Diag}_{\mathtt{R}^{-1},n,s}$ for $1 \le i \le N-1$ and $M_N = M_o \times \mathrm{Diag}_{\mathtt{R}^{-1},n,s}$, where $M_o \in \mathbb{F}_p^{n \times n}$ is a circulant MDS matrix and where $M_i(x) := M_i \times x$. For convenience, we simply let $M = M_i$ for $1 \le i \le N-1$ and $M_f = M_N$;
- $\mathcal{S} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ is an invertible nonlinear layer.

In particular, the non-linear layer[2] $y = (y_0, \ldots, y_{n-1}) = \mathcal{S}(x_0, \ldots, x_{n-1}) \in \mathbb{F}_p^n$ is defined as

$$y_i = \begin{cases} S(x_i) & \text{if } 0 \le i < s\,, \\ T(x_i) = x_i^7 & \text{if } s \le i < n\,. \end{cases}$$

where $S(\cdot)$ is a permutation over $\mathbb{F}_p$ whose details will be given in Section 4. For the concrete definition of the original MDS matrix M_o and the fixed round constants, we refer to [SLS+23] for Tip5, and to [Sal23] for Tip4 and Tip4'. For convenience, we call the permutations $S(\cdot)$ and $T(\cdot)$ the S-box and T-box, respectively. In addition, $t = n - s$ denotes the number of T-boxes in each round.

Note that the N rounds of the permutation of the Tip5 family are $\mathcal{R}^{(N)} \circ \mathcal{R}^{(N-1)} \circ \cdots \circ \mathcal{R}^{(1)} \circ \mathrm{Diag}_{\mathtt{R},n,s}$, i.e., the input $\mathbb{F}_p^n$ will first be element-wise multiplied with the constant vector $(\underbrace{\mathtt{R}, \ldots, \mathtt{R}}_{s}, \underbrace{1, \ldots, 1}_{n-s})$, and then passed through N round functions. Further note that the diagonal matrix $\mathrm{Diag}_{\mathtt{R},n,s}$ has no diffusion effect on the input due to the element-wise multiplication.

Parameters. The concrete parameters for the sponge-based hash functions Tip5, Tip4 and Tip4' are detailed in Table 2.

[2]In the original description, the linear layer is simply M_o, and the first s state words $(x_0, \ldots, x_{s-1})$ pass through the $\mathcal{S}$ layer according to $y_i = \mathtt{R}^{-1} \cdot S(\mathtt{R} \cdot x_i)$.

Table 2: Parameters for Tip5, Tip4 and Tip4' used in sponge construction.

Parameter	Symbol	Tip5	Tip4	Tip4'
Field modulus	p	$2^{64}-2^{32}+1$	$2^{64}-2^{32}+1$	$2^{64}-2^{32}+1$
Number of rounds	N	5	5	5
Claimed security (bits)	κ	128	128	128
State size	n	16	16	12
Sponge rate & capacity	(r,c)	(10,6)	(12,4)	(8, 4)
Digest length	d	5	4	4
# S-box	s	4	4	4
# T-box	t	12	12	8

2.4 Permutations of Monolith

Monolith [GKL+24] is a family of SPN-based permutations defined over $\mathbb{F}_p^n$ recently published at ToSC 2024/3. In particular, Monolith-64 and Monolith-31 correspond to $(p,n)=(2^{64}-2^{32}+1,12)$ and $(p,n)=(2^{31}-1,24)$. As shown in Figure 3, the i-th round function is defined as

$$\mathcal{R}^{(i)}:\mathbb{F}_p^n\to\mathbb{F}_p^n\,,\quad x\mapsto c^{(i)}+M(\mathcal{F}(\mathcal{B}(x)))$$

for $1\le i\le N$, where

- $c^{(i)}\in\mathbb{F}_p^n$ is a random round constant;
- $M\in\mathbb{F}_p^{n\times n}$ is defined via an MDS matrix (called "concrete" by the designers) such that $M(x):=M\times x$;
- $\mathcal{F}:\mathbb{F}_p^n\to\mathbb{F}_p^n$ (called "bars" by the designers) is a type-III Feistel construction instantiated via square maps

$$\mathcal{F}(x_0,x_1\ldots,x_{n-1}):=(x_0,x_1+x_0^2,x_2+x_1^2,\ldots,x_{n-1}+x_{n-2}^2)\,,$$

- $\mathcal{B}:\mathbb{F}_p^n\to\mathbb{F}_p^n$ is an invertible non-linear layer (called "bricks" by the designers).

Let us give more details of the nonlinear layer $\mathcal{B}$. Abusing notation, we also denote the permutation over $\mathbb{F}_p$ used in Monolith by $S(x)$, but it is not the same as that used in Tip5. In particular, the non-linear layer $y=\mathcal{B}(x)$ is defined as

$$y_i=\begin{cases}S(x_i) & \text{if } 0\le i<s\,,\\ x_i & \text{if } s\le i<n\,.\end{cases}$$

The details of $S(x)$ for Monolith-64 and Monolith-31 will be explained in Section 4. For the concrete MDS matrix M and the fixed round constants, we refer to [GKL+24] for more details.

Note that the N rounds of the permutation of the Monolith family is $\mathcal{R}^{(N)}\circ\mathcal{R}^{(N-1)}\circ\cdots\circ\mathcal{R}^{(1)}\circ M$, i.e., the input will pass through a linear layer defined by M and then passed through N round functions. Different from the Tip5 family, there is a mixing layer at the beginning of the Monolith permutation.

Parameters. The concrete parameters for the sponge-based hash functions Monolith-64 and Monolith-31 are detailed in Table 3. We should note that in both Monolith-64 and Monolith-31, there is $c=d=\frac{r}{2}=\frac{n}{3}$.

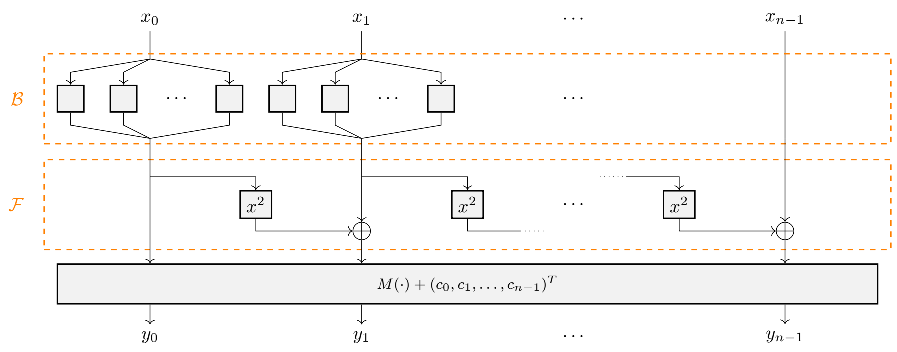


Figure 3: One round of the `Monolith` permutation, where $x_i, y_i \in \mathbb{F}_p$.

Table 3: Parameters for `Monolith` used in Sponge hashing mode.

Parameter	Symbol	`Monolith-64`	`Monolith-31`
Field modulus	p	$2^{64} - 2^{32} + 1$	$2^{31} - 1$
Number of rounds	N	6	6
Claimed security (bits)	κ	128	124
State size	n	12	24
Sponge Rate & capacity	(r, c)	(8,4)	(16,8)
Digest length	d	4	8
# S-Box	s	4	8

2.5 Gröbner Basis Attacks

The presented attacks involve solving a multivariate polynomial equation system over a finite prime field $\mathbb{F}_p$. We use the following notation: Let $\{F_1, \ldots, F_{n_e}\} \subset \mathbb{F}_p[x_1, \ldots, x_{n_v}]$ be a set of n_e polynomials in n_v variables. Let $\mathcal{I} := \langle F_1, \ldots, F_{n_e} \rangle$ denote the ideal generated by those polynomials, and let $d_{\mathcal{I}} := \dim_{\overline{\mathbb{F}}_p}(\mathbb{F}_p[x_1, \ldots, x_{n_v}]/\mathcal{I})$, called the *degree of the ideal*, denote the dimension of the quotient ring $\mathbb{F}_p[x_1, \ldots, x_{n_v}]/\mathcal{I}$ as $\overline{\mathbb{F}}_p$-vector space.

Multivariate system solving. A popular choice to find the common set of solutions to a system of n_e equations $F_1 = 0, \ldots, F_{n_e} = 0$ in n_v is to first compute a Gröbner basis [Buc76] of the ideal $\mathcal{I} = \langle f_1, \ldots, f_{n_e} \rangle$ with respect to the *lexicographic order*, which takes a triangular form [Bar04, §1.3]. This essentially reduces the problem of multivariate system solving to univariate root finding, as solutions can be recovered and extended iteratively by solving (a system of) univariate equation(s) [CLO15, §3.1]. A good summary of the procedure of univariate root finding over finite fields can be found in [BBLP22, §3.1].

In practice, directly computing the desired Gröbner basis is expensive. If the ideal $\mathcal{I}$ is zero-dimensional, that is, $d_{\mathcal{I}} < \infty$, the following approach has proven advantageous:

1. Compute the Gröbner basis of $\mathcal{I}$ with respect to the DRL (degree–reverse–lexicographic) order, using, for example, Faugère's F4 [Fau99] or F5 [Fau02] algorithm.

2. Apply the *FGLM* basis conversion algorithm [FGLM93] to convert it into a Gröbner basis with respect to the LEX (lexicographic) order.

Complexity. In the following, we assume that the system is well-defined ($n_e = n_v$), and that $\mathcal{I}$ is zero-dimensional.

1. A DRL Gröbner basis of $\mathcal{I}$ using the F4 or F5 algorithm can be computed in

$$\mathcal{O}\left(\binom{n_v + d_{\text{reg}}}{n_v}^{\omega}\right)$$

arithmetic operations over $\mathbb{F}_p$ [FBS04], where ω denotes the linear algebra constant and d_{reg} denotes the *degree of regularity*, as defined in [BSGL20, §A 2.2.1]. In general, calculating d_{reg} is as hard as computing the DRL Gröbner basis itself. Thus, tight bounds on d_{reg} are essential for deriving good complexity estimates.

For regular systems (cf. [BFS15]), the degree of regularity is given by the so-called *Macaulay bound*

$$d_{\text{MAC}} := 1 + \sum_{i=1}^{n_v} (\deg(F_i) - 1). \tag{1}$$

2. The basis conversion from a DRL to a LEX Gröbner basis using the original FGLM algorithm can be computed in $\mathcal{O}(n_v \cdot d_{\mathcal{I}}^3)$ arithmetic operations over $\mathbb{F}_p$ [FGLM93]. There exist probabilistic variants with a sub-cubic complexity [FGHR14]:

$$\mathcal{O}(n_v \cdot d_{\mathcal{I}}^{\omega}),$$

where ω denotes the linear algebra constant.

An upper bound for the degree $d_{\mathcal{I}}$ of the zero-dimensional ideal $\mathcal{I}$, which coincides with the number of solutions (over the algebraic closure, counted with multiplicities) to the polynomial equation system, is given by the Bézout bound:

$$B := \prod_{i=1}^{n_v} \deg(F_i). \tag{2}$$

In particular, if the system is regular, we have $d_{\mathcal{I}} = B$.

3. Given a univariate polynomial $F \in \mathbb{F}_p[x]$ of degree $d := \deg(F)$, its roots in $\mathbb{F}_p$ can be computed in $\mathcal{O}(d \log(d) \cdot (\log(d) + \log(p)) \cdot \log(\log(d)))$ finite field operations [BBLP22, §3.1]. In general, if F is the unique univariate polynomial of the LEX Gröbner basis, we have $\deg(F) \leq d_{\mathcal{I}}$. Thus, in practice, computing the solutions to a LEX Gröbner basis is mostly negligible in comparison to the previous steps, given that the system is not already in LEX order.

The linear algebra constant $\omega \geq 2$ accounts for the complexity of multiplying two dense matrices. From a designer's perspective, it is common to use the smallest possible value of $\omega = 2$ in the security assessment to account for future developments and add some additional security. However, we mention that in some cases, $\omega = 2$ also holds in practice [BFP09]. From an attacker's perspective, estimating the accurate value of ω in F4/F5 algorithms is not easy, though the most conservative value $\omega = 3$ can be used. Nevertheless, in practice, ω is always smaller than 3 for F4/F5 algorithms as the constructed Macaulay matrix is sparse, and Gaussian elimination is only performed on a submatrix of the Macaulay matrix. Indeed, it is common practice to use $\omega = 2.8$ or $\omega = 2.37$ to claim a successful Gröbner basis attack. According to our experiments, using $\omega = 2.37$ does not seem to underestimate the complexity. Hence, we choose $\omega = 2.37$ for F4 and F5 algorithms throughout this paper. To have a safe claim, we use $\omega = 3$ for the FGLM algorithm as it is not optimized in Magma, though we have to emphasize that ω might be smaller than 3 if optimized versions of the FGLM algorithms, e.g. [FGHR14], are used.

As the hidden factors in the asymptotic $\mathcal{O}(\cdot)$ notation are small (see, e.g., [ACG+19]), it is common to directly use $\binom{n_v+d_{\mathrm{MAC}}}{n_v}^{\omega}$ to estimate the time complexity. Hence, for a system of n_v polynomials in n_v variables generating a zero-dimensional ideal $\mathcal{I}$, we estimate the overall complexity of computing the set of common solutions in $\mathbb{F}_p$ via Gröbner basis as

$$\max\left\{\binom{n_v+d_{\mathrm{MAC}}}{n_v}^{\omega}, n_v\cdot B^3\right\}. \tag{3}$$

Experimental setup. All practical experiments are conducted on a machine with an Intel Xeon E5-2630 v3 @ 2.40GHz (using 8 cores) and 378GB RAM under Debian 11 using Magma V2.26-2. As previously observed [BSGL20, §A] [BBLP22], when using Magma, the basis conversion step (FGLM) is often the bottleneck of the Gröbner basis computation, as its implementation of the FGLM algorithm seems not to take advantage of fast linear algebra techniques.

3 Cryptanalysis of Tip5 and Monolith Independent of the S-box

In all Tip5 and Monolith families, there exists a special large S-box $S(\cdot)$ over $\mathbb{F}_p$ for which we have not yet given the details. However, we should keep in mind that it has a very complex high-degree polynomial representation over $\mathbb{F}_p$.

Such a special S-box $S(\cdot)$ was first used in Reinforced Concrete in order to prevent algebraic attacks like the Gröbner basis attack. To analyze hash functions using such special S-boxes, the designers of Tip5 proposed to use the available degrees of freedom in order to impose the inputs of such S-boxes to be equal to some fixed known constants. This strategy allows them to construct low-degree equations by skipping the influence of such S-boxes. In this section, we revisit the proposed strategy and show how it can be applied to straightforward collision attacks independent of $S(\cdot)$.

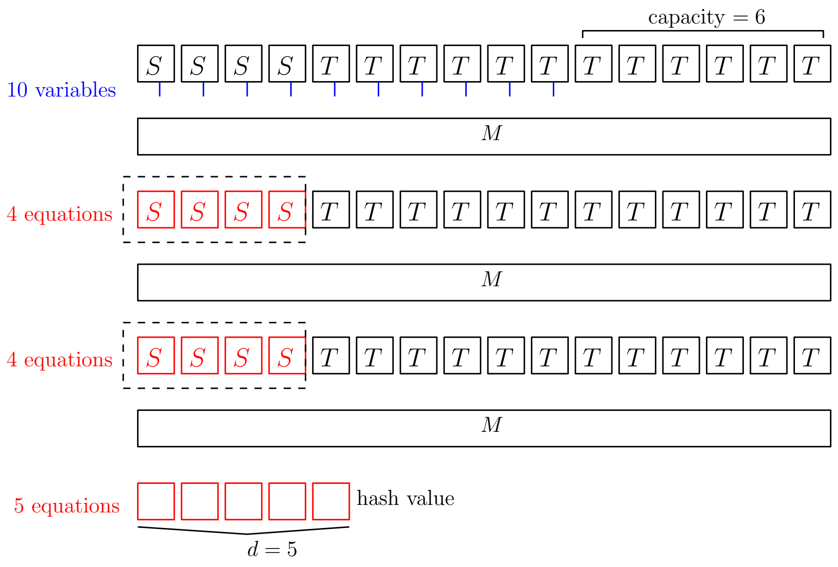


Figure 4: The preimage attack on 3-round Tip5 independent of the S-box

3.1 Preimage Attacks Independent of the S-box

Let us briefly describe the designers' analysis of Tip5 against the preimage attack. As shown in Figure 4, since there is no diffusion layer at the beginning, $r = 10$ variables can be introduced to denote the output words of all $s = 4$ S-boxes and the first $t - c = 6$ T-boxes at the 1st round. The remaining $c = 6$ output words are constants since they are in the inner part.

Then, randomly fix the 8 inputs of all the S-boxes at the 2nd and 3rd rounds, and compute their corresponding outputs by evaluating the S-box. Hence, the inputs of the 3rd round can be written as degree-7 polynomials in these 10 variables, and the hash value can be expressed as degree-49 polynomials in these 10 variables. However, these are based on the guess of the inputs of the S-boxes at the 2nd round and 3rd round. Therefore, we have the following equations in the 10 variables:

- $s = 4$ linear equations to ensure the guess of the inputs of the 4 S-boxes at the 2nd round;
- $s = 4$ degree-7 equations to ensure the guess of the inputs of the 4 S-boxes at the 3rd round;
- $d = 5$ degree-49 equations to match the hash value.

Therefore, to match the hash value with the above guess-and-determine method to ignore the details of $S(\cdot)$, it is required to set up 13 equations in 10 variables, which has a solution with an extremely low probability $p^{-3} \approx 2^{-192}$, letting alone the additional cost to solve such polynomial equations with Gröbner basis. Hence, such a guess-and-determine preimage attack cannot reach 3 rounds of Tip5 (equivalently, this attack can only reach 2 rounds of Tip5).

We recall that, without guessing the inputs of the S-boxes, the inputs of the 3rd round will be complex polynomials of extremely high degree in these 10 variables, and we even do not know what these polynomials are. Hence, the corresponding polynomial equations over $\mathbb{F}_p$ to match the hash value in these 10 variables are unknown, which somehow prevents the Gröbner basis attack.

The same analysis can be applied to Monolith, where we only have $r = 2s$ free variables representing the outer part of size r, and need to set the $2s$ inputs to the s S-boxes at the 1st and 2nd rounds as fixed values in order to express the hash value as low-degree polynomials in these $2s$ free variables. Hence, we have $2s + d = 3s$ equations in only $2s$ variables. This implies that the attack cannot reach beyond 2 rounds of Monolith.

While the above conclusion that the preimage attack cannot reach 3-round Tip5 and 2-round Monolith looks sound, this may not be the case for the collision attack since the target becomes to find two distinct inputs that collide in the hash value, rather than a single input that leads to a given hash value. In the following, we describe how to apply a similar idea to mount collision attacks on 3-round Tip5 and 2-round Monolith.

3.2 Collision Attacks Independent of the S-box

In the collision attack, our goal is to find two distinct inputs that collide in the hash value.

On Tip5 family. For the Tip5 family, we can introduce $2r$ variables to denote the output pairs of all s S-boxes and $t - c$ ($=r - s$) T-boxes at the 1st round. Then, we randomly fix the input pairs for the s S-boxes at the 2nd round and force the input difference of the s S-boxes at the 3rd round to be 0. In this way, the difference in the hash value can be expressed as d polynomials in these $2r$ variables of degree 49. Similarly, we have in total $2s + s + d = 3s + d$ equations in these $2r$ variables:

1. $2s$ linear equations in these $2r$ variables to ensure the fixed input pairs of the s S-boxes at the 2nd round;
2. s degree-7 equations in these $2r$ variables to ensure the input differences of the s S-boxes at the 3rd round;
3. d degree-49 equations in these $2r$ variables to ensure the difference in the hash value is zero.

By performing Gaussian elimination on the $2s$ linear equations, we only need to solve s degree-7 and d degree-49 equations in $2r - 2s$ variables. If $2r - 2s \geq s + d$, we can guess $2r - 2s - (s + d) = 2r - 3s - d$ variables and solve $s + d$ nonlinear equations in $s + d$ variables, and the complexity for the FGLM algorithm is $(s + d) \cdot 7^{3s} \cdot 49^{3d}$.

In Tip5, we have $(s, d, r) = (4, 5, 10)$, and hence we need to solve 4 degree-7 and 5 degree-49 equations in 9 variables after guessing 3 variables. Under the regularity assumption, the time complexity to compute the Gröbner basis is estimated as $2^{54.3\omega}$ field operations. With $\omega = 2.37$, it is already larger than 2^{128}.

In Tip4, we have $(s, d, r) = (4, 4, 12)$, and therefore we need to solve 4 degree-7 and 4 degree-49 equations in 8 variables after guessing 8 variables. Under the regularity assumption, the time complexity to compute the Gröbner basis is estimated as $2^{47.1\omega}$, and it is $2^{111.5}$ with $\omega = 2.37$.

In Tip4', we have $(s, d, r) = (4, 4, 8)$, and thus we need to solve 4 degree-7 and 4 degree-49 equations in 8 variables without guessing any variables. This time complexity is the same as in the above attack on 3-round Tip4.

On Monolith family. The above strategy can be directly applied to the inner collision attack on 2-round Monolith. As there is a diffusion layer at the beginning of Monolith, we can only introduce $2r = 4s$ free variables to denote the input pairs of the 2-round permutation. Then, randomly fix the s input pairs of the s S-boxes at the 1st round, and force the input differences of all s S-boxes at the 2nd round to be 0. In this way, the difference of the last c state words after the 2-round permutation can be expressed as $c = s$ polynomials in these $4s$ variables. Similarly, we have $2s$ linear equations, s degree-2 equations and s degree-4 equations in $4s$ variables. After performing Gaussian elimination on the linear equations, we get s degree-2 equations and s degree-4 equations in $2s$ variables.

The theoretic time complexity for the FGLM algorithm is estimated as $2s \cdot 2^{3s} \cdot 4^{3s}$. Under the regularity assumption, the time complexity of the Gröbner basis attack is estimated as $\max\{2^{20.0\omega}, 2^{39}\}$ for 2-round Monolith-64, and $\max\{2^{41.6\omega}, 2^{76}\}$ for 2-round Monolith-31.

Experiments have confirmed that the collision attack on 2-round Monolith-64 is practical, as shown in Table 4. It is found that the system does not behave like a regular system (due to $d_{\text{reg}} < d_{\text{MAC}}$) and using $\omega = 2.37$ does not underestimate the complexity. Since $d_{\text{reg}} < d_{\text{MAC}}$, the system can be solved much faster in practice. A concrete colliding message pair is given in Table 8.

However, finding a practical collision for 2-round Monolith-31 with this method is still costly. Several novel techniques will be developed in this paper to solve this challenge.

Table 4: Experimental results for the collision attack on 2-round Monolith-64. The running time in the columns "T_{GB}" and "T_{FGLM}" is in seconds.

Target	T_{GB}	d_{MAC}	d_{reg}	$\mathcal{C}_{\text{GB}}$ ($\omega = 2.37$)	T_{FGLM}	B	$d_{\mathcal{I}}$	$\mathcal{C}_{\text{FGLM}}$ ($\omega = 3$)	Memory (MB)
Monolith-64	3688	17	12	$2^{17.0\omega}$ ($2^{40.3}$)	1866	2^{12}	3011	$2^{3+11.6\omega}$ ($2^{37.8}$)	2194.5

4 Properties of the S-box

As stated above, there is yet no cryptanalysis exploiting the details of the special S-boxes used in Tip5, Tip4, Tip4' and Monolith due to their complex and unknown polynomial representations over $\mathbb{F}_p$. We take this challenge in this work, and propose an efficient algorithm to determine whether a randomly given input-output difference pair is valid for these S-boxes, and to find all possible input pairs for this difference transition if it is valid.

4.1 General Description of the S-box

Let us consider a general form of this S-box, i.e., $z = S(w)$ where $w, z \in \mathbb{F}_p$ and $S : \mathbb{F}_p \mapsto \mathbb{F}_p$ is a permutation over $\mathbb{F}_p$. For convenience, we call $(x_0, \ldots, x_{h-1})$ a split of $x \in \mathbb{F}_p$ with parameters $(l_0, \ldots, l_{h-1})$ where $\sum_{i=0}^{h-1} l_i = \lceil \log_2 p \rceil$ and

$$x_i = \begin{cases} x \wedge (2^{l_0} - 1) & \text{if } i = 0 \\ (x \gg \sum_{j=0}^{i-1} l_j) \wedge (2^{l_i} - 1) & \text{otherwise } (1 \le i < h)\,, \end{cases}$$

and we denote this by

$$(x_0, \ldots, x_{h-1}) = \sigma_{l_0,\ldots,l_{h-1}}(x).$$

Equivalently, x can be re-written as

$$x = x_0 + 2^{l_0} \cdot x_1 + 2^{l_0+l_1} \cdot x_2 + \cdots + 2^{l_0+l_1+\cdots+l_{h-2}} \cdot x_{h-1} \bmod 2^{\lceil \log_2 p \rceil} \quad \text{where} \quad 0 \le x_i < 2^{l_i},$$

where

$$x = \sigma^{-1}_{l_0,\ldots,l_{h-1}}(x_0, \ldots, x_{h-1}).$$

If $(l_0, \ldots, l_{h-1})$ are clear from the context, we simply use

$$(x_0, \ldots, x_{h-1}) = \sigma(x),\ x = \sigma^{-1}(x_0, \ldots, x_{h-1}).$$

For $z = S(w)$, we first need to compute $(w_0, \ldots, w_{h-1}) = \sigma(w)$. Then, compute $(z_0, \ldots, z_{h-1})$ with

$$z_i = S_i(w_i),\ \text{for } 0 \le i < h,$$

where S_i is a permutation. Finally, compute $z = \sigma^{-1}(z_0, \ldots, z_{h-1})$. Note that S_i for $0 \le i < h$ are carefully constructed such that S is a permutation over $\mathbb{F}_p$. In other words, for $\forall w < p$, $z < p$ has to hold.

In Tip5, Tip4 and Tip4', we have

$$h = 8, \quad l_i = 8, \quad S_i(x) = L(x) = (x+1)^3 - 1 \bmod 2^8 + 1 \ \text{ for } \ 0 \le i < 8.$$

In Monolith-64, we have

$$h = 8, \quad l_i = 8, \quad S_i(x) = \chi_{011}(x) \ \text{ for } \ 0 \le i < 8,$$

where

$$\chi_{011}(x) = (x \oplus (((\neg x) \lll 1) \wedge (x \lll 2) \wedge (x \lll 3))) \lll 1$$

is computed over $\mathbb{F}_2^8$. In Monolith-31, we instead have

$$h = 4, \quad (l_0, l_1, l_2, l_3) = (8, 8, 8, 7), \quad S_0(x) = S_1(x) = S_2(x) = \chi_{011}(x), \quad S_3(x) = \chi_{01}(x),$$

where

$$\chi_{01}(x) = (x \oplus (((\neg x) \lll 1) \wedge (x \lll 2))) \lll 1,$$

is computed over $\mathbb{F}_2^7$.

For such a nonlinear permutation $z = S(w)$ over $\mathbb{F}_p$, we will present an efficient algorithm to solve the differential equation $\Delta z = S(w + \Delta w) - S(w)$ for any given $(\Delta w, \Delta z)$. The main idea is to construct several small tables and use them to equivalently construct the whole differential distribution table (DDT) of $z = S(w)$, even though $p > 2^{30}$. Note that the naive algorithm will require time complexity and memory complexity of $\mathcal{O}(p^2)$ to construct this DDT, and hence it soon becomes impractical. The motivation to study this differential equation is to significantly improve the attacks on these ciphers, as will be clear in our later advanced attacks.

4.2 Efficient Algorithm to Solve $\Delta z = S(w + \Delta w) - S(w)$

Given $(\Delta w, \Delta z) \in \mathbb{F}_p^2$, how can we efficiently determine whether $\Delta w \to \Delta z$ is a valid difference transition through $S(\cdot)$? In other words, how to know whether there exists w such that $\Delta z = S(w + \Delta w) - S(w)$? Although S is built with h functions $S_0, \ldots, S_{h-1}$, it does not mean that we can simply use the DDT of S_i to construct the DDT of S, and solve this problem.

Let us consider an input pair (w, w') and output pair (z, z'). Throughout this section, we also set $(w_0, \ldots, w_{n-1}) = \sigma(w)$, $(w'_0, \ldots, w'_{n-1}) = \sigma(w')$, $(z_0, \ldots, z_{n-1}) = \sigma(z)$, and $(z'_0, \ldots, z'_{n-1}) = \sigma(z')$. Let

$$(\Delta w_0, \ldots, \Delta w_{h-1}) = \sigma(\Delta w) = \sigma(w' - w),\ (\Delta z_0, \ldots, \Delta z_{h-1}) = \sigma(\Delta z) = \sigma(z' - z).$$

Even though we know Δw_i according to Δw, we do not know the input difference between (w'_i, w_i) to each S_i. The same also applies to the difference between (z'_i, z_i). This is indeed due to the influence of the carry and the modular addition operation (i.e., mod p) when computing $w' = w + \Delta w$. Here we study this problem. (We limit ourselves to point out that since $S(w)$ is a permutation over $\mathbb{F}_p$, we will only consider the case $\Delta w \neq 0$ and $\Delta z \neq 0$.)

Removing the influence of modular addition. When $\Delta w \neq 0$, there are 2 possible cases, i.e., $w > w'$ or $w < w'$:

Case 1: If $w > w'$, we have $\Delta w = w' - w + p$ by definition. Therefore, $w = w' + (p - \Delta w)$ and $w' < \Delta w$. Hence, when computing $w' + (p - \Delta w)$ over the ring of integers, the result must be the same as within modulo p, i.e., no modular addition is required when computing $w = w' + (p - \Delta w)$.

In this case, let

$$\widetilde{\Delta w} = p - \Delta w,\ (\widetilde{\Delta w'_0}, \ldots, \widetilde{\Delta w'_{h-1}}) = \sigma(\widetilde{\Delta w}).$$

Then, we have

$$w_i = \text{carry}_i + \widetilde{\Delta w_i} + w'_i \bmod 2^{l_i},\ \text{for } 0 \leq i < h,$$

where

$$\text{carry}_{i+1} = \begin{cases} 1 & \text{if } i \geq 0 \text{ and } \text{carry}_i + \widetilde{\Delta w_i} + w'_i \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}.$$

Obviously, $\text{carry}_h = 0$ if and only if $w > w'$.

Case 2: If $w < w'$, we have $\Delta w = w' - w$. Similarly, when computing $w' = w + \Delta w$, no modular addition is required. In this case, we have

$$w'_i = \text{carry}_i + \Delta w_i + w_i \bmod 2^{l_i},\ \text{for } 0 \leq i < h,$$

where

$$\mathrm{carry}_{i+1} = \begin{cases} 1 & \text{if } i \geq 0 \text{ and } \mathrm{carry}_i + \Delta w_i + w_i \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}.$$

Similarly, $\mathrm{carry}_h = 0$ if and only if $w < w'$.

Similar analysis also applies to $(z, z', \Delta z)$.

For convenience, we can pre-compute h 3-dimensional tables $\mathrm{Tab}_0, \ldots, \mathrm{Tab}_{h-1}$ such that

$$\mathrm{Tab}_i[j_0][j_1][j_2] \quad = \quad \begin{cases} 1 & \text{if } j_0 + j_1 + j_2 \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}.$$

Constructing small tables to capture the carry. With Tab_i at hand, it is convenient to study the difference transition $\Delta w \to \Delta z$ through S. Specifically, given a random pair $(\Delta w, \Delta z)$, we consider in total 4 cases:

Case 0: $w > w'$ and $z > z'$;

Case 1: $w > w'$ and $z < z'$;

Case 2: $w < w'$ and $z > z'$;

Case 3: $w < w'$ and $z < z'$.

First, we compute $4h$ different 3-dimensional DDTs denoted by $\mathrm{DDT}_{i,j}$ where $0 \leq i < h$ and $0 \leq j \leq 3$. How to compute $\mathrm{DDT}_{i,j}$ is specified in Algorithm 1. Note that each cell of these 3-dimensional DDTs is a set of tuples reminiscent of right input pairs for a valid difference transition.

Specifically, if $w > w'$, we should consider the calculation $w = w' + (p - \Delta w) = w' + \widetilde{\Delta w}$. Otherwise, we consider the calculation $w' = w + \Delta w$. Similarly, this applies to z and z'. In this way, the influence of the modular addition operation (i.e., mod p) can be removed. These correspond to the underlying ideas in Line 13, Line 16, Line 23 and Line 26 of Algorithm 1, respectively. Indeed, the main idea of Algorithm 1 is very simple, and can be referred to Figure 5.

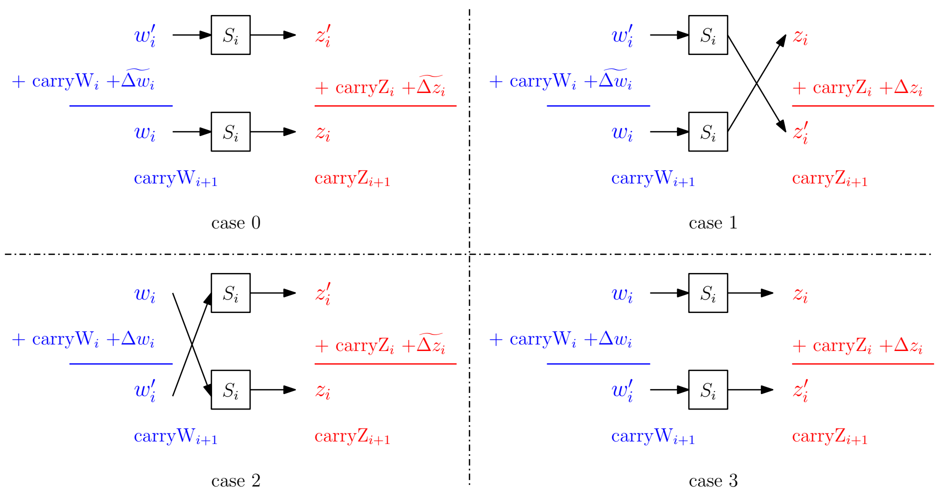
Roughly speaking, $\mathrm{DDT}_{i,j}$ is used to store the right input pairs as well as the corresponding information of carry for the difference transitions $\widetilde{\Delta w_i} \to \widetilde{\Delta z_i}$ if $j = 0$, $\widetilde{\Delta w_i} \to \Delta z_i$ if $j = 1$, $\Delta w_i \to \widetilde{\Delta z_i}$ if $j = 2$, and $\Delta w_i \to \Delta z_i$ if $j = 3$, through the small box $S_i(\cdot)$.

Specifically, let us elaborate more on $\mathrm{DDT}_{i,0}$, $\mathrm{DDT}_{i,1}$, $\mathrm{DDT}_{i,2}$ and $\mathrm{DDT}_{i,3}$.

- On $\mathrm{DDT}_{i,0}$: This corresponds to the case $w > w'$ and $z > z'$. Specifically, we have

$$w = w' + \widetilde{\Delta w}, \; z = z' + \widetilde{\Delta z}.$$

Hence, by the construction of $\mathrm{DDT}_{i,0}$, for each element $(\mathrm{carry}_{i+1}, w_i, w_i')$ stored in



w_i'
S_i
z_i'
$+$ carryW$_i$ $+\widetilde{\Delta w_i}$
$+$ carryZ$_i$ $+\widetilde{\Delta z_i}$
w_i
z_i
carryW$_{i+1}$
carryZ$_{i+1}$
case 0
case 1
case 2
case 3
$+$ carryW$_i$ $+\Delta w_i$
$+$ carryZ$_i$ $+\Delta z_i$

Figure 5: The underlying idea of Algorithm 1. Specially, once $w_i, w_i', \text{carryW}_i, \text{carryZ}_i$ are known, it is always feasible to compute $z_i, z_i', \text{carryW}_{i+1}, \text{carryZ}_{i+1}$, and the corresponding input-output differences according to the case j.

DDT$_{i,0}$[carry$_i$][$\widetilde{\Delta w_i}$][$\widetilde{\Delta z_i}$][j_3], we have

$$\begin{aligned}
\text{carryW}_i &= \text{carry}_i/2,\\
\text{carryZ}_i &= \text{carry}_i \bmod 2,\\
w_i &= \text{carryW}_i + \widetilde{\Delta w_i} + w_i' \bmod 2^{l_i},\\
z_i &= S_i(w_i),\\
z_i' &= S_i(w_i'),\\
z_i &= \text{carryZ}_i + \widetilde{\Delta z_i} + z_i' \bmod 2^{l_i},\\
\text{carryW}_{i+1} &= \begin{cases} 1 & \text{if } i \geq 0 \text{ and carryW}_i + \widetilde{\Delta w_i} + w_i' \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases},\\
\text{carryZ}_{i+1} &= \begin{cases} 1 & \text{if } i \geq 0 \text{ and carryZ}_i + \widetilde{\Delta z_i} + z_i' \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases},\\
\text{carry}_{i+1} &= 2 \times \text{carryW}_{i+1} + \text{carryZ}_{i+1}.
\end{aligned}$$

In particular, $w > w'$ and $z > z'$ hold if and only if carry$_h = 0$, i.e., carryW$_h = 0$ and carryZ$_h = 0$.

- On DDT$_{i,1}$: This corresponds to the case $w > w'$ and $z < z'$. Hence, we have

$$w = w' + \widetilde{\Delta w},\ z' = z + \Delta z.$$

For each element (carry$_{i+1}, w_i, w_i'$) stored in DDT$_{i,2}$[carry$_i$][$\widetilde{\Delta w_i}$][Δz_i][j_3], the relations in (z_i, z_i', carryZ$_i$, carryZ$_{i+1}$) are then updated as:

$$\begin{aligned}
z_i' &= \text{carryZ}_i + \Delta z_i + z_i \bmod 2^{l_i},\\
\text{carryZ}_{i+1} &= \begin{cases} 1 & \text{if } i \geq 0 \text{ and carryZ}_i + \Delta z_i + z_i \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}.
\end{aligned}$$

The remaining relations remain the same as in DDT$_{i,0}$. Similarly, $w > w'$ and $z < z'$ hold if and only if[3] carry$_h = 0$.

[3]By removing the influence of modular addition, we always consider the addition $a' = b' + c'$ over the

- On $\mathrm{DDT}_{i,2}$: This corresponds to the case $w < w'$ and $z > z'$. Hence, we have

$$w' = w + \Delta w,\ \ z = z' + \widetilde{\Delta z}.$$

For each element $(\mathrm{carry}_{i+1}, w_i, w_i')$ stored in $\mathrm{DDT}_{i,2}[\mathrm{carry}_i][\Delta w_i][\widetilde{\Delta z_i}][j_3]$, the relations in $(w_i, w_i', \mathrm{carryW}_i, \mathrm{carryW}_{i+1})$ are update as:

$$\begin{array}{rcl} w_i' & = & \mathrm{carryW}_i + \Delta w_i + w_i \bmod 2^{l_i}, \\ \mathrm{carryW}_{i+1} & = & \begin{cases} 1 & \text{if } i \geq 0 \text{ and } \mathrm{carryW}_i + \Delta w_i + w_i \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}. \end{array}$$

The remaining relations remain the same as in $\mathrm{DDT}_{i,0}$. Similarly, $w < w'$ and $z > z'$ hold if and only if $\mathrm{carry}_h = 0$.

- On $\mathrm{DDT}_{i,3}$: This corresponds to the case $w < w'$ and $z < z'$. Hence, we have

$$w' = w + \Delta w,\ \ z' = z + \Delta z.$$

For each element $(\mathrm{carry}_{i+1}, w_i, w_i')$ stored in $\mathrm{DDT}_{i,3}[\mathrm{carry}_i][\Delta w_i][\Delta z_i][j_3]$, the relations in $(w_i, w_i', \mathrm{carryW}_i, \mathrm{carryW}_{i+1})$ and $(z_i, z_i', \mathrm{carryZ}_i, \mathrm{carryZ}_{i+1})$ are updated as:

$$\begin{array}{rcl} w_i' & = & \mathrm{carryW}_i + \Delta w_i + w_i \bmod 2^{l_i}, \\ \mathrm{carryW}_{i+1} & = & \begin{cases} 1 & \text{if } i \geq 0 \text{ and } \mathrm{carryW}_i + \Delta w_i + w_i \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}, \\ z_i' & = & \mathrm{carryZ}_i + \Delta z_i + z_i \bmod 2^{l_i}, \\ \mathrm{carryZ}_{i+1} & = & \begin{cases} 1 & \text{if } i \geq 0 \text{ and } \mathrm{carryZ}_i + \Delta z_i + z_i \geq 2^{l_i} \\ 0 & \text{otherwise} \end{cases}. \end{array}$$

The remaining relations remain the same as in $\mathrm{DDT}_{i,0}$. Similarly, $w < w'$ and $z < z'$ hold if and only if $\mathrm{carry}_h = 0$.

With these $4h$ tables $\mathrm{DDT}_{i,j}$ at hand, for a randomly given pair $(\Delta w, \Delta z) \in \mathbb{F}_p^2$, we can use Algorithm 2 to find all $w \in \mathbb{F}_p$ satisfying $S(\Delta w + w) - S(w) = \Delta z$. The illustration of the procedure of Algorithm 2 can be referred to Figure 6.

4.3 Explanation of Algorithm 2

Given the input-output difference pair $(\Delta w, \Delta z) \in \mathbb{F}_p^2$ of $z = S(w)$, it is trivial to deal with the case $\Delta w = 0$ or $\Delta z = 0$, and hence we omit these cases in Algorithm 2. Instead, we only focus on the case $\Delta w \neq 0$ and $\Delta z \neq 0$. Our algorithm can also be easily modified to determine whether a randomly given $(\Delta w, \Delta z)$ is valid. Specifically, once we find one w satisfying $S(w + \Delta w) - S(w) = \Delta z$ with Algorithm 2, start backtracking and exit.

On Line 10 – Line 13 of Algorithm 2. With the current Algorithm 2, the goal is to find and store all possible w satisfying $S(w + \Delta w) - S(w) = \Delta z$. As it is unknown in advance whether such a w will cause $w > w'$ or $w < w'$ or $z = S(w) > z' = S(w + \Delta w)$ or $z = S(w) < z' = S(w + \Delta w)$, we traverse all 4 possible cases.

ring of integers where $b' > 0$, $a' > c'$ and $a', b', c' \in \mathbb{F}_p$. In this case, if there is a nonzero carry from the most significant bit when computing $b' + c'$, $a' > p$ and a' cannot be in $\mathbb{F}_p$. Hence, this carry must be 0. Similar analysis also applies to other cases.

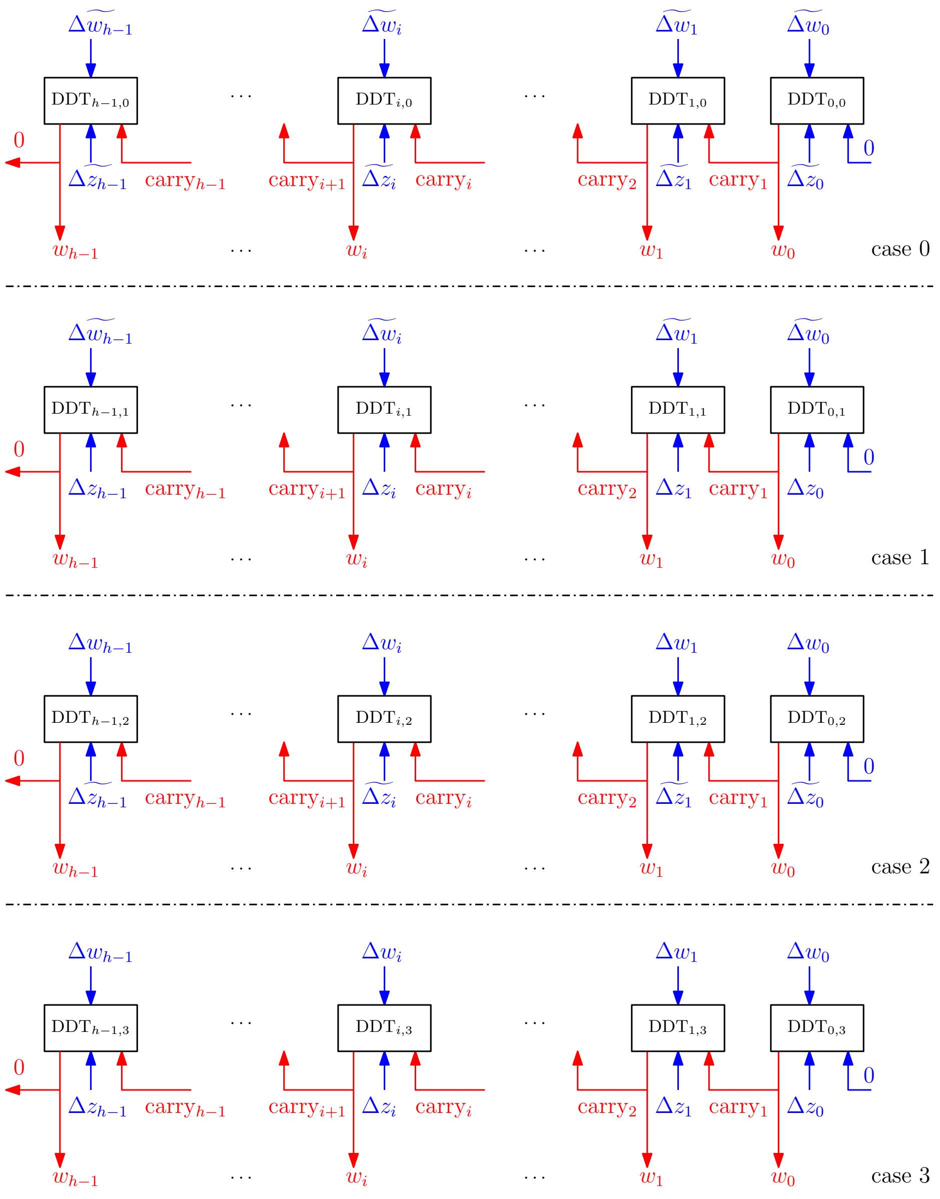


Figure 6: Description of the procedure of Algorithm 2, i.e., the procedure to retrieve the right input $w = \sigma^{-1}(w_{h-1}, \ldots, w_1, w_0)$.

Algorithm 1 Constructing the 3-dimensional table $\mathrm{DDT}_{i,j}$

```
 1: procedure CONSTRUCT-DDT($i, j, l_i, \mathrm{Tab}_i, \mathrm{DDT}_{i,j}$)
 2:     TUPLE.id = 0
 3:     TUPLE.x = 0
 4:     TUPLE.x′ = 0
 5:     for id = 0 to 3 do
 6:         for $u = 0$ to $2^{l_i} - 1$ do
 7:             for $u' = 0$ to $2^{l_i} - 1$ do
 8:                 $\mathrm{cx}_1 = \mathrm{id}/2$ // $\mathrm{id} = 2 \times \mathrm{cx}_1 + \mathrm{cy}_1$
 9:                 $\mathrm{cy}_1 = \mathrm{id} \bmod 2$
10:                 $x = u$
11:                 $x' = u'$
12:                 if $j = 0$ or $j = 1$ then
13:                     $\Delta x = x - (\mathrm{cx}_1 + x') \bmod 2^{l_i}$ //$x = \mathrm{cx}_1 + \Delta x + x' \bmod 2^{l_i}$
14:                     $\mathrm{cx}_2 = \mathrm{Tab}_i[\mathrm{cx}_1][\Delta x][x']$
15:                 else if $j = 2$ or $j = 3$ then
16:                     $\Delta x = x' - (\mathrm{cx}_1 + x) \bmod 2^{l_i}$ //$x' = \mathrm{cx}_1 + \Delta x + x \bmod 2^{l_i}$
17:                     $\mathrm{cx}_2 = \mathrm{Tab}_i[\mathrm{cx}_1][\Delta x][x]$
18:                 TUPLE.x = $x$
19:                 TUPLE.x′ = $x'$
20:                 $y = S_i(x)$
21:                 $y' = S_i(x')$
22:                 if $j = 0$ or $j = 2$ then
23:                     $\Delta y = y - (\mathrm{cy}_1 + y') \bmod 2^{l_i}$ //$y = \mathrm{cy}_1 + \Delta y + y' \bmod 2^{l_i}$
24:                     $\mathrm{cy}_2 = \mathrm{Tab}_i[\mathrm{cy}_1][\Delta y][y']$
25:                 else if $j = 1$ or $j = 3$ then
26:                     $\Delta y = y' - (\mathrm{cy}_1 + y) \bmod 2^{l_i}$ //$y' = \mathrm{cy}_1 + \Delta y + y \bmod 2^{l_i}$
27:                     $\mathrm{cy}_2 = \mathrm{Tab}_i[\mathrm{cy}_1][\Delta y][y]$
28:                 TUPLE.id = $2 \times \mathrm{cx}_2 + \mathrm{cy}_2$
29:                 $\mathrm{DDT}_{i,j}[\mathrm{id}][\Delta x][\Delta y]$.pushback(TUPLE)
```

On Line 22 — Line 27 of Algorithm 2 . For each of the 4 cases, based on the small tables $\mathrm{DDT}_{0,j}, \dots, \mathrm{DDT}_{h-1,j}$ where $0 \le j < 4$, we can efficiently retrieve possible candidate splits of w and w' with the depth-first search, which are stored in the arrays $wArr$ and $wArr'$, respectively.

On Line 17 — Line 20 of Algorithm 2. However, not each retrieved candidate is necessarily valid. Note that when constructing each $\mathrm{DDT}_{i,j}$, we consider all input-output pairs of $S_i(\cdot)$ from $\{0, \dots, 2^{l_i} - 1\}$. Hence, for a candidate $(wArr[0], \dots, wArr[h-1])$, it is possible to have

$$w = \sigma^{-1}(wArr[0], \dots, wArr[h-1]) \ge p,$$

which is an invalid input of the function $z = S(w)$. Similarly, it is also possible to get

$$w' = \sigma^{-1}(wArr'[0], \dots, wArr'[h-1]) \ge p.$$

Once $w < p$ and $w' < p$ both hold, by the construction of S, i.e., S is a permutation over $\mathbb{F}_p$, there must be $z = S(w) < p$ and $z' = S(w') < p$. In addition, according to the analysis of each $\mathrm{DDT}_{i,j}$ specified above, the additional condition id = 0 is equivalent to testing whether the following conditions hold:

- $w > w'$ and $S(w) > S(w')$ when $j = 0$;

- $w > w'$ and $S(w) < S(w')$ when $j = 1$;
- $w < w'$ and $S(w) > S(w')$ when $j = 2$;
- $w < w'$ and $S(w) < S(w')$ when $j = 3$.

This additional condition is required because we search for a valid w based on the guess of the relation in $(w, w' = w + \Delta w)$ and $(z, z') = (S(w), S(w'))$, respectively. Once we get a candidate of (w, w'), it has to be consistent with the guessed relation.

On the correctness. The correctness of Algorithm 2 is ensured by the construction of $\mathrm{DDT}_{i,j}$, whose properties have been explained in the above part. Specifically, we retrieve w_i from $\mathrm{DDT}_{i,j}$ in the order of $i = 0, \ldots, h-1$, according to either $(\widetilde{\Delta w_i}, \widetilde{\Delta z_i})$, or $(\widetilde{\Delta w_i}, \Delta z_i)$, or $(\Delta w_i, \widetilde{\Delta z_i})$ or $(\Delta w_i, \Delta z_i)$, as well as the carry. By the properties of $\mathrm{DDT}_{i,j}$, we obtain a possible candidate of $(w_0, \ldots, w_{h-1}) = \sigma(w)$ and the corresponding $(w'_0, \ldots, w'_{h-1}) = \sigma(w')$.

On the completeness. In particular, we do not miss any valid w satisfying $S(w + \Delta w) - S(w) = \Delta z$ since our algorithm can deal with all possible $(\Delta w, \Delta z)$, and all possible valid solutions of $(w_i, w'_i, \widetilde{\Delta w_i}, \widetilde{\Delta z_i})$, $(w_i, w'_i, \widetilde{\Delta w_i}, \Delta z_i)$, $(w_i, w'_i, \Delta w_i, \widetilde{\Delta z_i})$, and $(w_i, w'_i, \Delta w_i, \Delta z_i)$ have been taken into account when constructing each $\mathrm{DDT}_{i,j}$.

Remark 1. If the goal is to simply determine whether $(\Delta w, \Delta z)$ is a valid difference transition, one may observe that our algorithm may get stuck if the input-output differences of many small boxes are $(0, 0)$ because we need to enumerate all possible input pairs for these small boxes in the depth-first search. According to our observations on $\mathrm{DDT}_{i,j}[0][0][0]$ for the Tip5 and Monolith small boxes, we find that there are 2^{l_i} (i.e., the maximal possible number) tuples stored and each tuple (x, x', id) stored satisfies $\mathrm{id} = 0$. In addition, $\mathrm{DDT}_{i,j}[1][0][0]$ and $\mathrm{DDT}_{i,j}[2][0][0]$ are always empty. For $\mathrm{DDT}_{i,j}[3][0][0]$, there are at most 2 tuples stored. Hence, when reaching the case where $(\Delta w_i, \Delta z_i) = (0, 0)$, instead of directly running Line 22 − Line 27 in Algorithm 2, we can run the following code

```
**if** id = 0
    $wArr[i] = wArr'[i] = 0$
    RE($j, i+1, 0, h, wArr, wArr', \Delta w_0, \ldots, \Delta w_{h-1}, \Delta z_0, \ldots, \Delta z_{h-1}$, wSol)
**if** id = 3
    Line 22 − Line 27 of Algorithm 2
```

Moreover, before running the depth-first search in RE() (i.e., Line 16 - Line 27 in Algorithm 2), one can also pre-check[4] whether there exists i such that $\mathrm{DDT}_{i,j}[\mathrm{id}][\Delta w_i][\Delta z_i]$ is empty for $0 \leq \mathrm{id} \leq 3$. If so, the input pair is invalid and we do not need to run the search. This check takes at most $4h$ table lookups.

4.4 Application and Complexity Analysis

Our search algorithm is based on the depth-first search technique with early aborting, and only uses simple table look-ups. For such an algorithm, it is difficult to give an accurate theoretic estimation of the number of table look-ups, as well as the number of solutions for

[4]Indeed, after this procedure, we can get the set of id such that $\mathrm{DDT}_{i,j}[\mathrm{id}][\Delta w_i][\Delta z_i]$ is non-empty for each (i, j), and they can also be easily used to efficiently enumerate conforming input pairs or determine whether a difference transition is valid. If it is a valid difference transition, there will be sequences of id of length $h+1$, i.e., $\mathrm{id}_0 = 0, \mathrm{id}_1, \ldots, \mathrm{id}_{h-1}, \mathrm{id}_h = 0$, where $\mathrm{DDT}_{i,j}[\mathrm{id}_i][\Delta w_i][\Delta z_i]$ is non-empty and there exists j_3 such that $\mathrm{DDT}_{i,j}[\mathrm{id}_i][\Delta w_i][\Delta z_i][j_3].\mathrm{id} = \mathrm{id}_{i+1}$ for $0 \leq i \leq h-1$.

Algorithm 2 Finding w satisfying $S(w + \Delta w) - S(w) = \Delta z$

```
procedure FINDW($\Delta w, \Delta z$, wSol)
    $\widetilde{\Delta w} = p - \Delta w$
    $\widetilde{\Delta z} = p - \Delta z$
    $(\widetilde{\Delta w_0}, \ldots, \widetilde{\Delta w_{h-1}}) = \sigma(\widetilde{\Delta w})$
    $(\widetilde{\Delta z_0}, \ldots, \widetilde{\Delta z_{h-1}}) = \sigma(\widetilde{\Delta z})$
    $(\Delta w_0, \ldots, \Delta w_{h-1}) = \sigma(\Delta w)$
    $(\Delta z_0, \ldots, \Delta z_{h-1}) = \sigma(\Delta z)$
    $wArr = [\,]$
    $wArr' = [\,]$
    RE$(0, 0, 0, h, wArr, wArr', \widetilde{\Delta w_0}, \ldots, \widetilde{\Delta w_{h-1}}, \widetilde{\Delta z_0}, \ldots, \widetilde{\Delta z_{h-1}}$, wSol)
    RE$(1, 0, 0, h, wArr, wArr', \widetilde{\Delta w_0}, \ldots, \widetilde{\Delta w_{h-1}}, \Delta z_0, \ldots, \Delta z_{h-1}$, wSol)
    RE$(2, 0, 0, h, wArr, wArr', \Delta w_0, \ldots, \Delta w_{h-1}, \widetilde{\Delta z_0}, \ldots, \widetilde{\Delta z_{h-1}}$, wSol)
    RE$(3, 0, 0, h, wArr, wArr', \Delta w_0, \ldots, \Delta w_{h-1}, \Delta z_0, \ldots, \Delta z_{h-1}$, wSol)

procedure RE($j, i$, id, $h, wArr, wArr', \Delta w_0, \ldots, \Delta w_{h-1}, \Delta z_0, \ldots, \Delta z_{h-1}$, wSol)
    if $i = h$ then
        $w = \sigma^{-1}(wArr[0], \ldots, wArr[h-1])$
        $w' = \sigma^{-1}(wArr'[0], \ldots, wArr'[h-1])$
        if $w < p$ and $w' < p$ and id $= 0$ then
            wSol.pushback($w$)
    else if $i < h$ then
        size = DDT$_{i,j}$[id][$\Delta w_i$][$\Delta z_i$].size()
        for $j_3 = 0$ to size $- 1$ do
            nextId = DDT$_{i,j}$[id][$\Delta w_i$][$\Delta z_i$][$j_3$].id
            $wArr[i]$ = DDT$_{i,j}$[id][$\Delta w_i$][$\Delta z_i$][$j_3$].x
            $wArr'[i]$ = DDT$_{i,j}$[id][$\Delta w_i$][$\Delta z_i$][$j_3$].x'
            RE$(j, i+1$, nextId, $h, wArr, wArr', \Delta w_0, \ldots, \Delta w_{h-1}, \Delta z_0, \ldots, \Delta z_{h-1}$, wSol)
```

each different input $(\Delta w, \Delta z)$. This is because these values will vary for different $(\Delta w, \Delta z)$. Moreover, it is also important to know the probability that a randomly given $(\Delta w, \Delta z)$ is a valid input-output difference pair of $z = S(w)$.

To answer the above questions, we have implemented Algorithm 2 for Tip5, Monolith-64 and Monolith-31. We choose $N_{\text{test}} = 1\,000\,000$ random $(\Delta w, \Delta z)$ and compute

- N_{valid}: the number of valid $(\Delta w, \Delta z)$ such that $\exists w : \; S(w + \Delta w) - S(w) = \Delta z$;
- N_{sol}: the average number of solutions of w satisfying $S(w + \Delta w) - S(w) = \Delta z$;
- N_{cost}: the average number of table look-ups.

Then, the ratio $\gamma = \frac{N_{\text{valid}}}{N_{\text{test}}}$ is a good estimation of the probability that a randomly given input-output difference is valid.

According to our experiments, we get $(N_{\text{valid}}, N_{\text{sol}}, N_{\text{cost}}) = (2930, 680, 32)$ for Tip5, $(N_{\text{valid}}, N_{\text{sol}}, N_{\text{cost}}) = (803, 1752, 32)$ for Monolith-64, and $(N_{\text{valid}}, N_{\text{sol}}, N_{\text{cost}}) = (28981, 35, 15)$ for Monolith-31. Based on these data, we make the following conclusions:

- The probability γ that a randomly given $(\Delta w, \Delta z)$ is valid for the S-box of Tip5, Monolith-64 and Monolith-31 is about $2^{-8.5}$, $2^{-10.3}$ and $2^{-5.1}$, respectively;
- The cost to determine whether a random $(\Delta w, \Delta z)$ is valid can be viewed as negligible since the value of N_{cost} is quite small, i.e., our algorithm is considerably efficient.

Note that γ is the estimate of the fraction of non-empty cells[5] in the full DDT of the S-box (also note that there are in total p^2 cells for the DDT), which we do not explicitly build. Since the total number of elements stored in all cells must be p^2, each non-empty cell stores γ^{-1} elements on average. Hence, we also use γ^{-1} to approximate N_{sol} in the complexity analysis of our attacks.

4.5 Activating Partial Small Boxes

In our advanced attacks, we also exploit another property of $z = S(w)$. Specifically, as it is composed of h small-scale permutations $S_0, \ldots, S_{h-1}$ (or we call them small boxes), it is possible to generate an input-output difference pair $(\Delta w, \Delta z)$ such that only ℓ boxes are activated. We call it the *partially activating technique.* Specifically, we have Theorem 1 to generate such $(\Delta w, \Delta z)$ and lower bound the probability of such a difference transition.

Theorem 1 (Partially Activating Technique)**.** *Let the l_{h-1} most significant bits of p be all 1 and $l_i \geq 2$ for $0 \leq i < h$. Let $(\Delta w_0, \ldots, \Delta w_{h-1})$ be the split of Δw with parameters $(l_0, \ldots, l_{h-1})$ where $\sum_{i=0}^{h-1} l_i = \lceil \log_2 p \rceil$, and let $I = \{i_0, \ldots, i_{\ell-1}\} \subseteq \{0, \ldots, h-1\}$ be a set. There exist $\prod_{i \in I}(2^{l_i} - 2)$ different input-output differences $(\Delta w, \Delta z)$ for $z = S(w)$ where*

$$\begin{cases} 0 < \Delta w_i \leq 2^{l_i} - 2 \text{ for } i \in I, \\ \Delta w_i = 0 \text{ for } i \notin I, \\ \Delta z = S(\Delta w) - S(0). \end{cases} \tag{4}$$

Moreover, for each such $(\Delta w, \Delta z)$, the probability that a random $w \in \mathbb{F}_p$ satisfies $S(w + \Delta w) - S(w) = \Delta z$ is at least $p^{-1} \cdot \prod_{i \notin I, 0 \leq i < h}(2^{l_i} - 1)$.

Proof. Since the l_{h-1} most significant bits of p are all 1, we have

$$\Delta w = \sigma^{-1}(\Delta w_0, \ldots, \Delta w_{h-1}) < p$$

for each $(\Delta w_0, \ldots, \Delta w_{h-1})$ satisfying Eq. 4. Hence we have $\prod_{i \in I}(2^{l_i} - 2)$ such different input-output differences $(\Delta w, \Delta z)$ for the S-box $z = S(w)$.

For any $w = \sigma^{-1}(w_0, \ldots, w_{h-1})$ satisfying

$$w_i = 0 \text{ for } i \in I,\ 0 \leq w_i \leq 2^{l_i} - 2 \text{ for } i \notin I \text{ and } 0 \leq i < h,$$

we have $w < p$ and there are $\prod_{i \notin I, 0 \leq i < h}(2^{l_i} - 1)$ different choices of such w.

Hence, given any $(\Delta w, \Delta z)$ satisfying Eq. 4, for any of the above $\prod_{i \notin I, 0 \leq i < h}(2^{l_i} - 1)$ possible values of w, we always have

$$0 \leq \Delta w_i + w_i \leq 2^{l_i} - 2 \text{ for } 0 \leq i < h.$$

In other words, $w + \Delta w = \sigma^{-1}(\Delta w_0 + w_0, \ldots, \Delta w_{h-1} + w_{h-1}) < p$. Let $z = S(w)$ and $z' = S(w + \Delta w)$ for such $(\Delta w, \Delta z)$ and w. By definition, we have

$$\begin{aligned} z' - z &= \sigma^{-1}(S_0(\Delta w_0 + w_0), \ldots, S_{h-1}(\Delta w_{h-1} + w_{h-1})) - \sigma^{-1}(S_0(w_0), \ldots, S_{h-1}(w_{h-1})) \\ &= \sigma^{-1}(u_0, \ldots, u_{h-1}) \end{aligned}$$

where

$$u_i = \begin{cases} S_i(\Delta w_i + w_i) - S_i(w_i) = S_i(\Delta w_i) - S_i(0) & \text{if } i \notin I\,, \\ S_i(0 + w_i) - S_i(w_i) = S_i(0) - S_i(0) & \text{if } i \in I\,. \end{cases}$$

In this way, we have $\sigma^{-1}(u_0, \ldots, u_{h-1}) = S(\Delta w) - S(0)$ by definition. Hence, we always have $z' - z = S(\Delta w) - S(0)$, and the probability that a random $w \in \mathbb{F}_p$ satisfies $S(w + \Delta w) - S(w) = \Delta z$ for such $(\Delta w, \Delta z)$ is at least $p^{-1} \cdot \prod_{i \notin I, 0 \leq i < h}(2^{l_i} - 1)$. □

[5] Each cell stores the conforming inputs w satisfying $S(w + \Delta w) - S(w) = \Delta z$.

5 Two-stage Collision Attacks on Tip5, Tip4 and Tip4'

Let us focus on how to use 1-block input messages and the details of the S-box to devise efficient collision attacks in this section. In this setting, there is no need to consider the padding as it corresponds to the fixed-length input case.

The big picture of the two-stage attack. The big picture of the two-stage attack is to first fix the input difference and output difference of the nonlinear layer $\mathcal{S}(\cdot)$ at the 2nd round, such that there is no difference in the inner part and that all s S-Boxes in the 3rd round are inactive. This is called Stage 1. For Stage 2, we then compute the input of $\mathcal{S}(\cdot)$ at the 2nd round such that a collision can be generated by solving a polynomial system with Gröbner basis.

More details. As illustrated in Figure 7, abusing notation, let

$$\Delta x = (\Delta x_0, \dots, \Delta x_{n-1}),\ \Delta y = (\Delta y_0, \dots, \Delta y_{n-1})$$

denote the input and output difference of $\mathcal{S}(\cdot)$ at the 2nd round. Let $(e_r, \dots, e_{n-1}) \in \mathbb{F}_p^c$ denoted the c state words in the inner part after applying the nonlinear layer $\mathcal{S}$ in the very first round, i.e.,

$$(e_r, \dots, e_{n-1}) = (T(1), \dots, T(1)) = (1, \dots, 1)$$

since $r \geq s$ in Tip5, Tip4 and Tip4', and we target the fixed-length input case. (Recall that c denotes the capacity of the sponge construction.)

For the two-stage collision attack, we further assume that the first $s_1 \leq s$ S-boxes and the first $t_1 \leq t$ T-boxes are active at the 2nd round. In particular, let

$$s_0 = s - s_1,\ t_0 = t - t_1$$

denote the number of inactive S-boxes and T-boxes at the 2nd round, respectively. In other words, we will have

$$\Delta x = (\underbrace{\Delta x_0, \dots, \Delta x_{s_1-1}}_{s_1}, \underbrace{0, \dots, 0}_{s_0}, \underbrace{\Delta x_s, \dots, \Delta x_{s+t_1-1}}_{t_1}, \underbrace{0, \dots, 0}_{t_0})$$
$$\Delta y = (\Delta y_0, \dots, \Delta y_{s_1-1}, 0, \dots, 0, \Delta y_s, \dots, \Delta y_{s+t_1-1}, 0, \dots, 0)$$

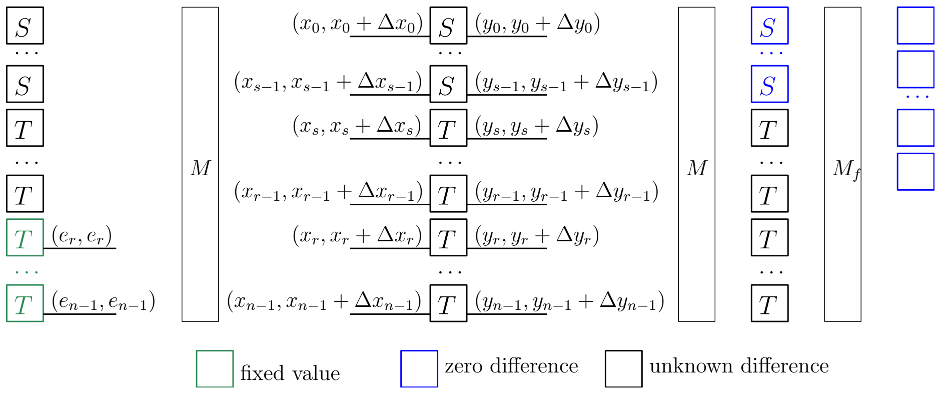

S
S
T
T
T
T
(e_r, e_r)
(e_{n-1}, e_{n-1})
M
$(x_0, x_0 + \Delta x_0)$
$(y_0, y_0 + \Delta y_0)$
$(x_{s-1}, x_{s-1} + \Delta x_{s-1})$
$(y_{s-1}, y_{s-1} + \Delta y_{s-1})$
$(x_s, x_s + \Delta x_s)$
$(y_s, y_s + \Delta y_s)$
$(x_{r-1}, x_{r-1} + \Delta x_{r-1})$
$(y_{r-1}, y_{r-1} + \Delta y_{r-1})$
$(x_r, x_r + \Delta x_r)$
$(y_r, y_r + \Delta y_r)$
$(x_{n-1}, x_{n-1} + \Delta x_{n-1})$
$(y_{n-1}, y_{n-1} + \Delta y_{n-1})$
M
M_f
fixed value
zero difference
unknown difference

Figure 7: Collision attacks on 3-round Tip5, Tip4 and Tip4' in the fixed-length input case, where the inner part is initialized to 1.

The high-level description of the two-stage collision attack is outlined below:

Stage 1: Find valid input-output differences $\Delta x, \Delta y$ for the second nonlinear layer such that there is no difference in the inner part and that all s S-Boxes in the 3rd round are inactive. Subsequently, compute inputs x_i to the active S-boxes and T-Boxes such that

$$\begin{cases} S(x_i + \Delta x_i) - S(x_i) = \Delta y_i & \text{for } 0 \le i < s_1, \\ T(x_i + \Delta x_i) - T(x_i) = \Delta y_i & \text{for } s \le i < s + t_1. \end{cases} \tag{5}$$

Stage 2: Randomly fix input values x_i for $s_1 \le i < s$ to the inactive S-Boxes in the second round (and thus the corresponding outputs $y_i = S(x_i)$). Find a solution for the remaining t_0 variables $(x_{s+t_1}, \ldots, x_{n-1})$ such that the initial inner part value is matched and that there is no difference in the hash value.

As can be observed from the description of Stage 2, we finally only have $t_0 = t - t_1 \le t$ free variables to fulfill $c + d$ equations over $\mathbb{F}_p$. (Recall that d denotes the digest length.) Hence, to make our attack succeed with an overwhelming probability, $c + d \le t$ at least should hold. Hence, in this sense, the attacks should be applicable to Tip5, Tip4 and Tip4'.

In the following, we give a detailed description of the two stages.

5.1 Stage 1: Finding Valid $(\Delta x, \Delta y)$ for the 2nd Nonlinear Layer

To ensure the first condition that there is no difference in the inner part, we can construct a system of c linear equations in $s_1 + t_1$ variables among Δx:

$$\texttt{Sys1:} \qquad \left(M^{-1}(\Delta x)\right)_i = 0 \qquad \text{for } r \le i < n. \tag{6}$$

To ensure the second condition that all S-Boxes in the 3rd round are inactive, that is, that the input difference of each S-box in the 3rd round is zero, we can construct a system of s linear equations in $s_1 + t_1$ variables among Δy:

$$\texttt{Sys2:} \qquad \left(M(\Delta y)\right)_i = 0 \qquad \text{for } 0 \le i < s. \tag{7}$$

Since M is an MDS matrix, all linear equations in $\texttt{Sys1} \cup \texttt{Sys2}$ are linearly independent. To ensure that there is at least one non-zero solution to both systems, the following conditions should hold:

$$s_1 + t_1 > \max\{c, s\}. \tag{8}$$

In total, there are at most $(p^{s_1+t_1-c} - 1) \times (p^{s_1+t_1-s} - 1) \approx p^{2(s_1+t_1)-c-s}$ nonzero solutions $(\Delta x, \Delta y) \in \mathbb{F}_p^n \times \mathbb{F}_p^n$ to the linear equation system $\texttt{Sys1} \cup \texttt{Sys2}$. However, not each such solution is necessarily valid, and we need to consider the details of the S-box and T-box.

Probability of valid S-box transitions. With Algorithm 2 for the S-box in Tip5, each solution of $(\Delta x_0, \ldots, \Delta x_{s_1-1}, \Delta y_0, \ldots, \Delta y_{s_1-1})$ is valid with probability $\gamma^{s_1} = 2^{-8.5 s_1}$, and we can determine whether it is valid in negligible time. In particular, if it is valid, our algorithm also outputs a set of solutions of $(x_0, \ldots, x_{s_1-1})$ such that $S(x_i + \Delta x_i) - S(x_i) = \Delta y_i$ holds for $0 \le i < s_1$.

Probability of valid T-box transitions. It is well-known that for a power map $x \mapsto x^\alpha$, where $\gcd(\alpha, p-1) = 1$, the maximal differential probability is given by $\frac{\alpha-1}{p}$ [Nyb91]. Consequently, any DDT entry is upper bounded by $\frac{\alpha-1}{p}$. Since every row in the DDT must sum up to 1, at least a fraction of $\frac{1}{\alpha-1}$ of DDT entries in a row must be non-zero, i.e., represent a valid transition across the T-box. Since this is true for all rows, we can conclude similarly for the entire table. This implies that a randomly given nonzero input-output difference of this power map is valid with a probability larger than $\frac{1}{\alpha-1}$. Since $T(x) = x^7$, we conclude that each solution of $(\Delta x_s, \ldots, \Delta x_{s+t_1-1}, \Delta y_s, \ldots, \Delta y_{s+t_1-1})$ is valid with probability larger than 6^{-t_1}.

How to determine whether such a solution is valid is also trivial, as we simply need to solve degree-6 univariate equations $(x_i + \Delta x_i)^7 - x_i^7 = \Delta y_i$ for $s \le i < s + t_1$. According to [BBLP22], the time complexity to find the roots of these equations can be estimated as $t_1 \cdot 6 \cdot \log_2 6 \cdot \log_2(6p) \cdot \log_2 \log_2 6 \approx 1415 \cdot t_1$ field operations.

On the constraint on (s_1, t_1, s, c). According to the above analysis, to ensure that there is at least one valid solution, the following condition should also hold:

$$p^{2(s_1+t_1)-c-s} > 2^{8.5s_1} \cdot 6^{t_1}, \tag{9}$$

where we consider the worst case for the DDT of the T-box, i.e., a random input-output difference holds with probability of $\frac{1}{6}$.

Practical implementation. Instead of enumerating all possible solutions to `Sys1` $\cup$ `Sys2`, it is more practical to do the following:

1. Apply Gaussian elimination to `Sys1`, and get $s_1 + t_1 - c$ free variables.
2. Apply Gaussian elimination to `Sys2`, and get $s_1 + t_1 - s$ free variables.
3. Pick a random value for the $s_1 + t_1 - c$ free variables in Δx, and determine the full Δx satisfying `Sys1`.
4. According to Δx, correctly guess the $s_1 + t_1 - s$ free variables in Δy such that the difference transitions in these $s_1 + t_1 - s$ boxes are valid. Specifically, we know the input difference Δx_i of each of these $s_1 + t_1 - s$ boxes (S-box or T-box), and hence we can randomly pick an input x_i and then compute the corresponding output difference Δy_i. Finally, determine the full Δy satisfying `Sys2`.
5. Check if $\Delta x \to \Delta y$ is valid. Due to Step 4, we only need to check difference transitions for s boxes.

It is also easy to observe that we can first determine Δy, and then determine Δx. In this case, we only need to check difference transitions for c boxes. Hence, depending on (c, s), we can adjust the above procedure. Note that in Tip5, Tip4 and Tip4', we have $s \le c$, and hence the above procedure is better. Also, since the valid difference transitions for the S-box have a lower probability than the T-box, it is optimal to have as few S-Boxes as possible to check among the $\min\{c, s\}$ boxes to check at the last step.

Due to the methods to check the validity of an input-output difference pair, if a solution $(\Delta x, \Delta y)$ to `Sys1` $\cup$ `Sys2` is found such that the transition $\Delta x \to \Delta y$ holds, at the same time, we also get the corresponding input-output pair (x_i, y_i) such that

$$\begin{aligned} &S(x_i + \Delta x_i) - S(x_i) = \Delta y_i, && y_i = S(x_i), && \text{for } 0 \le i < s_1, \\ &T(x_i + \Delta x_i) - T(x_i) = \Delta y_i, && y_i = T(x_i), && \text{for } s \le i < s + t_1. \end{aligned}$$

Complexity of Stage 1. The cost to perform Gaussian elimination to `Sys1` and `Sys2` is negligible, as there are only c and s linear equations, respectively. The main cost is caused by finding a valid solution, i.e., checking whether $\Delta x_i \to \Delta y_i$ is valid for $\min\{s, c\}$ boxes. Let the number of T-Boxes and S-Boxes to be checked be s_c and t_c, respectively, i.e.,

$$0 \le s_c \le s_1, \ 0 \le t_c \le t_1, \ s_c + t_c = \min\{s, c\}. \tag{10}$$

Then, we can estimate the time complexity of Stage 1 as

$$T_{\text{stage1}} = 2^{8.5 s_c} \cdot 6^{t_c} + 6^{t_c} \cdot 1415 \cdot t_c \tag{11}$$

field operations, where we treat the cost to check the difference transitions of the S-box as negligible since it just takes a few table lookups, as stated in Section 4.4.

Experimental verification. To demonstrate the efficiency and correctness of our algorithm, we give a concrete example of the input-output difference $(\Delta x, \Delta y)$ for the 2nd nonlinear layer of Tip4 by setting $(s, c, s_1, t_1, s_c, t_c) = (4, 4, 4, 1, 4, 0)$, such that the differences of the inner part and the input differences of all S-boxes at the 3rd round are zero. The expected time complexity is estimated as $2^{8.5\times 4} = 2^{34.0}$. By using 120 threads, we have practically found such a solution of $(\Delta x, \Delta y)$ in a few minutes on one thread after testing about 2^{27} different input pairs (x_4, x_4') for the active T-box, as specified below (in hex). As $120 \times 2^{27} \approx 2^{34}$, our estimation of the time complexity is reasonable.

x_0 = a708181014551f68, x_1 = 03a85a236b0a9435, x_2 = 15420104c8678db9,
x_3 = 1403ae07702a0617, x_4 = 9b348004e33e78c5.
x_0' = 39666624a6262e6b, x_1' = 724e5e9078193528, x_2' = 0b218d9613182324,
x_3' = 5e893f4723a43824, x_4' = 658096b7fa6a0246.
Δy_0 = 42050ef963e37b3b, Δy_1 = 8e14e7cfa1364d79, Δy_2 = 4ba72b3574e557be,
Δy_3 = 0eb6695527b8424d, Δy_4 = fb06f2832291a5d9.

5.2 Stage 2: Finding a Collision

After Stage 1, only $s_0 + t_0$ variables $(x_{s_1}, \ldots, x_{s-1}, x_{s+t_1}, \ldots, x_{n-1})$ remain unknown, symbolizing the inputs and outputs to the nonactive S-Boxes and T-Boxes in the 2nd round. After randomly fixing inputs x_i to the s_0 nonactive S-Boxes and calculating the corresponding outputs $y_i = S(x_i)$, only t_0 variables remain unknown. The reason to fix the inputs x_i to the s_0 nonactive S-Boxes is that we do not know the high-degree expression of the S-box over $\mathbb{F}_p$. It will be costly for the algebraic attack if they are treated as variables because the outputs of these S-Boxes will be high-degree polynomials in the inputs.

Note that after Stage 1 we have ensured that there is no difference in the inner part and the inputs of all S-Boxes in the 3rd round. To ensure that $(e_r, \ldots, e_{n-1})$ can be matched, we can set up c linear equations in the t_0 variables $(x_{s+t_1}, \ldots, x_{n-1})$:

$$(M^{-1}(x - c^{(1)}))_i = e_i \qquad \text{for } r \le i < n. \tag{12}$$

To ensure a hash collision, that is, zero-difference in the digest, we can set up the following d equations also in these t_0 variables:

$$\left(M_f \underbrace{\left(\mathcal{S}\big(M(\mathcal{S}(x + \Delta x)) + c^{(2)}\big) - \mathcal{S}\big(M(\mathcal{S}(x)) + c^{(2)}\big) \right)}_{\text{Output difference of the 3rd nonlinear layer}} \right)_i = 0 \quad \text{for } 0 \le i < d, \tag{13}$$

where Δx is already fixed in the previous stage and $\Delta x_i = 0$ for $s + t_1 \le i < n$. A trivial upper bound for the degree of these d equations is $7^2 - 1 = 48$. However, as only the power map is used for the T-box, the input to the j-th T-box at the 3rd round for the 2nd round input $x + \Delta x$ is a polynomial of the form $\Delta_{1,j} + \sum_{i=s+t_1}^{n-1} \alpha_{i,j} \cdot x_i^7$, while it is $\Delta_{2,j} + \sum_{i=s+t_1}^{n-1} \alpha_{i,j} \cdot x_i^7$ for the 2nd round input x, where $\alpha_{i,j}, \Delta_{1,j}, \Delta_{2,j} \in \mathbb{F}_p$ are some known constants. Hence, the output difference of the j-th T-box at the 3rd round is a polynomial in $(x_{s+t_1}, \ldots, x_{n-1})$ of degree $7 \times 6 = 42$, rather than 48. Note that even though we cannot actually calculate the concrete outputs of the s S-Boxes at the 3rd round, we know that their input differences are zero, and so will their output differences. Therefore, the degree of these d equations is 42.

Overall, there are c equations of degree 1 and d equations of degree 42. The following conditions should hold to ensure the existence of a solution to such an equation system:

$$t_0 = t - t_1 = n - s - t_1 \geq c + d. \tag{14}$$

After performing Gaussian elimination on the c linear equations, we only need to solve d degree-42 equations in $t_0 - c$ variables. Hence, we can guess $t_0 - c - d$ variables if $t_0 > c + d$, and solve d degree-42 equations in d variables. Otherwise, we have $t_0 = c + d$, and also need to solve d degree-42 equations in d variables. Once a solution is found, we get a 3-round collision.

Complexity of Stage 2. Under the regularity assumption on the d degree-42 equations in d variables, the Macaulay bound is given by $d_{\text{MAC}} = 1 + (42 - 1) \cdot d$. Under the assumption that the ideal is zero-dimensional, the Bézout bound is given by $B = 42^d$. Thus, the time complexity (measured in operations over $\mathbb{F}_p$) for stage 2 is estimated as

$$T_{\text{stage2}} = \max\left\{\binom{d + d_{\text{MAC}}}{d}^{\omega}, d \cdot B^3\right\} = \max\left\{\binom{1 + 42d}{d}^{\omega}, d \cdot 42^{3d}\right\}. \tag{15}$$

5.3 Applications to the Collision Attack on 3-Round Tip4

There are several constraints to ensure the application of the 2-stage collision attacks, i.e., (s_1, t_1, c, s, n, d) have to satisfy Eq. 8, Eq. 9 and Eq. 14. Unfortunately, since $(n, c, s, d) = (16, 6, 4, 5)$ in Tip5 and $(n, c, s, d) = (12, 4, 4, 4)$ in Tip4', these constraints cannot hold simultaneously, i.e., we do not have enough available degrees of freedom.

On the contrast, this two-stage method is quite efficient for 3-round Tip4. For simple implementations, i.e., without using Algorithm 2, we set $(s_1, t_1, s_c, t_c) = (1, 4, 0, 4)$. In this case, we have $T_{\text{stage1}} \approx 2^{22.9}$. For Stage 2, we only need to solve 4 degree-42 equations in 4 variables, whose cost is estimated as $T_{\text{stage2}} = \max\{2^{25.0\omega}, 4 \cdot 42^{12}\} = \max\{2^{25.0\omega}, 2^{66.8}\}$.

Hence, the time complexity of a collision attack on 3-round Tip4 is estimated as $\max\{2^{25.0\omega}, 2^{66.8}\}$ field operations under the regularity assumption.

Experimental verification. Due to the relatively high cost to find a real collision, we turn to finding a near collision where only 2 rather than 4 output words collide. As shown in Table 5, experiments have confirmed that the polynomial system behaves like a regular system and using $\omega = 2.37$ does not underestimate the complexity. A concrete near collision is given in Table 8.

Table 5: Experimental results for the near collision attack on 3-round Tip4, i.e., collision in 2 rather than 4 output state words. The running time in the columns "T_{GB}" and "T_{FGLM}" is in seconds.

Target	T_{GB}	d_{MAC}	d_{reg}	$\mathcal{C}_{\text{GB}}$ ($\omega = 2.37$)	T_{FGLM}	B	$d_{\mathcal{I}}$	$\mathcal{C}_{\text{FGLM}}$ ($\omega = 3$)	Memory (MB)
Tip4	1	83	83	$2^{11.9\omega}$ ($2^{28.2}$)	139	42^2	42^2	$2^{1+10.8\omega}$ ($2^{33.4}$)	597.22

5.4 Straightforward Applications to 3-Round SFS Collision Attacks

Next, we show how to adapt the previous attack in the case of a SFS attack scenario. Compared with the collision attacks, we no more need to match $(e_r, \ldots, e_{n-1})$ in the SFS collision attacks. Hence, the above 2-stage attack can be easily adapted to the 3-round SFS collision attack.

Specifically, at Stage 1, we perform the same procedure to find a desired input difference Δx and output difference Δy for the 2nd nonlinear layer. Then, at Stage 2, instead of treating x as variables, we can treat y as variables as we do not need to match $(e_r, \dots, e_{n-1})$. Our remaining task is to ensure that there is no difference in the hash value.

Note that after Stage 1, $s_1 + t_1$ variables in y have been fixed to ensure $(\Delta x, \Delta y)$. We are still left with $s_0 + t_0$ unknown variables, i.e.,

$$(y_{s_1}, \dots, y_{s-1}, y_{s+t_1}, \dots, y_{n-1}).$$

Different from the above collision attack, we no longer need to fix the s_0 outputs (i.e., inputs) of the nonactive S-Boxes at the 2nd round since we start from the output of the second nonlinear layer, and there is no need to match the inner part.

Then, to ensure that there is no difference in the hash value, we only need to set up d degree-6 equations in y, as shown below:

$$\left(M_f\,\underbrace{\left(\mathcal{S}\big(M(y+\Delta y)+c^{(2)}\big)-\mathcal{S}\big(M(y)+c^{(2)}\big)\right)}_{\text{Output difference of the 3rd nonlinear layer}}\right)_i = 0 \qquad \text{for } 0 \le i < d, \tag{16}$$

Hence, to ensure that there is at least one solution, we have the constraint

$$s_0 + t_0 = n - s_1 - t_1 \ge d. \tag{17}$$

If $n - s_1 - t_1 > d$, we can guess $n - s_1 - t_1 - d$ variables, and then solve d degree-6 equations in d variables.

Complexity of Stage 1 and Stage 2. This 3-round SFS collision attack can work for Tip5, Tip4 and Tip4'. By setting $s_1 = 0$, we still have sufficient degrees of freedom, and only need to solve d degree-6 equations for the SFS collision attacks on 3-round Tip5, Tip4 and Tip4', respectively. The time complexity for Stage 1 is then $6^4 \times 1415 \times 4$ field operations, where we only activate $c+1$ T-boxes and at the 2nd round, i.e., $s_1 = 0$ and $t_1 = c+1$. Hence, the cost of Stage 1 is negligible. At Stage 2, under the regularity assumption, the cost to solve such a polynomial system is estimated as $\max\left\{\binom{1+6d}{d}^{\omega},\, d\cdot 6^{3d}\right\}$ field operations.

Experimental verification. Experiments have confirmed that these SFS collision attacks are practical, as shown in Table 6. It is found that these systems behave like regular systems and using $\omega = 2.37$ does not underestimate the complexity. Concrete SFS colliding message pairs are given in Table 8.

Table 6: Practical results of Stage 2 of the two-stage SFS collision attacks on 3-round Tip5, Tip4 and Tip4'. The running time in the columns "T_{GB}" and "T_{FGLM}" is in seconds.

Target	T_{GB}	d_{MAC}	d_{reg}	$\mathcal{C}_{\mathrm{GB}}$ ($\omega = 2.37$)	T_{FGLM}	B	$d_{\mathcal{I}}$	$\mathcal{C}_{\mathrm{FGLM}}$ ($\omega = 3$)	Memory (MB)
Tip5	2503	26	26	$2^{17.4\omega}$ $(2^{41.2})$	17355	6^5	6^5	$2^{2.4+13.0\omega}$ $(2^{41.4})$	13646.72
Tip4	8	21	21	$2^{13.7\omega}$ $(2^{32.5})$	92	6^4	6^4	$2^{2.0+10.4\omega}$ $(2^{33.2})$	345.66
Tip4'	7	21	21	$2^{13.7\omega}$ $(2^{32.5})$	87	6^4	6^4	$2^{2.0+10.4\omega}$ $(2^{33.2})$	345.66

5.5 Application to SFS Collision Attack on 4-Round Tip4

We now explain how the above 3-round SFS collision attack can be extended to 4 rounds. However, due to the issue of available degrees of freedom, this attack only works for Tip4.

Therefore, for simplicity, we will not describe the attack in a general way, but instead focus on the parameters of Tip4, though it is easy to adapt it to the general case.

Let us still use the same notation as in the above attacks. The 4-round SFS collision attack also has 2 stages, as specified below:

Stage 1: Find $(\Delta x, \Delta y)$ such that there is no difference in the inner part, and fix them, where there are s_1 active S-boxes and t_1 active T-boxes.

Stage 2: Set $(y_{s_1}, \ldots, y_{s-1}, y_{s+t_1}, \ldots, y_{15})$ as $16 - s_1 - t_1$ unknown variables, and set up

- 4 linear equations to ensure that the inputs to the 4 S-boxes at the 3rd round are randomly chosen constants;
- 4 degree-6 equations to ensure that the input differences of the 4 S-boxes at the 4th round are 0;
- 4 degree-48 equations to ensure that the difference of the hash value is 0.

According to Stage 2, we need at least 12 unknown variables among y. On the other hand, at Stage 1, to ensure that there is a nonzero solution to Δx, at least 5 boxes have to be active. If we fix the input and output values for all active S-boxes and T-boxes as in previous attacks, we then only have at most $16 - 5 = 11$ unknown variables among y, i.e., $16 - s_1 - t_1 \leq 11$, which will make the Stage 2 succeed with a probability of p^{-1}. To address this issue, we propose a dedicated method for Stage 1.

Refined Stage 1 for the 4-round SFS collision attack. Specifically, we will set 4 T-boxes and 1 S-box as active, i.e.,

$$s_1 = 1, \; t_1 = 4.$$

However, $\Delta x_0 \neq 0$ is not randomly chosen. Instead, we use the partially activating technique to activate only 1 small box in the S-box to find a $(\Delta x_0, \Delta y_0)$, i.e., set $\ell = 1$. Hence, there are in total $8 \times (2^8 - 2)$ possible valid choices of $(\Delta x_0, \Delta y_0)$.

Randomly choose a valid $(\Delta x_0, \Delta y_0)$, and then solve 4 linear equations in $(\Delta x_4, \ldots, \Delta x_7)$ specified below:

$$\left(M^{-1}(\Delta x)\right)_i = 0, \text{ for } 12 \leq i < 16.$$

In this way, we get a desired Δx. Then, we randomly choose values for $(x_4, \ldots, x_7)$, and compute $(y_4, \ldots, y_7)$ as well as $(\Delta y_4, \ldots, \Delta y_7)$. In this way, $(\Delta x, \Delta y)$ is fully determined and there exists x to ensure such a difference transition.

Stage 2 for the 4-round SFS collision attack. After Stage 1, $(\Delta x, \Delta y)$ are fully determined, and hence the input differences of the S-boxes at the 3rd round are also known. Moreover, $(y_4, \ldots, y_7)$ are also fixed.

At Stage 2, we need to set up 12 equations. Different from the 4 active T-boxes, although $(\Delta x_0, \Delta y_0) \neq (0, 0)$ have been determined, we can still set y_0 as an unknown, and each solution of y_0 can ensure $S(S^{-1}(y_0) + \Delta x_0) - S(S^{-1}(y_0)) = \Delta y_0$ with probability larger than 2^{-8} according to Theorem 1.

Hence, we set

$$(y_0, \ldots, y_3, y_8, \ldots, y_{15})$$

as 12 unknowns.

First, we set the inputs of the $s = 4$ S-boxes at the 3rd round as 4 randomly chosen numbers $(a_0, \ldots, a_3) \in \mathbb{F}_p^4$, and set up the following 4 linear equations:

$$\left(M(y + \Delta y) + c^{(2)}\right)_i = a_i \qquad \text{for } 0 \leq i < 4.$$

Next, we set up $s = 4$ degree-6 equations to ensure the input differences of the S-boxes at the 4th round are zero:

$$\Big(\underbrace{M\Big(\mathcal{S}\big(M(y+\Delta y)+c^{(2)}\big)-\mathcal{S}\big(M(y)+c^{(2)}\big)\Big)}_{\text{Input difference of the 4th nonlinear layer}}\Big)_i = 0 \qquad \text{for } 0 \le i < 4.$$

Finally, we set up $d = 4$ degree-48 equations to ensure a collision in the hash value, where $0 \le i < 4$:

$$\big(M_f(\underbrace{S(M(\mathcal{S}(M(y+\Delta y)+c^{(2)}))+c^{(3)})-S(M(\mathcal{S}\big(M(y)+c^{(2)}\big))+c^{(3)}))}_{\text{Output difference of the 4th nonlinear layer}}\big)_i = 0.$$

By first performing Gaussian elimination on the 4 linear equations, we then only need to solve 4 degree-6 equations, and 4 degree-48 equations in 8 variables. Hence, we can expect one solution with probability close to 1.

However, not each solution is necessarily valid, since we further need to check whether $S(S^{-1}(y_0)+\Delta x_0) - S(S^{-1}(y_0)) = \Delta y_0$ holds, whose probability is at least 2^{-8}. If the solution is invalid, we can choose another value for $(a_0, \ldots, a_3)$, and repeat the attack.

Based on the above analysis, under the regularity assumption on the 8 polynomials, the time complexity of the 4-round SFS collision attack on Tip4 can be estimated as

$$2^8 \times \max\left\{\binom{1+5\times 4+47\times 4+8}{8}^{\omega}, 8\cdot(6\cdot 48)^{12}\right\} = \max\{2^{46.6\omega+8}, 2^{101.1}\}$$

field operations. Under $\omega = 2.37$, the complexity of our attack is about $2^{118.5}$ field operations.

Remark 2. We remark that there are not sufficiently many available degrees of freedom to mount similar successful SFS collision attacks on 4-round Tip5 or Tip4'. However, if we do not fix $(\Delta x, \Delta y)$ in advance, and also treat them as variables, we can devise the 4-round SFS collision attacks on the two ciphers, but the time complexity is too high even with $\omega = 2$, and they are not meaningful. A quite similar attack without fixing $(\Delta x, \Delta y)$ can be later found in the SFS collision attack on 3-round Monolith-64.

6 Three-stage Collision Attacks on 3-Round Tip5 and Tip4'

To address the issue of insufficient degrees of freedom in the 3-round collision attacks on Tip5 and Tip4', we propose the so-called three-stage collision attacks in this section, where we consider to use multiple input blocks rather than a single input block. In particular, we focus on the inner collision attack in order to ignore the padding rule, i.e., we consider the collision in the last c state words rather than the first d state words. Note that the same idea can also be directly applied to the collision attack on the first d state words when considering padding.

6.1 Finding a Collision in the Inner Part with 2 Input Blocks

Let us focus on the problem to find a collision in the inner part rather than in the hash value. In this case, instead of constructing equations to ensure no difference in the first d state words after 3 rounds, we need to set up c equations to ensure no difference in the last c state words after 3 rounds, as illustrated in Figure 8.

In the three-stage collision attack, the 1st input block pair (m_0, m_0') is absorbed, and the corresponding pair $(e_i, e_i') = (e_i, e_i + \Delta e_i)$ for $r \le i < n-1$ is computed, which is the output pair of the inner part after the 1st nonlinear layer in the next permutation to absorb the 2nd input block (m_1, m_1').

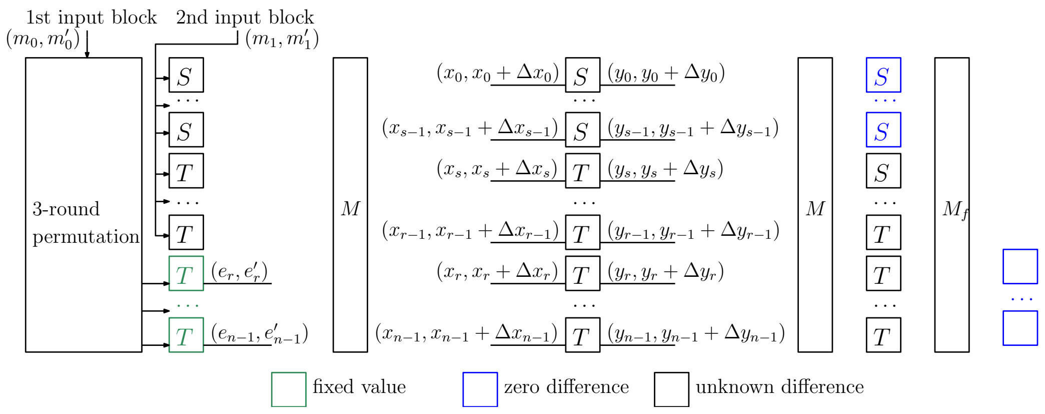

1st input block
2nd input block
(m_0, m_0')
(m_1, m_1')
3-round permutation
S
T
(e_r, e_r')
(e_{n-1}, e_{n-1}')
M
$(x_0, x_0 + \Delta x_0)$
$(y_0, y_0 + \Delta y_0)$
$(x_{s-1}, x_{s-1} + \Delta x_{s-1})$
$(y_{s-1}, y_{s-1} + \Delta y_{s-1})$
$(x_s, x_s + \Delta x_s)$
$(y_s, y_s + \Delta y_s)$
$(x_{r-1}, x_{r-1} + \Delta x_{r-1})$
$(y_{r-1}, y_{r-1} + \Delta y_{r-1})$
$(x_r, x_r + \Delta x_r)$
$(y_r, y_r + \Delta y_r)$
$(x_{n-1}, x_{n-1} + \Delta x_{n-1})$
$(y_{n-1}, y_{n-1} + \Delta y_{n-1})$
M_f
fixed value
zero difference
unknown difference

Figure 8: The three-stage 3-round collision attacks, where padding can be ignored.

The big picture. Let us use the same notation as in the two-stage collision attacks. In particular, $(\Delta x, \Delta y)$ are the input and output difference of the 2nd nonlinear layer, respectively, where there are s_1 active S-boxes and t_1 T-boxes at the 2nd round. In our new attacks on 3-round Tip5 and Tip4, we set

$$s_1 = s,\ t_1 = 0.$$

Hence, we have $s_0 = s - s_1 = 0$ and $t_0 = n - s - t_1 = n - s$, and $(\Delta x, \Delta y)$ are of the following form:

$$\Delta x = (\Delta x_0, \ldots, \Delta x_{s-1}, \underbrace{0, \ldots, 0}_{n-s}),\ \Delta y = (\Delta y_0, \ldots, \Delta y_{s-1}, \underbrace{0, \ldots, 0}_{n-s}).$$

If we use the same strategy as in the two-stage attack, where all s S-boxes at the 3rd round are inactive, we then get s linear equations in s unknowns $(\Delta y_0, \ldots, \Delta y_{s-1})$. In this case, the only solution to this linear system is $\Delta y_0 = \ldots = \Delta y_{s-1} = 0$, resulting in $\Delta x = \Delta y = (\underbrace{0, \ldots, 0}_{n})$ and an invalid collision attack. Hence, the previous strategy to find such a $(\Delta x, \Delta y)$ has to be revised.

In the three-stage collision attack, we have 3 stages called Stage 0, Stage 1 and Stage 2. The high-level description is outlined below:

Stage 0: Collect some pairs (m_0, m_0') that can lead to some differences $(\Delta e_r, \ldots, \Delta e_{n-1})$ satisfying certain conditions.

Stage 1: Find a proper solution of the above $(\Delta x, \Delta y)$ and fix the input pairs for all active boxes at the 2nd round. Different from the previous two-stage attack, we allow active S-boxes to appear at the 3rd round, but we activate the S-boxes with the partially activating technique.

Stage 2: Find a solution of the input x to the 2nd round such that we can get a collision.

Pre-processing phase if $s < c$. Note that we have the following c linear equations in Δx:

$$\texttt{Sys4:} \qquad \left(M^{-1}(\Delta x)\right)_i = \Delta e_i = e_i' - e_i \qquad \text{for } r \le i < n, \tag{18}$$

This system is overdefined since there are only s unknowns $(\Delta x_0, \ldots, \Delta x_{s-1})$ and $c > s$ linear equations. For convenience, rewrite the equation system as

$$M'_{c\times s} \times (\Delta x_0, \ldots, \Delta x_{s-1})^T = E_{c\times c} \times (\Delta e_r, \ldots, \Delta e_{n-1})^T,$$

where $M'_{c\times s}$ is the coefficient matrix with c rows and s columns, and $E_{c\times c}$ is an identity matrix with c rows. Apply Gaussian elimination to $M'_{c\times s}||E_{c\times c}$, and get $M''_{c\times s}||E'_{c\times c}$ such that the matrix $M''_{c\times s}$ is in row echelon form. Since M is an MDS matrix, the rank of $M''_{c\times s}$ must be s, and the last $c-s$ rows of $M''_{c\times s}$ must be all zero. Let $E''_{(c-s)\times c}$ be the submatrix of $E'_{c\times c}$ composed by the last $c-s$ rows of $E'_{c\times c}$. Then, if $(\Delta e_r,\ldots,\Delta e_{n-1})$ satisfy Eq. 19:

$$E''_{(c-s)\times c}\times(\Delta e_r,\ldots,\Delta e_{n-1})^T=(\underbrace{0,\ldots,0}_{c-s})^T, \tag{19}$$

there is always a solution of Δx to `Sys4`. In particular, Eq. 19 is equivalent to

$$E''_{(c-s)\times c}\times(e_r,\ldots,e_{n-1})^T=E''_{(c-s)\times c}\times(e'_r,\ldots,e'_{n-1})^T. \tag{20}$$

Stage 0: Finding many distinct $(e_r,\ldots,e_{n-1},e'_r,\ldots,e'_{n-1})$ satisfying Eq. 20. Let the required number of distinct $(e_r,\ldots,e_{n-1},e'_r,\ldots,e'_{n-1})$ satisfying Eq. 20 be p^β. Note that $(e_r,\ldots,e_{n-1})$ is uniquely determined by m_0. Hence, if $s<c$, finding $(e_r,\ldots,e_{n-1},e'_r,\ldots,e'_{n-1})$ satisfying Eq. 20 is equivalent to finding (m_0,m'_0) such that the corresponding

$$E''_{(c-s)\times c}\times(e_r,\ldots,e_{n-1})^T \text{ and } E''_{(c-s)\times c}\times(e'_r,\ldots,e'_{n-1})^T$$

collide. With the birthday attack, finding p^β such (m_0,m'_0) takes time complexity $\sqrt{2p^{\beta+c-s}}$.

For Tip4 and Tip4', we have $s=c$. Indeed, if $s\geq c$, we can simply randomly choose about $\sqrt{2p^\beta}$ different m_0, and form p^β distinct $(e_r,\ldots,e_{n-1},e'_r,\ldots,e'_{n-1})$. For each such a choice, they can always ensure that there is a solution to Eq. 18, since M is an MDS matrix and the number of linear equations is not larger than the number of variables. The time complexity is then simply $\sqrt{2p^\beta}$.

Stage 1: Finding $(\Delta x,\Delta y)$. As a main difference to the two-stage collision attack, we allow each S-box to be active in the 3rd round, but we choose its input-output difference with the partially activating technique stated in Section 4.5. Specifically, if there are in total $\ell_1>0$ active small boxes among the s S-boxes at the 3rd round, there are in total $(2^8-2)^{\ell_1}\cdot\binom{8s}{\ell_1}$ possible choices for the input-output differences for the S-boxes at the 3rd round with the partially activating technique. For convenience, let the input difference and output difference of the S-boxes at the 3rd round be

$$(\Delta x_0^{(3)},\ldots,\Delta x_{s-1}^{(3)}),(\Delta y_0^{(3)},\ldots,\Delta y_{s-1}^{(3)}),$$

respectively. Then, the procedure of Stage 1 is as follows:

Step 1: Pick one solution of $(\Delta e_r,\ldots,\Delta e_{n-1})$ obtained at Stage 0, and solve `Sys4` to obtain one solution of Δx.

Step 2: Pick one solution of $(\Delta x_0^{(3)},\ldots,\Delta x_{s-1}^{(3)}),(\Delta y_0^{(3)},\ldots,\Delta y_{s-1}^{(3)})$ as described above, and solve `Sys5` to get a solution of Δy.

$$\texttt{Sys5:}\qquad \big(M(\Delta y)\big)_i=\Delta x_i^{(3)} \qquad \text{for } 0\leq i<s. \tag{21}$$

Note that the number of variables and equations in `Sys5` are both s. Since M is an MDS matrix, there must exist a unique nonzero solution.

Step 3: Check the validity of $\Delta x\to\Delta y$. If it is invalid, return to Step 2. If all possible $(\Delta x_0^{(3)},\ldots,\Delta x_{s-1}^{(3)},\Delta y_0^{(3)},\ldots,\Delta y_{s-1}^{(3)})$ are used, return to Step 1. If $\Delta x\to\Delta y$ is valid, exit.

As there are p^β possible choices of $(\Delta e_r, \ldots, \Delta e_{n-1})$, and $(2^8-2)^{\ell_1} \times \binom{8s}{\ell_1}$ possible choices of $(\Delta x_0^{(3)}, \ldots, \Delta x_{s-1}^{(3)})$, we can expect to have in total $p^\beta \cdot (2^8-2)^{\ell_1} \cdot \binom{8s}{\ell_1}$ solutions to $\texttt{Sys4} \cup \texttt{Sys5}$. However, each solution is valid with probability of $\gamma^s = 2^{-8.5s}$, and thus we need the following condition to ensure the existence of a valid $(\Delta x, \Delta y)$:

$$p^\beta \cdot (2^8-2)^{\ell_1} \cdot \binom{8s}{\ell_1} \geq \gamma^{-s} = 2^{8.5s}.$$

The time complexity of Stage 1 to find 1 proper $(\Delta x, \Delta y)$ is then estimated as $2^{8.5s}$.

Stage 2: Finding a collision. After fixing $(\Delta x, \Delta y)$, we fix $(x_0, \ldots, x_{s-1})$ that ensure such a difference transition. It is expected to have $2^{8.5s}$ possible choices of $(x_0, \ldots, x_{s-1})$. Then, we set $(x_s, \ldots, x_{n-1})$ as $n-s$ unknowns.

Then, we set up $c = n - r$ linear equations in these $n-s$ unknowns to match the inner part, and $c = n - r$ degree-42 equations in these $n-s$ unknowns to ensure the collision in the inner part. These equations are as follows:

$$\begin{aligned} (M^{-1}(x - c^{(1)}))_i &= e_i \qquad \text{for } r \leq i < n, \\ \left(M_f \underbrace{\left(\mathcal{S}\big(M(\mathcal{S}(x+\Delta x)) + c^{(2)}\big) - \mathcal{S}\big(M(\mathcal{S}(x)) + c^{(2)}\big) \right)}_{\text{Output difference of the 3rd nonlinear layer}} \right)_i &= 0 \qquad \text{for } r \leq i < n, \end{aligned}$$

Hence, at Stage 2, we need to solve c degree-42 equations in $n - s - c$ unknowns.

Similarly, for each solution of these $n-s$ unknowns, it is valid with probability of $2^{-8\ell_1}$, and we then either try another $(x_0, \ldots, x_{s-1})$ or $(\Delta x, \Delta y)$ to repeat Stage 2.

6.2 Application to 3-Round Tip5 and Tip4'

Application to Tip5. In Tip5, $(n, s, c) = (16, 4, 6)$. Hence, the pre-processing phase is required, and the time complexity of Stage 0 is $\sqrt{2p^{\beta+2}}$. The time complexity of Stage 1 is $2^{8.5s} = 2^{34.0}$. Since $n - s - c = c = 6$, at Stage 2, we will need to solve 6 degree-42 equations in 6 variables. Let $8\ell_1 \leq 8.5s = 34.0 \to \ell_1 \leq 4$. The time complexity is then

$$\sqrt{2p^{\beta+2}} + 2^{34.0} + 2^{8\ell_1} \cdot \max\left\{ \binom{1 + 42 \times 6}{6}^\omega, 6 \cdot 42^{18} \right\} \approx \sqrt{p^\beta} \cdot 2^{64.5} + 2^{8\ell_1} \cdot \max\{2^{38.4\omega}, 2^{99.7}\},$$

where

$$1 \leq \ell_1 \leq 4,\ p^\beta \cdot (2^8-2)^{\ell_1} \cdot \binom{32}{\ell_1} \geq 2^{34.0}.$$

Under $\omega = 2.37$, the optimal choice is $(p^\beta, \ell_1) = (2^{21.1}, 1)$, and the complexity is $\max\{2^{8+38.4\omega}, 2^{107.7}\} = 2^{107.7}$.

Application to Tip4'. While the effect for Tip5 is limited, the new strategy is quite suitable for Tip4'. In Tip4', $(n, s, c) = (12, 4, 8)$. Hence, $s = c$ and there is no need to do the pre-processing phase, and the time complexity of Stage 0 is $\sqrt{2p^\beta}$. The time complexity of Stage 2 is also $2^{8.5s} = 2^{34.0}$. At Stage 2, we need to solve $c = 4$ degree-42 equations in $n - s - c = 4$ unknowns. Hence, the time complexity of the attack on 3-round Tip4' is

$$\sqrt{2p^\beta} + 2^{34.0} + 2^{8\ell_1} \cdot \max\left\{ \binom{1 + 42 \times 4}{4}^\omega, 4 \cdot 42^{12} \right\} \approx \sqrt{p^\beta} \cdot 2^{0.5} + 2^{8\ell_1} \cdot \max\{2^{25.0\omega}, 2^{66.8}\},$$

where $1 \leq \ell_1 \leq 4$, $p^\beta \cdot (2^8-2)^{\ell_1} \cdot \binom{32}{\ell_1} \geq 2^{34.0}$. Under $\omega = 2.37$, the optimal choice is $(p^\beta, \ell_1) = (2^{21.1}, 1)$, and the complexity is $\max\{2^{8+25.0\omega}, 2^{74.8}\} = 2^{74.8}$.

Application to Tip5 considering padding. Since $d = 5 < c = 6$ in Tip5, to reduce the number of nonlinear equations to solve at Stage 2, we can also consider a direct two-block collision attack on 3-round Tip5 with a similar idea by setting the length of second message block as $r - 1 = 11$. In this case, at the pre-processing phase, we need to solve $c + 1 = 7$ rather than $c = 6$ linear equations in $s = 4$ variables, and hence the time complexity of Stage 0 is $\sqrt{2p^{\beta+3}}$ if we want to find p^β pairs of (m_0, m_0') such that such an overdefined linear equation system is solvable. To procedure to find $(\Delta x, \Delta y)$ at Stage 1 is almost the same. At Stage 2, instead of considering collisions in the last $c = 6$ state words, we consider the collision in the first $d = 5$ state words, i.e., we solve $c + 1 = 7$ linear equations and $d = 5$ degree-42 equations instead of $c = 6$ linear equations and $c = 6$ degree-42 equations. Hence, if $8\ell_1 \le 34.0 \to \ell_1 \le 4$, the time complexity becomes

$$\sqrt{2p^{\beta+3}} + 2^{34.0} + 2^{8\ell_1} \cdot \max\left\{ \binom{1 + 42 \times 5}{5}^{\omega}, 5 \cdot 42^{15} \right\} \approx \sqrt{p^\beta} \cdot 2^{96.5} + 2^{8\ell_1} \cdot \max\{2^{31.7\omega}, 2^{83.3}\}.$$

where $p^\beta \cdot (2^8 - 2)^{\ell_1} \cdot \binom{32}{\ell_1} \ge 2^{34.0}$. Under $\omega = 2.37$, the optimal choice is $(p^\beta, \ell) = (2^{9.1}, 2)$, and the complexity is $2^{101.1} + \max\{2^{31.7\omega+16}, 2^{99.3}\} \approx 2^{101.1}$.

7 Application to Round-reduced Monolith

In this section, we describe a similar collision attack on 2-round Monolith. In addition, we also present a SFS collision attack on 3-round Monolith.

7.1 On the Composition $y = \mathcal{F}(\mathcal{B}(x))$

Our nontrivial 2-round collision attacks on Monolith are highly related to the differential property of the special nonlinear linear $y = \mathcal{F}(\mathcal{B}(x))$. Hence, we first study such a nonlinear layer.

Abusing notation, let $x = (x_0, \ldots, x_{n-1})$ and $y = (y_0, \ldots, y_{n-1})$ be be the input and output of composition $\mathcal{F} \circ \mathcal{B}$, respectively. Let the input and output difference of $\mathcal{F} \circ \mathcal{B}$ be $\Delta x = (\Delta x_0, \ldots, \Delta x_{n-1})$ and $\Delta y = (\Delta y_0, \ldots, \Delta y_{n-1})$, respectively. Then, we have

$$\Delta y_i = \begin{cases} S(x_i + \Delta x_i) - S(x_i) & \text{for } i = 0, \\ \big(S(x_{i-1} + \Delta x_{i-1})^2 - S(x_{i-1})^2\big) + \big(S(x_i + \Delta x_i) - S(x_i)\big) & \text{for } 0 < i < s, \\ \big(S(x_{i-1} + \Delta x_{i-1})^2 - S(x_{i-1})^2\big) + \Delta x_i & \text{for } i = s, \\ \big((x_{i-1} + \Delta x_{i-1})^2 - x_{i-1}^2\big) + \Delta x_i & \text{for } s < i < n. \end{cases} \tag{22}$$

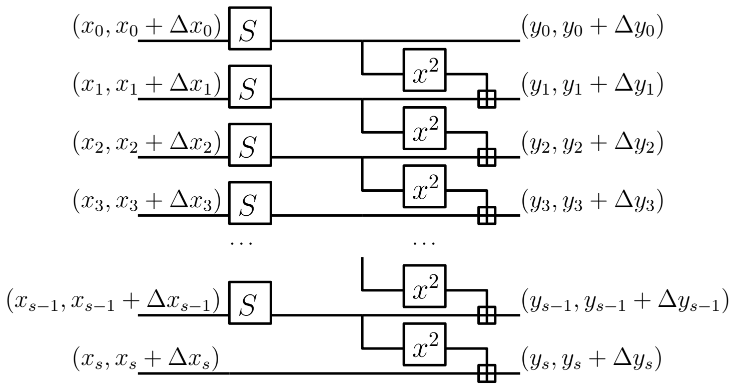

$(x_0, x_0 + \Delta x_0)$
S
$(y_0, y_0 + \Delta y_0)$
$(x_1, x_1 + \Delta x_1)$
x^2
$(y_1, y_1 + \Delta y_1)$
$(x_2, x_2 + \Delta x_2)$
$(y_2, y_2 + \Delta y_2)$
$(x_3, x_3 + \Delta x_3)$
$(y_3, y_3 + \Delta y_3)$
$\cdots$
$(x_{s-1}, x_{s-1} + \Delta x_{s-1})$
$(y_{s-1}, y_{s-1} + \Delta y_{s-1})$
$(x_s, x_s + \Delta x_s)$
$(y_s, y_s + \Delta y_s)$

Figure 9: The composition $y = \mathcal{F}(\mathcal{B}(x))$

As shown in Figure 9, let us only focus on the first $s+1$ state words. In particular, we are interested in the following problem:

- If $\Delta x_0 \to \Delta y_0$ is a valid difference transition through the S-box, for a randomly given $(\Delta x_1, \ldots, \Delta x_s)$ and $(\Delta y_1, \ldots, \Delta y_s)$, how to efficiently determine whether $(\Delta x_0, \ldots, \Delta x_s) \to (\Delta y_0, \ldots, \Delta y_s)$ is a valid difference transition through $y = \mathcal{F}(\mathcal{B}(x))$, and get the conforming input $(x_0, \ldots, x_{s-1})$ satisfying the first $s+1$ equations in Eq. 22?

Due to the special S-box used in Monolith, we propose the following algorithmic procedure to solve the above problem:

Step 1: With Algorithm 2, get all valid x_0 ensuring $S(x_0 + \Delta x_0) - S(x_0) = \Delta y_0$.

Step 2: Run recursively for $0 < i < s$. For each valid x_{i-1}, compute $x'_{i-1} = x_{i-1} + \Delta x_{i-1}$ and

$$\Delta y_i - \left((S(x'_{i-1}))^2 - (S(x_{i-1}))^2\right). \tag{23}$$

Check whether $\Delta x_i \to \Delta y_i - \left((S(x'_{i-1}))^2 - (S(x_{i-1}))^2\right)$ is a valid difference transition through the S-box with Algorithm 2. If not, try another valid x_{i-1}, and if all possible valid x_{i-1} are used up, start backtracking. Otherwise, store all possible x_i ensuring $S(x_i + \Delta x_i) - S(x_i) = \Delta y_i - \left((S(x'_{i-1}))^2 - (S(x_{i-1}))^2\right)$.

Step 3: For each valid x_{s-1}, compute $x'_{s-1} = x_{s-1} + \Delta x_{s-1}$ and check whether

$$(S(x'_{s-1}))^2 - (S(x_{s-1}))^2 + \Delta x_s = \Delta y_s \tag{24}$$

holds. If it holds, $(\Delta x_0, \ldots, \Delta x_s)$ and $(\Delta y_0, \ldots, \Delta y_s)$ correspond to a valid difference transition, and we also get the inputs $(x_0, \ldots, x_{s-1})$ ensuring this difference transition.

Analysis of the above algorithmic procedure. Abusing notation, let γ be the probability that a randomly given value of the input-output difference of the S-box is valid. As $\Delta x_0 \to \Delta y_0$ is a valid difference transition, we expect to get on average γ^{-1} solutions of x_0 ensuring this difference transition, and can traverse them to check $\Delta x_1 \to \Delta y_1 - \left((S(x'_0))^2 - (S(x_0))^2\right)$. Recursively for $1 \le i < s$, we get γ^{-1} solutions of x_{i-1}, and then traverse them to check $\Delta x_i \to \Delta y_i - \left((S(x'_{i-1}))^2 - (S(x_{i-1}))^2\right)$, which corresponds to checking γ^{-1} different input-output differences for the $(i+1)$-th S-box, and we expect to get one valid difference transition since each random input-output difference of the S-box holds with probability γ, as well as the corresponding γ^{-1} solutions of x_i satisfying this valid difference transition. Hence, if $\Delta x_0 \to \Delta y_0$ is valid, we expect to have one solution of $(x_0, \ldots, x_{s-2})$ and γ^{-1} solutions of x_{s-1} satisfying the first s equations in Eq 22.

Then, for each solution of x_{s-1}, we need to check Eq. 24, which holds with probability p^{-1}. In other words, after trying all x_{s-1}, Eq. 24 holds with probability $\gamma^{-1}p^{-1}$.

Hence, for a valid $(\Delta x_0, \Delta y_0)$, the probability that $(\Delta x_0, \ldots, \Delta x_s) \to (\Delta y_0, \ldots, \Delta y_s)$ is a valid difference transition for a randomly given $(\Delta x_1, \ldots, \Delta x_s, \Delta y_1, \ldots, \Delta y_s)$ is estimated as $\gamma^{-1}p^{-1}$. The time complexity to determine its validity is $\mathcal{O}(\gamma^{-1})$, i.e., at each layer, we perform on average $\mathcal{O}(\gamma^{-1})$ tests.

The above analysis indicates that after trying $\gamma \cdot p$ random $(\Delta x_0, \ldots, \Delta x_s, \Delta y_0, \ldots, \Delta y_0)$ with the above algorithmic procedure where $(\Delta x_0, \Delta y_0)$ is always set as valid, we expect to obtain one valid difference transition $(\Delta x_0, \ldots, \Delta x_s) \to (\Delta y_0, \ldots, \Delta y_s)$ as well as a solution of $(x_0, \ldots, x_{s-1})$ satisfying the first $s+1$ equations in Eq. 22. The time complexity is then $\mathcal{O}(\gamma^{-1} \cdot \gamma \cdot p) = \mathcal{O}(p)$.

7.2 Application to 2-Round Collision Attacks

Abusing notation, for the 2-round Monolith permutation, let the input and output of $\mathcal{F} \circ \mathcal{B}$ at the first round be $x \in \mathbb{F}_p^n$ and $y \in \mathbb{F}_p^n$, respectively. Let the difference be $\Delta x \in \mathbb{F}_p^n$ and $\Delta y \in \mathbb{F}_p^n$, respectively.

In our 2-round collision attacks, we consider the following input and output difference of the first nonlinear layer:

$$\Delta x = (\Delta x_0, \ldots, \Delta x_{s-1}, 0, \ldots, 0, \Delta x_{n-1}),\ \Delta y = (\Delta y_0, \ldots, \Delta y_{s-1}, \Delta y_s, 0, \ldots, 0, \Delta y_{n-1}).$$

In this case, by definition of $\mathcal{F} \circ \mathcal{B}$, we have $\Delta y_{n-1} = \Delta x_{n-1}$.

We use a similar two-stage method to construct a collision for 2-round Monolith. Specifically, we first aim to find the above $(\Delta x, \Delta y)$ such that there is no difference in the inner part and that the input differences of all s S-boxes at the 2nd round are zero.

Stage 1: In both Monolith-64 and Monolith-31, we observe that $n - r = c = s$. To ensure no difference in the inner part at the beginning, we can therefore set up s linear equations in $s + 1$ variables $(\Delta x_0, \ldots, \Delta x_{s-1}, \Delta x_{n-1})$, i.e., $(M^{-1}(\Delta x))_i = 0$ for $r \leq i < n$.

To ensure zero input difference for each S-box at the 2nd round, we similarly set up s linear equations in $s + 2$ variables $(\Delta y_0, \ldots, \Delta y_s, \Delta y_{n-1})$, i.e., $(M(\Delta y))_i = 0$ for $0 \leq i < s$.

Our goal is to find a valid solution of these variables as well as $(x_0, \ldots, x_{s-1})$ to ensure this difference transition.

For this purpose, we can randomly pick $(x_0, \Delta x_0)$, and compute the corresponding $\Delta y_0 = S(x_0 + \Delta x_0) - S(x_0)$. As Δx_0 is fixed, we are left with s linear equations in s variables $(\Delta x_1, \ldots, \Delta x_{s-1}, \Delta x_{n-1})$. Hence, we obtain one solution of $(\Delta x_1, \ldots, \Delta x_{s-1}, \Delta x_{n-1})$. As $(\Delta y_0, \Delta y_{n-1} = \Delta x_{n-1})$ are fixed, we are left with s linear equations in s variables $(\Delta y_1, \ldots, \Delta y_{s-1}, \Delta y_s)$. Hence, we obtain one solution of $(\Delta y_1, \ldots, \Delta y_{s-1}, \Delta y_s)$. Then, we determine whether $(\Delta x_0, \ldots, \Delta x_{s-1}, 0) \rightarrow (\Delta y_0, \ldots, \Delta y_s)$ correspond to a valid difference transition with the algorithmic procedure stated above.

As $(\Delta x_0, \Delta y_0)$ is always valid by construction, according to the above analysis, we expect to find such a solution of $(\Delta x, \Delta y)$ as well as the conforming inputs $(x_0, \ldots, x_{s-1})$ with time complexity $\mathcal{O}(p)$.

Experimental verification. To demonstrate the correctness of our algorithm and the estimation of the time complexity, we have run practical experiments using 120 threads to find such a valid $(\Delta x, \Delta y)$ for Monolith-31, as shown below (in hex). In less than 2 minutes, we could find a solution. In particular, we have counted the number of tests at each layer (totally 8 layers) in such a depth-first search. It is found that about 2^{22} tests have been performed at each layer, and this shows that the estimated time complexity $\mathcal{O}(p)$ is reasonable since $120 \times 2^{22} \approx 2^{29}$.

$$\begin{aligned}
x_0 &= \texttt{69737ce4}, & x_1 &= \texttt{1c495c53}, & x_2 &= \texttt{7ded143b}, & x_3 &= \texttt{4b7d0d09},\\
x_4 &= \texttt{584797fd}, & x_5 &= \texttt{4366d3ea}, & x_6 &= \texttt{5ed23b19}, & x_7 &= \texttt{4993e839}.\\
x'_0 &= \texttt{4a9f18c5}, & x'_1 &= \texttt{19e1eaeb}, & x'_2 &= \texttt{69557eae}, & x'_3 &= \texttt{1da62a4d},\\
x'_4 &= \texttt{6d308da5}, & x'_5 &= \texttt{73fd2ecd}, & x'_6 &= \texttt{5393fed7}, & x'_7 &= \texttt{19299394}.\\
\Delta y_0 &= \texttt{25cab5b6}, & \Delta y_1 &= \texttt{3695939b}, & \Delta y_2 &= \texttt{797a8566}, & \Delta y_3 &= \texttt{2b2c2278},\\
\Delta y_4 &= \texttt{66187ae6}, & \Delta y_5 &= \texttt{0bcd5cf2}, & \Delta y_6 &= \texttt{74bd3f39}, & \Delta y_7 &= \texttt{68d8749a},\\
\Delta y_8 &= \texttt{774096b5}, & \Delta y_{23} &= \texttt{267acfcc}, & \Delta x_{23} &= \texttt{267acfcc}.
\end{aligned}$$

Stage 2: $(\Delta x, \Delta y)$ have been fixed, where $\Delta x_i = 0$ for $s \leq i < n - 1$ and Δx_{n-1} is now a constant. In addition, as $(x_0, \ldots, x_{s-1})$ are also fixed to ensure $\Delta x \rightarrow \Delta y$, we can set $(x_s, \ldots, x_{n-1})$ as $n - s$ free variables. With these variables, we can first set up s linear equations to match the inner part. In addition, we can set up c quadratic equations in these variables to ensure there is no difference in the last c state words after the 2-round permutation. Note that these c equations are of degree 2 rather 3. The reason is similar as in the attacks on Tip5. Specifically, the input difference of $\mathcal{F}(\cdot)$ at the 2nd round is fixed according to Δy and $\mathcal{F}(\cdot)$ is quadratic, and hence the output difference of $\mathcal{F}(\cdot)$ at the 2nd round becomes linear in its input, which is quadratic in x.

In both Monolith-31 and Monolith-64, we have $d = c = s$ and $n = 3s$. Hence, we set up s linear equations, and s quadratic equations in $n - s = 2s$ variables. Solving this equation system is equivalent to solving s quadratic equations in s variables, and we can expect one solution.

Complexity evaluation. As stated above, the time complexity of Stage 1 is $\mathcal{O}(p)$. For Stage 2, we need to solve s quadratic equations in s variables. Under the regularity assumption, the time complexity is estimated as $\max\left\{\binom{1+2s}{s}^{\omega}, s \cdot 2^{3s}\right\}$ field operations. Hence, for Monolith-31 where $s = 8$, the overall complexity is estimated as $2^{31} + \max\{2^{14.6\omega}, 2^{27}\}$. For Monolith-64 where $s = 4$, the overall complexity is estimated as $2^{64} + \max\{2^{7.0\omega}, 2^{14}\}$, which indicates that our new attack on 2-round Monolith-64 cannot outperform the collision attack independent of the S-box.

Experimental verification. Experiments have confirmed that the collision attack on 2-round Monolith-31 is practical, as shown in Table 7. It is found that the system behaves like a regular system and using $\omega = 2.37$ does not underestimate the complexity. A concrete colliding message pair is given in Table 8.

Table 7: Experimental results for the collision attack on 2-round Monolith-31. The running time in the columns "T_{GB}" and "T_{FGLM}" is in seconds.

Target	T_{GB}	d_{MAC}	d_{reg}	$\mathcal{C}_{\text{GB}}$ ($\omega = 2.37$)	T_{FGLM}	B	$d_{\mathcal{I}}$	$\mathcal{C}_{\text{FGLM}}$ ($\omega = 3$)	Memory (MB)
Monolith-31	2	9	9	$2^{14.6\omega}$ ($2^{34.6}$)	2	2^8	2^8	$2^{3.0+8.0\omega}$ ($2^{27.0}$)	32.09

7.3 Application to 3-Round SFS Collision Attack

We also briefly describe a SFS collision attack on 3-round Monolith-64. We use the same notation as in the above 2-round collision attacks. In the 3-round SFS collision attack, we set $(S(x_0), \ldots, S(x_{s-1}), x_s, \ldots, x_{n-1})$ and $(\Delta x_s, \ldots, \Delta x_{n-1})$ as $2n - s$ unknowns, while $(\Delta x_0, \ldots, \Delta x_{s-1})$ are fixed as all 0. The attack is specified below:

Step 1: Set $\Delta x_i = 0$ for $0 \leq i < s$.

Step 2: Randomly choose values for the input pairs of all the s S-boxes at the 2nd round, and denote them by $(x_i^{(2)}, x_i'^{(2)})$ where $0 \leq i < s$.

Step 3: Set $(S(x_0), \ldots, S(x_{s-1}), x_s, \ldots, x_{n-1})$ and $(\Delta x_s, \ldots, \Delta x_{n-1})$ as $2n - s = 5s$ unknowns.

Step 4: Set up $c = s$ linear equations in $(\Delta x_s, \ldots, \Delta x_{n-1})$ to ensure no difference in the inner part in the beginning of the permutation.

Step 5: Set up $s + s$ degree-2 equations in the $5s$ unknowns to ensure the pre-fixed input pairs $(x_i^{(2)}, x_i'^{(2)})$ of the S-boxes at the 2nd round can be matched.

Step 6: Set up s degree-4 equations in the $5s$ unknowns to ensure no difference in the s S-boxes at the 3rd round;

Step 7: Set up $c = s$ degree-8 equations in the $5s$ unknowns to ensure no difference in the inner part after 3-round permutation.

Step 8: Solve the $5s$ equations in $5s$ variables, and get a solution. If there is no solution, move to Step 2, and repeat.

Complexity analysis. Among the $5s$ equations, there are s linear equations. Hence, it is equivalent to solving $2s$ degree-2 equations, s degree-4 equations and s degree-8 equations in $4s$ variables. Under the regularity assumption, the time complexity of the SFS collision attack on 3-round Monolith-64 (where $s = 4$) then becomes

$$\max\left\{\binom{1+2s+3s+7s+4s}{4s}^{\omega}, 4s\cdot(2^{2s}\cdot 4^{s}\cdot 8^{s})^{3}\right\} = \max\{2^{49.2\omega}, 2^{88}\}.$$

Under $\omega = 2.37$, the time complexity of the attack is $2^{116.7}$.

Acknowledgements

This work was initiated during the group sessions of the 10th Asian Workshop on Symmetric Cryptography (ASK 2023) held in Guangzhou, China. We thank the reviewers of ToSC 2024 Issue 4 for insightful comments to improve the quality of this paper. In addition, we also thank Mohammad Mahzoun for some advice on MAGMA.

Fukang Liu has been supported by JSPS KAKENHI Grant Numbers JP22K21282, JP24K20733, as well as the commissioned research (No. JPJ012368C05801) by National Institute of Information and Communications Technology (NICT). Lorenzo Grassi has been supported by the Ethereum Foundation via the Academic Grant Program 2024, and by the German Research foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa. Hailun Yan has been supported by the National Key Research and Development Program of China (2022YFB2701900) and the National Natural Science Foundation of China (62202444). This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority. Finally, this work is partly supported by the European Union under the project Confidential6G with Grant agreement ID: 101096435.

References

[AAB$^+$20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. *IACR Trans. Symm. Cryptol.*, 2020(3):1–45, 2020. doi:10.13154/TOSC.V2020.I3.1-45.

[ABK$^+$23] Tomer Ashur, Amit Singh Bhati, Al Kindi, Mohammad Mahzoun, and Léo Perrin. XHash: Efficient STARK-friendly hash function. Cryptology ePrint Archive, Paper 2023/1045, 2023. URL: https://ia.cr/2023/1045.

[ACG$^+$19] Martin R. Albrecht, Carlos Cid, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, and Markus Schofnegger. Algebraic cryptanalysis of STARK-friendly designs: Application to MARVELlous and MiMC. In *ASIACRYPT 2019*, volume 11923 of *LNCS*, pages 371–397. Springer, 2019. doi:10.1007/978-3-030-34618-8_13.

[AD18] Tomer Ashur and Siemen Dhooghe. MARVELlous: a STARK-friendly family of cryptographic primitives. Cryptology ePrint Archive, Paper 2018/1098, 2018. URL: https://ia.cr/2018/1098.

[AGR$^+$16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 191–219, 2016. doi:10.1007/978-3-662-53887-6_7.

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Pierre and Marie Curie University, Paris, France, 2004. URL: https://theses.hal.science/tel-00449609.

[BBC$^+$23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New design techniques for efficient arithmetization-oriented hash functions: Anemoi permutations and Jive compression mode. In *CRYPTO 2023*, volume 14083 of *LNCS*, pages 507–539. Springer, 2023. doi:10.1007/978-3-031-38548-3_17.

[BBL$^+$24] Augustin Bariant, Aurélien Boeuf, Axel Lemoine, Irati Manterola Ayala, Morten Øygarden, Léo Perrin, and Håvard Raddum. The algebraic FreeLunch: Efficient Gröbner basis attacks against arithmetization-oriented primitives. In *CRYPTO 2024*, volume 14923 of *LNCS*, pages 139–173. Springer, 2024. doi:10.1007/978-3-031-68385-5_5.

[BBLP22] Augustin Bariant, Clémence Bouvier, Gaëtan Leurent, and Léo Perrin. Algebraic attacks against some arithmetization-oriented primitives. *IACR Trans. Symm. Cryptol.*, 2022(3):73–101, 2022. doi:10.46586/TOSC.V2022.I3.73-101.

[BBVY21] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New attacks on LowMC instances with a single plaintext/ciphertext pair. In *ASIACRYPT 2021*, volume 13090 of *LNCS*, pages 303–331, 2021. doi:10.1007/978-3-030-92062-3_11.

[BCD$^+$20] Tim Beyne, Anne Canteaut, Itai Dinur, Maria Eichlseder, Gregor Leander, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Yu Sasaki, Yosuke Todo, and Friedrich Wiemer. Out of oddity - new cryptanalytic techniques against symmetric primitives optimized for integrity proof systems. In *CRYPTO 2020*, volume 12172 of *LNCS*, pages 299–328. Springer, 2020. doi:10.1007/978-3-030-56877-1_11.

[BDPA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the Sponge construction. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008. doi:10.1007/978-3-540-78967-3_11.

[BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.*, 3(3):177–197, 2009. doi:10.1515/JMC.2009.009.

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 Gröbner basis algorithm. *J. Symb. Comput.*, 70:49–70, 2015. doi:10.1016/J.JSC.2014.09.025.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. URL: https://ia.cr/2018/046.

[BSGL20] Eli Ben-Sasson, Lior Goldberg, and David Levit. STARK friendly hash – survey and recommendation. Cryptology ePrint Archive, Paper 2020/948, 2020. URL: https://ia.cr/2020/948.

[Buc76] Bruno Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.*, 10(3):19–29, 1976. doi:10.1145/1088216.1088219.

[CHWW22] Jiamin Cui, Kai Hu, Meiqin Wang, and Puwen Wei. On the field-based division property: Applications to MiMC, Feistel MiMC and GMiMC. In *ASIACRYPT 2022*, volume 13793 of *LNCS*, pages 241–270. Springer, 2022. doi:10.1007/978-3-031-22969-5_9.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms.* Springer, 4 edition, 2015. doi:10.1007/978-3-319-16721-3.

[Din21] Itai Dinur. Cryptanalytic applications of the polynomial method for solving multivariate equation systems over GF(2). In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021*, volume 12696 of *LNCS*, pages 374–403. Springer, 2021. doi:10.1007/978-3-030-77870-5_14.

[EGL$^+$20] Maria Eichlseder, Lorenzo Grassi, Reinhard Lüftenegger, Morten Øygarden, Christian Rechberger, Markus Schofnegger, and Qingju Wang. An algebraic attack on ciphers with low-degree round functions: Application to full MiMC. In *ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 477–506. Springer, 2020. doi:10.1007/978-3-030-64837-4_16.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1):61–88, 1999. doi:10.1016/S0022-4049(99)00005-5.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *ISSAC 2002*, page 75–83. ACM, 2002. doi:10.1145/780506.780516.

[FBS04] Jean-Charles Faugère, Magali Bardet, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *PICPSS 2004*, page 71–74, 2004.

[FGHR14] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. Sub-cubic change of ordering for Gröbner basis: a probabilistic approach. In *ISSAC 2014*, pages 170–177. ACM, 2014. doi:10.1145/2608628.2608669.

[FGLM93] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993. doi:10.1006/JSCO.1993.1051.

[GAH+23] Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, Morten Øygarden, Håvard Raddum, and Qingju Wang. Cryptanalysis of symmetric primitives over rings and a key recovery attack on Rubato. In *CRYPTO 2023*, volume 14083 of *LNCS*, pages 305–339. Springer, 2023. doi:10.1007/978-3-031-38548-3_11.

[GBJR23] Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard. Cryptanalysis of Elisabeth-4. In *ASIACRYPT 2023*, volume 14440 of *LNCS*, pages 256–284. Springer, 2023. doi:10.1007/978-981-99-8727-6_9.

[GHR+23] Lorenzo Grassi, Yonglin Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. Horst meets Fluid-SPN: Griffin for zero-knowledge applications. In *CRYPTO 2023*, volume 14083 of *LNCS*, pages 573–606. Springer, 2023. doi:10.1007/978-3-031-38548-3_19.

[GKL+22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced Concrete: A fast hash function for verifiable computation. In *CCS 2022*, pages 1323–1335. ACM, 2022. doi:10.1145/3548606.3560686.

[GKL+24] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Monolith: Circuit-friendly hash functions with new nonlinear layers for fast and constant-time implementations. *IACR Trans. Symm. Cryptol.*, 2024(3):44–83, 2024. doi:10.46586/TOSC.V2024.I3.44-83.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*, pages 519–535. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi.

[GKRS22] Lorenzo Grassi, Dmitry Khovratovich, Sondre Rønjom, and Markus Schofnegger. The Legendre symbol and the Modulo-2 operator in symmetric schemes over $\mathbb{F}_p^n$: Preimage attack on full Grendel. *IACR Trans. Symm. Cryptol.*, 2022(1):5–37, 2022. doi:10.46586/TOSC.V2022.I1.5-37.

[GOPS22] Lorenzo Grassi, Silvia Onofri, Marco Pedicini, and Luca Sozzi. Invertible quadratic non-linear layers for MPC-/FHE-/ZK-friendly schemes over $\mathbb{F}_p^n$ – Application to Poseidon. *IACR Trans. Symm. Cryptol.*, 2022(3):20–72, 2022. doi:10.46586/TOSC.V2022.I3.20-72.

[GRS21] Lorenzo Grassi, Christian Rechberger, and Markus Schofnegger. Proving resistance against infinitely long subspace trails: How to choose the linear layer. *IACR Trans. Symm. Cryptol.*, 2021(2):314–352, 2021. doi:10.46586/TOSC.V2021.I2.314-352.

[KBM23] Dmitry Khovratovich, Mario Marhuenda Beltrán, and Bart Mennink. Generic security of the SAFE API and its applications. In *ASIACRYPT 2023*, volume 14445 of *LNCS*, pages 301–327. Springer, 2023. doi:10.1007/978-981-99-8742-9_10.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC 1992*, pages 723–732. ACM, 1992. doi:10.1145/129712.129782.

[KLR24] Katharina Koschatko, Reinhard Lüftenegger, and Christian Rechberger. Exploring the six worlds of Gröbner basis cryptanalysis: Application to Anemoi. Cryptology ePrint Archive, Paper 2024/250, 2024. URL: https://ia.cr/2024/250.

[KR21] Nathan Keller and Asaf Rosemarin. Mind the middle layer: The HADES design strategy revisited. In *EUROCRYPT 2021*, volume 12697 of *LNCS*, pages 35–63. Springer, 2021. doi:10.1007/978-3-030-77886-6_2.

[LAW$^+$23] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. Coefficient grouping: Breaking Chaghri and more. In *EUROCRYPT 2023*, volume 14007 of *LNCS*, pages 287–317. Springer, 2023. doi:10.1007/978-3-031-30634-1_10.

[LKSM24] Fukang Liu, Abul Kalam, Santanu Sarkar, and Willi Meier. Algebraic attack on FHE-friendly cipher HERA using multiple collisions. *IACR Trans. Symm. Cryptol.*, 2024(1):214–233, 2024. doi:10.46586/TOSC.V2024.I1.214-233.

[LMØM23] Fukang Liu, Mohammad Mahzoun, Morten Øygarden, and Willi Meier. Algebraic attacks on RAIN and AIM using equivalent representations. *IACR Trans. Symm. Cryptol.*, 2023(4):166–186, 2023. doi:10.46586/TOSC.V2023.I4.166-186.

[LSMI21] Fukang Liu, Santanu Sarkar, Willi Meier, and Takanori Isobe. Algebraic attacks on Rasta and Dasta using low-degree equations. In *ASIACRYPT 2021*, volume 13090 of *LNCS*, pages 214–240. Springer, 2021. doi:10.1007/978-3-030-92062-3_8.

[LSW$^+$22] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-the-middle attack on LowMC. In *ASIACRYPT 2022*, volume 13791 of *LNCS*, pages 225–255. Springer, 2022. doi:10.1007/978-3-031-22963-3_8.

[Nyb91] Kaisa Nyberg. Perfect nonlinear S-boxes. In Donald W. Davies, editor, *EUROCRYPT 1991*, volume 547 of *LNCS*, pages 378–386. Springer, 1991. doi:10.1007/3-540-46416-6_32.

[RST23] Arnab Roy, Matthias Johann Steiner, and Stefano Trevisani. Arion: Arithmetization-oriented permutation and hashing from generalized triangular dynamical systems. *CoRR*, abs/2303.04639, 2023. doi:10.48550/ARXIV.2303.04639.

[Sal23] Robin Salen. Two additional instantiations from the Tip5 hash function construction, 2023. URL: https://toposware.com/paper_tip5.pdf.

[SLS$^+$23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, Bobbin Threadbare, and Al-Kindi. The tip5 hash function for recursive STARKs. Cryptology ePrint Archive, Paper 2023/107, 2023. URL: https://ia.cr/2023/107.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, 2008. doi:10.1007/978-3-540-78524-8_1.

[YZY⁺24] Hongsen Yang, Qun-Xiong Zheng, Jing Yang, Quanfeng Liu, and Deng Tang. A new security evaluation method based on resultant for arithmetic-oriented algorithms. In *ASIACRYPT 2024*, LNCS. Springer, 2024. URL: https://ia.cr/2024/886.

[ZWY⁺23] Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic attacks on round-reduced Rain and full AIM-III. In *ASIACRYPT 2023*, volume 14440 of *LNCS*, pages 285–310. Springer, 2023. doi:10.1007/978-981-99-8727-6_10.

Supporting Material

Contents

Table 8: Experimental results forTip5 family and Monolith family, where we denote the *first* d output words after the permutation by Out_f^d, and the *last* c output words after the permutation by Out_l^c.

	near collision for 3-round Tip4 (Sect. 5.3, Table 5)
IV	1 1 1 1
m_0	c6302a9416e1e7b3 191740c2110b592d 12b397fb06e33789 29f060477c46d55f 16994cb4ce4f0cc8 076bdd4505177614 44ffcea890f7f274 295542230aec227e 6cc6658831bc4f72 2cc445c05ac6250c 4721e7d54ca12911 a487829e83cdc2a9
m_0'	1c0032b4ce793a22 721d6b79e9cfb508 c8b06bb2c12f1015 a3458bf3e71280c6 a3458bf3e71280c6 4d3744b090c37448 2425f22fe08cb9d4 830f280903acb5ef 9b62df4108d31f9c b63b522d993f6526 3ed2820678c991ed 06d101604089fa5c
Out_f^2	ea4442b2c6cde046 94587130d50f9581
	SFS collision for 3-round Tip5 (Sect. 5.4, Table 6)
IV	63b5be0187fa05b0 8f4db2bfcabd2302 0a5c6021ee35d565 11f40c2b7c8d452b bce318baebb56f57 e98b25e991ed9829
m_0	0510053a63393b5e 070b433153c77856 a9df7ece7c71c3b6 dea507128a64faa1 a567a4cd0d87dbc1 e4e0df5a55148b11 189548e9e2d5578e f021f746d17c1717 3c5e5bdf6e27f570 f54e8c4543ed13c8
m_0'	ac95d195f2bd7273 967bebc4a6922bf0 4447e4bfc03b125f a8c7a57ed55a84ac 76fda4ee213a7dc1 de3f65a907005d3f ee35e7d8c8ae4ed0 910d60b34fad5797 c9821c0110240935 201466ca5a0de071
Out_f^5	acb46c393decc891 e3abad309c016a4d 2ffb1a0ba6264680 4ba83933b09c17b5 6fabfd5b7b5ac0e3
	SFS collision for 3-round Tip4 (Sect. 5.4, Table 6)
IV	05455d646a9fb498 5b98a5cdcbf75625 36611580308009e5 f13961bc0bf79a35
m_0	f065c68799b9224f 5f6be20c98289f24 2a35d4c999731105 2f336da9b12d8e3b d5bf22405d1d1c6e 62ea5eb72ef241af 71dbff1985f169c4 398e9ff03a7e62a8 6ab41b1ef7261fee ae33d6bed1d6af17 49f320563c9ad5fa 91f85d88dd643c90
m_0'	f99251c7d59303fc 0244d3ac998244b2 5071809bfcedd711 d2587fef54685ba9 bf83eeefc8aafcd7 b2a6864b9ead20c0 e6299df48ae58aba 1fb4938ab0faf420 53d74eab02fe04a9 f690f3967d1c28c8 e0e56a7b656bda5d 8a57287029e36c73
Out_f^4	199feda6f6577448 73977eac80d5f0db 7864fb871040f78a 08ac2e5ec2d03347
	SFS collision for 3-round Tip4' (Sect. 5.4, Table 6)
IV	84aefeb85e1c736a 99852f9ac85c8be0 aa2ccabaf5e705e1 6206827be78f9cd2
m_0	f9e8311f95cb2b77 4835b953ea03ed41 0b47521c88b06001 ac8436d8c449aa10 4ae194055c249e8e 41ffa5ff7fc583ca dd37c9a863fe195b ad014cca15abdc41
m_0'	abf3f56fbab88503 2f73090e75f87967 1144a30cef40aee0 c975f46f94e8a65e 22b84ee8dd304de5 b37f710c695a5ea3 bc57a1d2c071fd0a 16604e2c176495ae
Out_f^4	4f885efd35b10b46 3ce80cda16752fc8 93a563d23acb5028 a2cc2fdf2b4b0d6f
	collision for 2-round Monolith-31 (Sect. 7.2, Table 7)
IV	0 0 0 0 0 0 0 0
m_0	29c00f10 3b7173a8 79ea0cd6 0d55fe73 04b4aaef 578ce0fd 1945b9f5 5178fbea 6d2b218c 7942abf3 583952bf 6185fe9f 5339a47f 233560f3 0010614c 5b831c47
m_0'	622558a9 3751266d 7586a6a1 6190bb51 43728d5a 43d555ae 299fa6fa 55327b56 6ba000e3 12f56ade 4c930fde 5b16a551 5bd241bf 4fc1844e 47adf609 2c7a8474
Out_l^8	7c957753 09b10941 0bdfbfee 51f1a150 56e2ccfb 31650e10 5855422f 67e6d6af
	collision for 2-round Monolith-64 (Sect. 3.2, Table 4)
IV	0 0 0 0
m_0	0dafb1f39af4f126 123c636bf95605db 20d9d4075c10f47c 9a1a6496a3fd807a 44f906f66d366299 db85d71b6755757f baba90fecb346771 5080e621266de94d
m_0'	2855a6c059af53d7 6c41bcca84872039 548baaf10ba5cd31 8ee31cb3244efa57 a4df4c6bf38a6728 0aaaacfc373fe34a 902346c472ac3c2e 5b865d4a1a3b69b4
Out_l^4	000cda713a1183c5 881fafb16f7a36a5 7f3ce13953660b23 d018d5dde86dcace