IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2024EAP1117

Publicized:2025/01/21



A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

# Improved Upper Bound of Algebraic Degrees for Some Arithmetization-Oriented Ciphers

**Jianqiang NI**[†], **Gaoli WANG**[†a)], **Yingxin LI**[†], ***and*** **Siwei SUN**[††,†††], ***Nonmembers***

**SUMMARY** Recently, the practical applications of advanced cryptographic protocols, such as Multi-Party Computation (MPC), Fully Homomorphic Encryption (FHE), and Zero Knowledge Proofs (ZKP), have spurred the development of a series of new symmetric encryption primitives. These novel symmetric encryption primitives, referred to as *Arithmetization-Oriented* (AO) ciphers, aim to minimize the number of field multiplications in large finite fields, including $\mathbb{F}_{2^n}$ or $\mathbb{F}_p$. In order to evaluate the algebraic degrees of AO ciphers over $\mathbb{F}_{2^n}$, the *general monomial prediction* technique was proposed by Cui et al. at ASIACRYPT 2022. However, when using their searching tool to evaluate the algebraic degrees of AO ciphers with complex affine layers, the efficiency is low, preventing solutions within a predetermined timeframe. In this study, we extend the propagation rules of monomials for field-based operations and present an automatic searching tool based on Mixed Integer Linear Programming(MILP) and Boolean Satisfiability(SAT) Problem for evaluating the upper bound of the algebraic degrees. Moreover, to accurately calculate the algebraic degrees of monomials in the SAT model, we improve the sequence encoding method, enabling it to accurately determine whether the monomials of degree $d$ exist in the output. We apply our new searching tool to various AO ciphers, including **Chaghri**, **MiMC**, and **Ciminion**. For **Chaghri**, we compare our results with the Coefficient Grouping technique proposed by Liu et al. at EUROCRYPT 2023, and our results yield tighter upper bounds compared to Liu et al.'s findings. Additionally, we evaluate the algebraic degrees of **Chaghri** and **MiMC** with arbitrary complex affine layers and obtain tighter bounds compared to the results from Liu et al. at CRYPTO 2023. Regarding **Ciminion**, we have observed that starting from the 4-th round, the upper bound on the algebraic degrees for each round actually 1 degree lower than the previous bound. Our searching tool enables a more precise evaluation of the algebraic degrees of AO ciphers, contributing to a deeper understanding of the design and analysis of such primitives.

***key words:*** *Degree evaluation, Arithmetization-oriented ciphers, General monomial prediction, MILP, SAT*

## 1. Introduction

In recent years, the popularity of advanced encryption protocols such as Multi-Party Computation (MPC), Fully Homomorphic Encryption (FHE), and Zero Knowledge Proofs (ZKP) has led to new demands for the design of symmetric cryptographic primitives [1]–[9]. The main reason is that traditional symmetric cryptographic primitives, such as AES and SHA-2/SHA-3, are not efficient enough in these protocols. The role of symmetric cryptographic primitives in these protocols is mainly to ensure the integrity of computations. However, the fundamental operations used by these protocols are not directly equivalent to standard CPU instructions, such as bitwise AND or rotations, or conventional hardware components like wires that are commonly employed as foundational elements. Instead, implementers primarily rely on finite field operations conducted on a large field $\mathbb{F}_q$, where the size $q$ typically exceeds $2^{64}$. Frequently, $q$ is selected as a prime number or a power of 2. These innovative primitives, known as *Arithmetization-Oriented* (AO) ciphers, aim to minimize the number of multiplications or the multiplication depth within these primitives, following the design metric commonly used in most protocols. The design of AO ciphers significantly differs from that of traditional symmetric cryptographic primitives. To minimize the number of multiplications, AO ciphers typically employ low-degree invertible functions as their nonlinear layers, such as $x \mapsto x^d$, where $\gcd(d, p-1) = 1$ or $\gcd(d, 2^n - 1) = 1$. Various examples of these ciphers include **MiMC** [1], Jarvis/Friday [10], GMiMC [11], HadesMiMC [12], Rescue [3], Poseidon [13], Anemoi [7] and **Chaghri** [6] etc.

The characteristic of these ciphers significantly influences the potential attacks against them. While traditional symmetric cryptographic primitives are susceptible to statistical attacks, such as differential [14] and linear [15] attacks, these new primitives are particularly vulnerable to algebraic attacks. Therefore, the overall security of these ciphers is often determined by algebraic attacks, specifically Gröbner basis attack, higher-order differential attacks [16], [17] and interpolation attacks [18] etc. For instance, in ASIACRYPT 2019, the AO cipher Jarvis was found to be vulnerable to Gröbner basis attack [2]. In ASIACRYPT 2020, Eichlseder et al.[19] introduced novel upper bounds on the algebraic degrees for AO ciphers over $\mathbb{F}_{2^n}$. They observed that the algebraic degrees growth of **MiMC** is linear rather than exponential and successfully mounted higher-order differential attacks against full-round **MiMC**. In essence, the algebraic degree stands as a critical security attribute for AO ciphers, emphasizing the need for the development of new tools to evaluate their algebraic degree.

**Related work.** Higher-order differential attacks are capable of exploiting the insufficient growth of algebraic degrees of cryptographic primitives to launch attacks. Given a function $F$ with algebraic degree $\deg(F)$ over $\mathbb{F}_2^n$, a higher-order differential attack is launched by computing the sum of $F$ over all affine subspaces $V \subseteq F_2^n$ and finding $\sum_{x \in V} F = 0$ where

[†]Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, Shanghai, China

[††]School of Cryptology, University of Chinese Academy of Sciences, Beijing, China

[†††]State Key Laboratory of Cryptology, Beijing, China

a) E-mail: glwang@sei.ecnu.edu.cn

the dimension of $V$ is greater than or equal to $\deg(F) + 1$. This property was first discovered by Lai [16] in the context of traditional symmetric cryptography. In [19], Eichlseder et al. redefined higher-order differential attacks for AO ciphers based on a finite field $\mathbb{F}_{2^n}$. It revealed the linear growth of the algebraic degree of **MiMC**. The research conducted in [20] involves an analysis of the algebraic degree pertaining to SPN (Substitution-Permutation Network) ciphers over $\mathbb{F}_{2^n}^m$.

Additionally, in [21], the algebraic degrees growth of **MiMC** defined over $\mathbb{F}_{2^n}$ was studied, and they determined the exact algebraic degree of **MiMC**. In [22], the monomial prediction [23] was extended to $\mathbb{F}_{2^n}$, referred to as *general monomial prediction*(GMP). The authors employed the bit-vector theory of Satisfiability Modulo Theories (SMT) to model general monomial prediction and evaluate the algebraic degrees of **MiMC**, Feistel MiMC, and GMiMC, leading to upper tighter bounds on their algebraic degrees. In an effort to analyze a broader spectrum of AO ciphers over $\mathbb{F}_p$, Beyne et al. extended the integral attack to large finite fields of any characteristic at CRYPTO 2020 [24].

To gain a deeper understanding of the growth of algebraic degree in AO ciphers over $\mathbb{F}_{2^n}$, Liu et al. introduced an innovative technique known as the coefficient grouping technique [25] at EUROCRYPT 2023. This innovative technique simplifies the evaluation of algebraic degrees by transforming them into a well-structured optimization problem. The authors applied this technique to launch a high-order differential attack on **Chaghri**, an FHE-friendly block cipher [6]. As a result, they successfully broke the original **Chaghri** and proposed a new affine layer, leading to an exponential increase in the algebraic degree of **Chaghri**. At CRYPTO 2023, Liu et al. [26] studied the algebraic degree growth of SPN-based AO ciphers over $\mathbb{F}_{2^n}^t$ with complex affine layer. Based on the original coefficient grouping technique, Liu et al. proposed a variant technique that effectively identifies affine layers leading to exponential growth of the algebraic degree. Additionally, it enables efficient computation of the upper bound of the algebraic degrees for arbitrary affine layers. However, the variant technique employed relaxed constraints, thereby sacrificing the accuracy in evaluating the upper bound of the algebraic degree to some extent.

**Our contributions.** In this paper, we show how to model the general monomial prediction as both a MILP(Mixed Integer Linear Programming) problem and an SAT (Satisfiability) problem. Additionally, to accurately compute the algebraic degree of monomials within the SAT model, we enhance the sequential encoding method [27], facilitating precise determination of the presence of monomials of degree $d$ in the output. Building upon this, we propose an automatic searching tool for evaluating the upper bound on the algebraic degree. We apply our new searching tool to several significant AO ciphers, including **Chaghri**, **MiMC**, and **Ciminion**. The source codes of our model are available at `https://github.com/minionsjay/GMP_SAT`. For the AO ciphers with affine layers, we define the affine layer as $x \mapsto c_0 + \sum_{i=1}^{w} c_i x^{2^{h_i}}$, where $c_1, \ldots, c_w \in \mathbb{F}_{2^n} \backslash \{0\}$. We use AO cipher name-$(h_1, \ldots, h_w)$ to denote the AO ciphers with different affine layers, e.g. **Chaghri**-$(h_1, \ldots, h_w)$.

| $(n, d, h)$ | References | $r$ steps | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 27 | 28 |
| $(63, 32, 3)$ | [25] | 16 | 21 | 26 | 32 | 37 | 42 | 47 | 52 | 57 | 62 | 63 | - |
| | Section 4.1 | 16 | 21 | 26 | **31** | 37 | **41** | 47 | **51** | 57 | **61** | **62** | **63** |

**Table 1** The new algebraic degree upper bound of the original `Chaghri`.

- For the original **Chaghri**-(3), using our new searching tool, we can obtain a upper tighter bound on the algebraic degree in certain rounds compared to the coefficient grouping technique [25] proposed by Liu et al. Our results show that at 27 steps, we have an algebraic degree upper bound by 62, and at 28 steps, the upper bound is 63, while Liu et al.'s result is 63 at 27 steps, as depicted in Table 1. In **Chaghri**, there are two-step functions in one round. For more details, please refer to Section 4.1. This indicates that we can construct a higher-order differential distinguisher for one more step. Additionally, when solving with our SAT model using CaDiCaL[28] as the solver, the solving time is less than 10 seconds. However, when utilizing the model described in [22], the solving time often exceeds the predefined duration of 200 seconds when the number of rounds is greater than or equal to 6. The comparison of solving times for different models is presented in Table 2. For **Chaghri**-$(h_1, h_2)$ with $w = 2$ affine layers, we compare our results with Liu et al.'s new technique [26]. We perform degree evaluation for **Chaghri**-(0,3), **Chaghri**-(0,6), and **Chaghri**-(0,36). Similarly, we obtained upper tighter bounds for certain steps and achieved a higher number of steps at the critical point of degree evaluation. The upper tighter bounds for the algebraic degree obtained are shown in Table 3.

**Table 2** Comparison of times (seconds) required to solve for the upper bounds on algebraic degree across various models for **Chaghri**-$(63, 32, 3)$. The $\text{GMP}_{\text{SAT}}$, $\text{GMP}_{\text{MILP}}$, and $\text{GMP}_{\text{SMT}}^{[22]}$ represent the conversion of the general monomial prediction into different models. $*$ indicates that the solution time exceeded the preset time limit of 200 seconds.

| Methods | $r$ steps | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $\text{GMP}_{\text{SAT}}$ | 0.03 | 0.06 | 0.09 | 0.12 | 0.25 | 0.38 | 0.49 | 0.56 | 0.69 | 0.72 | 0.96 | 1.04 |
| $\text{GMP}_{\text{MILP}}$ | 0.01 | 0.83 | 0.02 | 0.04 | 0.08 | 0.14 | 0.56 | 0.86 | 1.55 | 3.25 | 6.32 | 15.0 |
| $\text{GMP}_{\text{SMT}}^{[22]}$ | 0.03 | 0.10 | 0.15 | 0.32 | 1.68 | * | * | * | * | * | * | * |
| Methods | $r$ steps | | | | | | | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | ... | 28 |
| $\text{GMP}_{\text{SAT}}$ | 1.34 | 1.38 | 2.44 | 1.93 | 3.23 | 2.90 | 3.84 | 3.68 | 5.22 | 5.37 | <10 | 5.62 |
| $\text{GMP}_{\text{MILP}}$ | 13.6 | 27.6 | 127 | * | * | * | * | * | * | * | * | * |
| $\text{GMP}_{\text{SMT}}^{[22]}$ | * | * | * | * | * | * | * | * | * | * | * | * |

- In comparison to the upper bounds on the algebraic

**Table 3** The new algebraic degree upper bound of the **Chaghri** with different affine layers.

| $(n, d, h_1, h_2)$ | References | $r$ steps | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\leq 3$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| (63,32,0,3) | [26] | $2^r$ | 15 | 27 | 42 | 52 | 62 | 63 | - |
| | Section 4.1 | $2^r$ | **14** | **24** | **36** | **46** | **55** | **62** | **63** |
| (63,32,0,6) | [26] | $2^r$ | 15 | 27 | 47 | 63 | - | - | - |
| | Section 4.1 | $2^r$ | **14** | **24** | **38** | **53** | **63** | - | - |
| (63,32,0,36) | [26] | $2^r$ | 15 | 27 | 48 | 63 | - | - | - |
| | Section 4.1 | $2^r$ | **14** | **24** | **39** | **59** | **62** | **63** | - |

degree of **MiMC** with complex affine layers provided by Liu et al. [26], we have also obtained more precise upper bounds. The more precise upper bounds for the algebraic degree obtained are shown in Table 4 of Section 4.1. For **Ciminion**, using our new searching tool to evaluate the algebraic degree, we find that the actual algebraic degree upper bound of **Ciminion** over the field $\mathbb{F}_{2^n}$ is lower than the previous result found in [29]. Here, we provide a new upper bound of the algebraic degree and can achieve one more round compared to the previous higher-order differential distinguisher. The details are presented in Table 5 of Section 4.2.

**Outline.** In Section 2, some notations and background are introduced. In Section 3, we show how to model general monomial prediction as both MILP and SAT problems, introducing an enhanced sequencetial encoding method within the SAT model to construct the objective function. Building on this, we propose a novel automatic tool for evaluating the algebraic degree. In Section 4, we apply the new tool to several AO ciphers to evaluate their upper bounds of algebraic degree. Finally, we conclude the paper and discuss some problems for further potential study in Section 5.

## 2. Preliminaries

### 2.1 Notations

The following notations will be used throughout the paper. $\mathbb{F}_2^n$ denote the $n$-dimensional vector space over the finite field $\mathbb{F}_2$. The notation $\mathbb{F}_{2^n}^t$ can be interpreted as a $t$-dimensional vector space over the field $\mathbb{F}_{2^n}$. To represent bit or word vectors, bold italic lowercase letters are used. For example, $\boldsymbol{u} \in \mathbb{F}_2^n$ represents the $n$-bit vector $(u_0, \ldots, u_{n-1})$, where the Hamming weight of $\boldsymbol{u}$ is denoted as $\mathrm{H}(\boldsymbol{u}) = \sum_{i=0}^{n-1} u_i$. Similarly, $\boldsymbol{u} \in \mathbb{F}_{2^n}^m$ represents the $m$-word vector $(u_0, \ldots, u_{m-1})$, where the Hamming weight of $\boldsymbol{u}$ is denoted as $\mathrm{H}(\boldsymbol{u}) = \sum_{i=0}^{n-1} \mathrm{H}(u_i)$. For any $x \in \mathbb{F}_{2^n}$, we have $x = \sum_{i=0}^{n-1} x_i \cdot 2^i$ for $x_i \in \mathbb{F}_2$ and $\mathrm{H}(x) = \sum_{i=0}^{n-1} x_i$.

We denote $\mathbf{0}^n$ as the $n$-dimensional zero vector, and $\boldsymbol{e}_i$ as the n-dimensional vector with a value of 1 at the $i$-th position and 0 elsewhere, where the elements belong to $\mathbb{F}_{2^n}$. The $a|b$ denotes $a$ divides $b$. For any $x, y \in \mathbb{F}_{2^n}$, if $x_i \geq y_i$ for all $i$, we denote it as $x \succeq y$. We denote addition (and subtraction) over $\mathbb{F}_2$ or $\mathbb{F}_{2^n}$ by the symbol $\oplus$.

**Higher-order differential attack over binary extension fields.** Higher-order differential attacks [16], [17] are significant cryptographic attack techniques that exploit the low algebraic degree of nonlinear transformations, like classical block ciphers. When the algebraic degree of a Boolean function is sufficiently low, this method can differentiate it from a random function. Let $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function with algebraic degree $d$. Then, by selecting an affine subspace of dimension $\geq d + 1$, the sum of the function $F$ over this subspace equals 0. In other words, for any affine subspace $\mathcal{V} \oplus c$ where the dimension is $\geq d + 1$ and $c \in \mathbb{F}_2^n$, we have:

$$\sum_{x \in \mathcal{V} \oplus c} F(x) = 0.$$

To launch higher-order differential attacks on AO ciphers over $\mathbb{F}_{2^n}$, it is crucial to gain a deeper understanding of the algebraic degree of their polynomial representations.

### 2.2 Algebraic Degree for Polynomials over Binary Extension Fields

To mount a higher-order differential attack, the first step is to find an upper bound of the algebraic degree of the polynomial representation of the output in terms of the input. In order to do this, in the following we heavily exploit the link between the algebraic degree of the functions over $\mathbb{F}_{2^n}$ and over $\mathbb{F}_2^n$. First, let us recall the two notions of degree that apply to a function over a finite field with characteristic 2.

**Definition 1** (ANF and Algebraic Degree): Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. The Algebraic Normal Form (ANF) of $f$ can be uniquely represented as a multivariate polynomial in $\mathbb{F}_2[x_0, \ldots, x_{n-1}]/\langle x_0^2 \oplus x_0, \ldots, x_{n-1}^2 \oplus x_{n-1} \rangle$. Its ANF is given as

$$f(\boldsymbol{x}) = f(x_0, \ldots, x_{n-1}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \cdot \pi_{\boldsymbol{u}}(\boldsymbol{x}),$$

where the coefficient $a_{\boldsymbol{u}} \in \mathbb{F}_2$ and $\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{n-1} x_i^{u_i}$. The algebraic degree of $f$ is

$$d_f = \max\{\mathrm{H}(\boldsymbol{u}) | \boldsymbol{u} \in \mathbb{F}_2^n, a_{\boldsymbol{u}} \neq 0\}.$$

If $\boldsymbol{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a vector Boolean function, then its algebraic degree is defined as the maximum algebraic degree of the coordinates of $\boldsymbol{f}$. Equivalently, $d_{\boldsymbol{f}} = \max\{d_{f_i} | 0 \leq i < m\}$.

**Proposition 1** ([30]): For any univariate function $F : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ as

$$F(x) = \sum_{i=0}^{2^n - 1} \phi_i \cdot x^i,$$

where $\phi_i \in \mathbb{F}_{2^n}$, the algebraic degree $d_F$ of $F$ as a vectorial Boolean function is the maximum Hamming weight of

exponents of non-vanishing monomials, that is

$$d_F = \max_{0 \le i \le 2^n-1} \{\mathrm{H}(i) | \phi_i \neq 0\}.$$

**The multivariate case.** Let $\boldsymbol{F}(x_0, \ldots, x_{t-1}) : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}$ be a function over $\mathbb{F}_{2^n}[x_0, \ldots, x_{t-1}]/\langle x_0^{2^n}+x_0, \ldots, x_{t-1}^{2^n}+x_{t-1}\rangle$. $\boldsymbol{F}$ can be uniquely represented as

$$\boldsymbol{F}(x_0, \ldots, x_{t-1}) = \sum_{\boldsymbol{u}=(u_0,\ldots,u_{t-1}) \in \mathbb{F}_{2^n}^t} \phi(\boldsymbol{u}) \cdot \pi_{\boldsymbol{u}}(\boldsymbol{x}),$$

where the coefficient $\phi(\boldsymbol{u}) \in \mathbb{F}_{2^n}$ and the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{t-1} x_i^{u_i}$.

If the coefficient of the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ in $\boldsymbol{F}$ is a non-zero constant, then we say that $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ appears in the expression of $\boldsymbol{F}$ and denote it as $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \to F$. If the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ does not appear in $\boldsymbol{F}$, we denote it as $\pi_{\boldsymbol{u}}(\boldsymbol{x}) \nrightarrow \boldsymbol{F}$. The algebraic degree is then defined as

$$d_{\boldsymbol{F}} = \max \left\{ \sum_{i=0}^{t-1} \mathrm{H}(u_i) : u_i \in [0, 2^n-1], \phi(\boldsymbol{u}) \neq 0 \right\}.$$

### 2.3 Monomial Prediction and General Monomial Prediction

#### 2.3.1 Monomial Prediction

The monomial prediction [23] is a novel method used to identify the presence of a specific monomial in the product of the coordinate functions of a vectorial Boolean function. This approach is particularly useful when directly constructing the product is computationally impractical. By computing the so-called monomial trail, the monomial prediction technique can determine whether a particular monomial appears in the output of a cryptographic function. This technique is equivalent to the three-subset bit-based division property without unknown subset [31].

**Definition 2** (Monomial Trail [23]): Let $\boldsymbol{x}^{(i)} = \boldsymbol{f}_i(\boldsymbol{x}^{(i-1)})$ for $1 \le i \le r$. A sequence of monomials $(\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}), \ldots, \pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}), \ldots, \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}))$ is a $r$-round monomial trail connecting $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ with respect to the composition function $\boldsymbol{f} = \boldsymbol{f}_{r-1}(\boldsymbol{x}^{(r-1)}) \circ \cdots \circ \boldsymbol{f}_0(\boldsymbol{x}^{(0)})$ if

$$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \cdots \to \pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}) \to \cdots \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}).$$

If there is at least one monomial trail from $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ to $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, we denote $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. Otherwise, $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. The set of all monomial trails from $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ to $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ are denoted by $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. If the size of the monomial trails is an odd number, we say that $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ indeed contains $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$, and we denote it as $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. Otherwise, we denote it as $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \nrightarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$.

#### 2.3.2 General Monomial Prediction

Let $\boldsymbol{y} = \boldsymbol{F}(\boldsymbol{x})$ be a function from $\mathbb{F}_{2^n}^t$ to $\mathbb{F}_{2^n}^s$, where $\boldsymbol{x} = (x_0, \ldots, x_{t-1}) \in \mathbb{F}_{2^n}^t$ and $\boldsymbol{y} = (y_0, \ldots, y_{s-1}) \in \mathbb{F}_{2^n}^s$. The general monomial prediction [22] is a novel technique designed to identify the presence of a specific monomial in the product of the coordinate functions of $\boldsymbol{F}$. It is especially useful when direct computation of the product is computationally impractical. This technique leverages the fact that a cryptographic composite function $\boldsymbol{F} : \mathbb{F}_{2^n}^{n_0} \to \mathbb{F}_{2^n}^{n_r}$ is typically a composition of several simpler functions $F_i : \mathbb{F}_{2^n}^{n_i} \to \mathbb{F}_{2^n}^{n_{i+1}}$, $0 \le i \le r-1$, such that $\boldsymbol{x}^{(i+1)} = F_i(\boldsymbol{x}^{(i)})$, i.e.,

$$\boldsymbol{F} = F_{r-1} \circ F_{r-2} \circ \cdots \circ F_0.$$

**Definition 3** (General Monomial Trail [22]): Let $F_i$ be a sequence of polynomials over $\mathbb{F}_{2^n}$ for $0 \le i < r$, and let $\boldsymbol{x}^{(i+1)} = F_i(\boldsymbol{x}^{(i)})$. A sequence of monomials $(\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}), \ldots, \pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}), \ldots, \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}))$ forms an $r$-round general monomial trail connecting $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ with respect to the composition function $\boldsymbol{F} = F_{r-1} \circ F_{r-2} \circ \cdots \circ F_0$ if

$$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \cdots \to \pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}) \to \cdots \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}).$$

If there is at least one general monomial trail connecting $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, we denote $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. Otherwise, $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. When $n = 1$, the general monomial trail is equivalent to the monomial trail.

**Lemma 1** ([22]): If $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, then $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$; or, equivalent $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ implies $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \nrightarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$.

Since the coefficients of the polynomials defined in $\mathbb{F}_{2^n}$ range from $[0, 2^n-1]$, a monomial will only be absent in the next state if its coefficient is zero or divisible by $2^n$. This is different from monomial prediction in $\mathbb{F}_2$ where coefficients are either 0 or 1.

## 3. Automatic Evaluation of Algebraic Degree Upper Bounds based on MILP and SAT

In this section, we first model the propagation rules of basic field-based operations using MILP and SAT methods. By appropriately setting the initial constraints and stopping rules, the problem of finding the upper bound on the algebraic degree of AO ciphers can be transformed into a MILP or SAT problem and efficiently solved.

The design structure of AO ciphers is consistent with traditional structures, where we denote the a key, plaintext, and ciphertext variables by $\boldsymbol{k} \in \mathbb{F}_{2^n}^k$, $\boldsymbol{x} \in \mathbb{F}_{2^n}^t$, and $\boldsymbol{y} \in \mathbb{F}_{2^n}^t$ respectively. Similar to the monomial prediction modelling framework for traditional block ciphers in [32], an AO block cipher $\boldsymbol{y} = E_{\boldsymbol{k}}(\boldsymbol{x})$ can be considered as a family of functions indexed by

$$E_{\boldsymbol{k}} : \mathbb{F}_{2^n}^t \times \mathbb{F}_{2^n}^k \to \mathbb{F}_{2^n}^t, (\boldsymbol{k}, \boldsymbol{x}) \mapsto \boldsymbol{y} = E_{\boldsymbol{k}}(\boldsymbol{x}).$$

Hence, any product of ciphertext words $\pi_{\boldsymbol{z}}(\boldsymbol{y})$ can be

expressed as a function $F(\boldsymbol{k}, \boldsymbol{x})$:

$$\begin{aligned}\pi_z(\boldsymbol{y}) &= \sum_{(\boldsymbol{u},\boldsymbol{v})\in\mathbb{F}_{2^n}^t\times\mathbb{F}_{2^n}^k} a_{\boldsymbol{u},\boldsymbol{v}}\cdot\pi_{\boldsymbol{v}}(\boldsymbol{k})\pi_{\boldsymbol{u}}(\boldsymbol{x})\\ &= \sum_{\boldsymbol{u}\in\mathbb{F}_{2^n}^t}\left(\sum_{\boldsymbol{v}\in\mathbb{F}_{2^n}^k} a_{\boldsymbol{u},\boldsymbol{v}}\cdot\pi_{\boldsymbol{v}}(\boldsymbol{k})\right)\cdot\pi_{\boldsymbol{u}}(\boldsymbol{x})\\ &= \sum_{\boldsymbol{u}\in\mathbb{F}_{2^n}^t} a_{\boldsymbol{u}}(\boldsymbol{k})\cdot\pi_{\boldsymbol{u}}(\boldsymbol{x}),\end{aligned}$$

where $k$ denotes the key size, $a_{\boldsymbol{u},\boldsymbol{v}} \in \mathbb{F}_{2^n}$, for all $(\boldsymbol{u},\boldsymbol{v}) \in \mathbb{F}_{2^n}^t \times \mathbb{F}_{2^n}^k$, and $a_{\boldsymbol{u}}(\boldsymbol{k}) = \sum_{\boldsymbol{v}\in\mathbb{F}_{2^n}^k} a_{\boldsymbol{u},\boldsymbol{v}}\cdot\pi_{\boldsymbol{v}}(\boldsymbol{k})$. During modelling, each round's subkey is typically treated as an independent variable. If the AO cipher is a nonce-based stream ciphers, then $\boldsymbol{x}$ is considered as the Nonce $\mathcal{N}$, and the output is treated as the generated keystream used for encryption, like in **Ciminion**.

### 3.1 Propagation Rules for General Monomial Prediction

According to Definition 3, we consider one general monomial trail

$$(\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \cdots \to \pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}) \to \cdots \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})),$$

where $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(i+1)}$ are the input and ouput of $F_i$. Each pair $(\boldsymbol{u}^{(i)}, \boldsymbol{u}^{(i+1)})$ is a valid general monomial trail through $F_i$ if and only if $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}) \to \pi_{\boldsymbol{u}^{(i+1)}}(\boldsymbol{x}^{(i+1)})$. If $F_i$ is the round function of AO ciphers with affine layers, then $F_i$ can be represented as a sequence of basic operations such as **XOR**,**AND**, **COPY**, **POWER** and **AFFINE**. In this subsection, we revisit the propagation rules in [22], and further extend their propagation rules, providing the propagation rules for **t-XOR** and **AFFINE**.

**Rule 1** (Field-based **XOR** [22]): Let $F_{XOR} : \mathbb{F}_{2^n}^2 \to \mathbb{F}_{2^n}$ be a function that consists of an XOR, where the input $\boldsymbol{x} = (x_0, x_1)$ takes values from $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ and the output $y$ is calculated as $y = x_0 \oplus x_1$. If the monomial $y^v$ contains $\boldsymbol{x}^{\boldsymbol{u}}$, then

$$\begin{cases} v = u_0 + u_1, \\ v \succeq u_i \quad \text{for } i \in \{0, 1\}, \end{cases}$$

where $0 \le v, u_0, u_1 \le 2^n - 1$.

**Rule 2** (Field-based **t-XOR**): Let $F_{\text{t-XOR}} : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}, t \ge 3$ be a function that consists of $t-1$ XOR, where the input $\boldsymbol{x} = (x_0, x_1, \ldots, x_{t-1})$ takes values from $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \times \cdots \times \mathbb{F}_{2^n}$ and the output $y$ is calculated as $y = x_0 \oplus x_1 \oplus x_2 \oplus \cdots \oplus x_{t-1}$. If the monomial $y^v$ contains $\boldsymbol{x}^{\boldsymbol{u}}$, then

$$\begin{cases} v = u_0 + u_1 + \cdots + u_{t-1}, \\ v \succeq u_i \quad \text{for } i \in \{0, 1, \ldots, t-1\}, \end{cases}$$

where $0 \le v, u_0, \ldots, u_{t-1} \le 2^n - 1$.

**Rule 3** (Field-based **AND** [22]): Let $F_{\text{AND}} : \mathbb{F}_{2^n}^2 \to \mathbb{F}_{2^n}$ be a function that consists of an AND, where the input $\boldsymbol{x} = (x_0, x_1)$ takes values from $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ and the output $y$ is calculated as $y = x_0 \cdot x_1$. If the monomial $y^v$ contains $\boldsymbol{x}^{\boldsymbol{u}}$, then

$$v = u_0 = u_1 = \cdots = u_{t-1},$$

where $0 \le v, u_0, u_1 \le 2^n - 1$.

**Rule 4** (Field-based **COPY** [22]): Let $F_{\text{COPY}} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}^2$ be a COPY function, where the input $x$ takes values from $\mathbb{F}_{2^n}$ and the output $\boldsymbol{y} = (y_0, y_1) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ is calculated as $\boldsymbol{y} = (x, x)$. If the monomial $\boldsymbol{y}^{\boldsymbol{v}}$contains $x^u$, then

$$\begin{aligned}\boldsymbol{v} &= (v_0, v_1)\\ &= \begin{cases}(0,0), & \text{if } u = 0;\\ (i, u-i) \text{ or } (j, u+2^n-1-j), & \text{else.}\end{cases},\end{aligned}$$

for $0 \le i \le u, u \le j \le 2^n - 1$, where $0 \le v_0, v_1, u \le 2^n - 1$.

**Rule 5** (Field-based **t-COPY** [22]): Let $F_{t-COPY} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}^t$ be a t-COPY function, where the input $x$ takes values from $\mathbb{F}_{2^n}$ and the output $\boldsymbol{y} = (y_0, \ldots, y_{t-1}) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \times \cdots \times \mathbb{F}_{2^n}$ is calculated as $\boldsymbol{y} = (\underbrace{x, x, \ldots, x}_{t})$. If the monomial $\boldsymbol{y}^{\boldsymbol{v}}$contains $x^u$, then

$$\begin{aligned}\boldsymbol{v} &= (v_0, v_1, \ldots, v_{t-1})\\ &= \begin{cases}(0,0,\ldots,0), & \text{if } u = 0;\\ (i_0^s, i_1^s, \ldots, i_{t-1}^s) \text{ for } 0 \le s \le t-1, & \text{else.}\end{cases}\end{aligned}$$

Here, $i_{t-1}^s = u + (s-1)(2^n-1) - \sum_{j=0}^{t-2} i_j^s$, $0 \le i_j^s \le 2^n - 1$ for $0 \le j < t$, where $u, v_0, v_1, \ldots, v_{t-1} \in [0, 2^n - 1]$.

**Rule 6** (Field-based **POWER** [22]): Let $F_{\text{POWER}} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a POWER function, where the input $x$ takes values from $\mathbb{F}_{2^n}$ and the output $y$ is calculated as $y = x^d$, for $\gcd(d, 2^n - 1) = 1$. If the monomial $y^v$ contains $x^u$, then

$$v = \begin{cases} u, & \text{if } u = 0 \text{ or } 2^n - 1;\\ (d^{-1}) \cdot u \pmod{2^n - 1}, & \text{else.}\end{cases}$$

where $0 \le v, u \le 2^n - 1$.

Consider the function $f : x \mapsto (x^d)^{2^h} = x^{d'}$ over $\mathbb{F}_{2^n}$, where $0 \le d, d' \le 2^n - 1$. For each $d \in [0, 2^n - 1]$, we can assign a bit vector $(d_{n-1}, d_{n-2}, \ldots, d_0)$ for $d$, where $d = \sum_{i=0}^{n-1} 2^i \cdot d_i$. So, we can calculate $d'$ as

$$\begin{aligned}&\left(d'_{n-1}, d'_{n-2}, \ldots, d'_0\right) &(1)\\ &\quad = \left(d_{(n-1-h)\%n}, d_{(n-2-h)\%n}, \ldots, d_{(0-h)\%}\right), &(2)\end{aligned}$$

because $d' = 2^h \cdot d \pmod{2^n - 1}$. This is equivalent to performing a left circular shift of $h$ positions for $d$.

**Rule 7** (Field-based **AFFINE**): Let $F_{\text{AFFINE}} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be an AFFINE function, where the input $x$ takes values from $\mathbb{F}_{2^n}$ and the output $y$ is calculated as $y = x^{2^h}$. If the monomial $y^v$ contains $x^u$, then

$$(u_{n-1}, u_{n-2}, \ldots, u_0) = (v_{(n-1-h)\%n}, v_{(n-2-h)\%n}, \ldots, v_{(0-h)\%n}),$$

where $v, u \in [0, 2^n - 1]$ and $(v_{n-1}, v_{n-2}, \cdots, v_0)$,$(u_{n-1}, u_{n-2}, \ldots, u_0)$ are the binary representation of $v, u$.

### 3.2 MILP and SAT Models for Field-Based Operations

When we use the SMT method to model general monomial prediction like [22], we observe that the model becomes computationally infeasible when dealing with AO ciphers featuring complex affine layers such as **Chaghri** et al. To address this issue, we explore alternative approaches such as MILP and SAT methods for modelling general monomial prediction. The propagation rules for general monomial prediction of AO ciphers have been described in the previous subsection. To account for XOR operation, we extend this rule to cover $t$-XOR and establish a comprehensive model. Additionally, in order to analyze AO ciphers with complex affine layers, we incorporate the AFFINE operation into our model. To depict the propagations of these operations, we translate the rules in Section 3.1 into linear inequalities for MILP and formulas in CNF for SAT. The solutions correspond to all valid general monomial trails. The MILP and SAT models for **XOR**, **3-XOR**, **AND**, **COPY**, **3-COPY**, and **POWER** and **AFFINE** are introduced as follows.

**Model 1** (Field-based **XOR**): Denote $(u_0, u_1) \xrightarrow{\textbf{XOR}} (v)$ a valid general monomial trail of the field-based XOR function, where $0 \le u_0, u_1, v \le 2^n - 1$. Then, the following linear equalities in the MILP model and clause normal forms in SAT model are sufficient to depict it.
The MILP linear equalities:

$$\begin{cases} u_{0,0} + u_{1,0} - v_0 = 0, \\ u_{0,1} + u_{1,1} - v_1 = 0, \\ \vdots \\ u_{0,n-1} + u_{1,n-1} - v_{n-1} = 0, \\ u_{i,j} \text{ and } v_j (0 \le i \le 1, 0 \le j \le n-1) \text{ are bit variables.} \end{cases}$$

The SAT clause normal forms:

$$\begin{cases} (\neg u_{0,i} \vee u_{1,i}) = 1, \\ (u_{1,i} \vee \neg v_i) = 1, \\ (\neg u_{0,i} \vee \neg v_i) = 1, \\ (u_{0,i} \vee \neg u_{1,i} \vee v_i) = 1, \end{cases} \quad \text{for } 0 \le i \le n-1.$$

**Model 2** (Field-based **3-XOR**): Denote $(u_0, u_1, u_2) \xrightarrow{\textbf{3-XOR}} (v)$ a valid general monomial trail of the field-based 3-XOR function, where $0 \le u_0, u_1, u_2, v \le 2^n - 1$. Then, the following linear equalities in the MILP model and clause normal forms in the SAT model are sufficient to depict it.
The MILP linear equalities:

$$\begin{cases} u_{0,0} + u_{1,0} + u_{2,0} - v_0 = 0, \\ u_{0,1} + u_{1,1} + u_{2,1} - v_1 = 0, \\ \vdots \\ u_{0,n-1} + u_{1,n-1} + u_{2,n-1} - v_{n-1} = 0, \\ u_{i,j} \text{ and } v_j (0 \le i \le 2, 0 \le j \le n-1) \text{ are bit variables.} \end{cases}$$

The SAT clause normal forms:

$$\begin{cases} (\neg u_{1,i} \vee \neg u_{2,i}) = 1, \\ (\neg u_{0,i} \vee \neg u_{2,i}) = 1, \\ (\neg u_{0,i} \vee \neg u_{1,i}) = 1, \\ (\neg u_{0,i} \vee v_i) = 1, \\ (\neg u_{1,i} \vee v_i) = 1, \\ (\neg u_{2,i} \vee v_i) = 1, \\ (u_{0,i} \vee u_{1,i} \vee u_{2,i} \vee \neg v_i) = 1, \end{cases} \quad \text{for } 0 \le i \le n-1.$$

**Model 3** (Field-based **AND**): Denote $(u_0, u_2) \xrightarrow{\textbf{AND}} (v)$ a valid general monomial trail of the field-based AND function, where $0 \le u_0, u_2, v \le 2^n - 1$. Then, according to Rule 3, the following linear equalities in the MILP model and clause normal forms in the SAT model are sufficient to depict it.
The MILP linear equalities:

$$\begin{cases} u_{0,0} = u_{1,0} = v_0, \\ u_{0,1} = u_{1,1} = v_1, \\ \vdots \\ u_{0,n-1} = u_{1,n-1} = v_{n-1}, \\ u_{i,j} \text{ and } v_j (0 \le i \le 1, 0 \le j \le n-1) \text{ are bit variables.} \end{cases}$$

The SAT clause normal forms:

$$\begin{cases} (u_{0,i} \vee \neg u_{1,i}) = 1, \\ (u_{1,i} \vee \neg v_i) = 1, \\ (\neg u_{0,i} \vee v_i) = 1, \end{cases} \quad \text{for } 0 \le i \le n-1.$$

**Model 4** (Field-based **AFFINE**): Denote $(u) \xrightarrow{\textbf{AFFINE}} (v)$ a valid general monomial trail of the field-based AFFINE function, where $0 \le u, v \le 2^n - 1$. Then, according to Rule 7, the following linear equalities in the MILP model and clause normal forms in the SAT model are sufficient to depict it.
The MILP linear equalities:

$$\begin{cases} u_0 - v_{(0-h)\%n} = 0, \\ u_1 - v_{(1-h)\%n} = 0, \\ \vdots \\ u_{n-1} - v_{(n-1-h)\%n} = 0, \\ u_i \text{ and } v_i (0 \le i \le n-1) \text{ are bit variables.} \end{cases}$$

The SAT clause normal forms:

$$\begin{cases} (\neg u_i \vee v_{(i-h)\%n}) = 1, \\ (u_i \vee \neg v_{(i-h)\%n}) = 1, \end{cases} \quad \text{for } 0 \le i \le n-1.$$

For two monomials $X^a$ and $X^b$ in the polynomial ring $\mathbb{F}_{2^n}[X]$, there is

$$X^a \cdot X^b = \begin{cases} X^{2^n-1} & \text{if } 2^n-1|(a+b), (a+b) \geq 2^n-1, \\ X^{(a+b) \pmod{2^n-1}} & \text{otherwise.} \end{cases}$$

Specifically, for the two-modulo addition $x + y \equiv z \pmod{2^n-1}$, where $x, y, z \in \mathbb{F}_{2^n}$, their binary representations are given by $(x_{n-1}, x_{n-2}, \ldots, x_0)$, $(y_{n-1}, y_{n-2}, \ldots, y_0)$, and $(z_{n-1}, z_{n-2}, \ldots, z_0)$. We can introduce two $(n+1)$-bit vectors $c = (c_n, c_{n-1}, \ldots, c_0)$, $c' = (c'_n, c'_{n-1}, \ldots, c'_0)$ and one $n$-bit vector $q = (q_{n-1}, q_{n-2}, \ldots, q_0)$ to represent intermediate values in order to compute the addition $x + y \equiv z \pmod{2^n-1}$. We have

$$\begin{cases} c_0 = 0, \\ 2c_{i+1} + q_i = x_i + y_i + c_i & \text{for } 0 \leq i \leq n-1, \\ c'_0 = c_n, \\ 2c'_{i+1} + z_i = q_i + c'_i & \text{for } 0 \leq i \leq n-1. \end{cases} \tag{3}$$

By substituting $c_0 = 0$ into the above equation, we can reduce the introduction of one bit variable. Therefore, the computation of $x + y \equiv z \pmod{2^n-1}$ requires a total of $3n + 1$ introduced bit variables. For the addition of $t$ elements in $\mathbb{F}_{2^n}$ modulo $2^n - 1$, $x_1 + x_2 + \cdots + x_t \equiv z \pmod{2^n-1}$, we can transform it into $t-1$ two-modulo additions and introduce $(t-2)$ additional $n$-bit vectors $((y_{1,n-1}, \ldots, y_{1,0}), \ldots, (y_{t-2,n-1}, \ldots, y_{t-2,0}))$ to record the value of each two-modulo addition. Therefore, we need to introduce $(3n+1)\cdot(t-1)+(t-2)\cdot n$ bit variables to compute the $t$-modulo addition.

$$\begin{cases} \text{TWO-MODULO}(x_1, x_2) \rightarrow y_1, \\ \text{TWO-MODULO}(x_{i+1}, y_{i-1}) \rightarrow y_i & \text{for } 2 \leq i \leq t-2, \\ \text{TWO-MODULO}(x_t, y_{t-2}) \rightarrow z. \end{cases} \tag{4}$$

Since the 3-COPY operation requires the computation of 3 elements in $\mathbb{F}_{2^n}$ modulo $2^n - 1$, when we use Equation 4 to calculate the three-modulo addition, we need to introduce $(3n + 1) \cdot 2 + n = 7n + 2$ bit variables. Therefore, we employ some techniques to reduce the number of introduced bit variables for the three-modulo addition. For the addition of 3 elements in $\mathbb{F}_{2^n}$ modulo $2^n - 1$, $x + y + z \equiv k \pmod{2^n-1}$, their binary representations are given by $(x_{n-1}, x_{n-2}, \ldots, x_0)$, $(y_{n-1}, y_{n-2}, \ldots, y_0)$, $(z_{n-1}, z_{n-2}, \ldots, z_0)$, and $(k_{n-1}, k_{n-2}, \ldots, k_0)$. We introduce three $n+1$-bit vectors $w = (w_n, \ldots, w_0)$, $w' = (w'_n, \ldots, w'_0)$, $w''' = (w'''_n, \ldots, w'''_0)$, one $n+2$-bit vector $w'' = (w''_{n+1}, \ldots, w''_0)$, and two $n$-bit vectors $p = (p_{n-1}, \ldots, p_0)$, $q = (q_{n-1}, \ldots, q_0)$ to represent intermediate values. We have

$$\begin{cases} w_0 = 0, \\ w''_i = 0 & \text{for } 0 \leq i \leq 2, \\ 4w''_{i+2} + 2w_{i+1} + q_i - x_i - y_i \\ \quad -z_i - w_i - w''_i = 0 & \text{for } 0 \leq i \leq n-1, \\ w'_0 - w_n = 0, \\ 2w'_1 + p_0 - q_0 - w'_0 = 0, \\ 2w'_2 + p_1 - q_1 - w'_1 - w''_{n+1}, \\ 2w'_{i+1} + p_i - q_i - w'_i = 0 & \text{for } 2 \leq i \leq n-1, \\ w'''_0 - w'_n = 0, \\ 2w'''_{i+1} + k_i - p_i - w'''_i & \text{for } 0 \leq i \leq n-1. \end{cases} \tag{5}$$

By substituting the determined variables into the above equation, the computation of the three-modulo addition only requires the introduction of $6n + 1$ bit variables instead of $7n + 2$.

**Model 5** (Field-based **POWER**): Denote $(u) \xrightarrow{\text{POWER}} (v)$ a valid general monomial trail of the field-based POWER function $F_{POWER} : x \mapsto x^d, \gcd(d, 2^n-1) = 1$, where $0 \leq u, v \leq 2^n - 1$. The Hamming weight of d is typically 2. Let $d = 2^{k_0}+2^{k_1}$, then $v\cdot d = v\cdot(2^{k_0}+2^{k_1}) = (v\cdot 2^{k_0}+v\cdot 2^{k_1}) \equiv u \pmod{2^n-1}$. The Hamming weight of d is typically 2. Let $d = 2^{k_0}+2^{k_1}$, then $v \cdot d = v \cdot (2^{k_0}+2^{k_1}) = (v \cdot 2^{k_0}+v \cdot 2^{k_1}) \equiv u \pmod{2^n-1}$. According to Rule 6, the following linear equalities in the MILP model are sufficient to depict it.
The MILP linear equalities:

$$\begin{cases} 2c_0 + q_0 - (v_{(0-k_0)\%n}) \\ \quad -(v_{(0-k_1)\%n}) = 0, \\ 2c_i + q_i - (v_{(i-k_0)\%n}) \\ \quad -(v_{(i-k_1)\%n}) - c_{i-1} = 0 & \text{for } 1 \leq i \leq n-1, \\ c'_0 - c_{n-1} = 0, \\ 2c'_{i+1} + u_i - q_i - c'_i = 0 & \text{for } 0 \leq i \leq n, \\ \text{All variables are bit variables.} \end{cases}$$

**Model 6** (Field-based **COPY**): Denote $(u) \xrightarrow{\text{COPY}} (v_0, v_1)$ a valid general monomial trail of the field-based COPY function $F_{COPY}$, where $0 \leq u, v_0, v_1 \leq 2^n - 1$. Then, according to Rule 4, the following linear equalities in the MILP model are sufficient to depict it.
The MILP linear equalities:

$$\begin{cases} 2g_0 + q_0 - (v_{0,0}) + (v_{1,0}) - g_0 = 0, \\ 2g_{i+1} + q_i - (v_{0,i}) - (v_{1,i}) - g_i = 0 & \text{for } 1 \leq i \leq n-1, \\ g'_0 - g_{n-1} = 0, \\ 2g'_{i+1} + u_i - q_i - g'_i = 0 & \text{for } 0 \leq i \leq n-1, \\ \text{All variables are bit variables.} \end{cases}$$

**Model 7** (Field-based **3-COPY**): Denote $(u) \xrightarrow{\text{3-COPY}} (v_0, v_1, v_2)$ a valid general monomial trail of the field-based COPY function $F_{3-COPY}$, where $0 \leq u, v_0, v_1, v_2 \leq 2^n - 1$. Then, according to Rule 5, the following linear equalities in the MILP model are sufficient to depict it. The MILP linear equalities:

$$
\begin{cases}
2w_0 + q_0 - v_{0,0} - v_{1,0} - v_{2,0}, \\
w'_0 - w_{n-1} = 0, \\
2w'_1 + p_0 - q_0 - w'_0 = 0, \\
w''' - w' = 0, \\
4w''_0 + 2w_1 + q_1 - v_{0,1} - v_{1,1} - v_{2,1} \\
\quad -w_0 = 0, \\
2w'_2 + p_1 - q - 1 - w'_1 - w''_{n-2} = 0, \\
4w''_1 + 2w_2 + q_2 - v_{0,2} - v_{1,2} - v_{2,2} \\
\quad -w_1 = 0, \\
4w''_{i-1} + 2w_i + q_i - v_{0,i} - v_{1,i} - v_{2,i} \\
\quad -w_{i-1} - w''_{i-3} = 0 & \text{for } 3 \le i \le n-1, \\
2w'_{i+1} + p_i - q_i - c'_i = 0 & \text{for } 2 \le i \le n-1, \\
2w'''_{i+1} + u_i - p_i + w'''_i = 0 & \text{for } 0 \le i \le n-1, \\
\text{All variables are bit variables.}
\end{cases}
$$

The SAT clause normal forms for the POWER, COPY, and 3-COPY operations can be found at `https://github.com/minionsjay/GMP_SAT/blob/main/SAT-CNF.pdf`.

### 3.3 Algorithms for Evaluating the Upper Bound of Algebraic Degree

In this subsection, we first show how to introduce variables to represent the propagation of monomials for different structures of AO ciphers, including block ciphers and nonce-based stream cipherss. Since the round functions of all AO ciphers can currently be decomposed into the propagation rules mentioned in the previous section, we can construct an initial model for AO ciphers. It is a general framework applicable to various AO ciphers constructed based on $\mathcal{F} : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$. We then explain how to set initial constraint, stopping rule, and objective functions in the model to evaluate the algebraic degree upper bounds of AO ciphers. Furthermore, we propose two automatic search algorithms based on MILP and SAT, which can effectively evaluate the algebraic degree upper bounds of AO ciphers.

**General model for AO ciphers.** General AO ciphers are composed of their round function $\mathcal{F} : \mathbb{F}_{2^n}^t \to \mathbb{F}_{2^n}^t$, iterated for $r$ rounds. Typically, the round function consists of a nonlinear layer (S-box), linear layer (affine transformations and MDS matrices), and round key addition (for block ciphers) or round constants (for nonce-based stream cipherss and hash functions). We denote the input and output monomial of $i$-th round as $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)})$ and $\pi_{\boldsymbol{u}^{(i+1)}}(\boldsymbol{x}^{(i+1)})$, where $0 \le i \le r$. The monomials $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ represent the plaintext and ciphertext corresponding monomials. Assuming round keys are mutually independent, we treat them as variables in model construction. We denote $\pi_{\boldsymbol{v}^{(i)}}(\boldsymbol{k}^{(i)}), 1 \le i \le r$ as round keys for the $i$-th round, and $\pi_{\boldsymbol{v}^{(0)}}(\boldsymbol{k}^{(0)})$ as the whitening key monomial. The primary variables for constructing the general monomial propagation model of AO block cipher are illustrated in the Figure 1.

If AO ciphers are nonce-based stream ciphers or hash functions and we only consider modelling them using permutation, the input variables would be the nonce and key for nonce-based stream cipherss and the message to be hashed for hash functions. As depicted in Figure 1, we simply need to treat the key variables as constants by setting $\boldsymbol{v}^{(i)}$ as constants rather than considering them as free variables. By connecting variables within each round using the propagation rules described in the Subsection 3.1, we establish the initial model for monomial propagation in AO ciphers. For an $r$-round AO cipher, to determine if a monomial exists in the output monomials, we need to add an initial constraint, a stopping rule, and an objective function to the initial model.

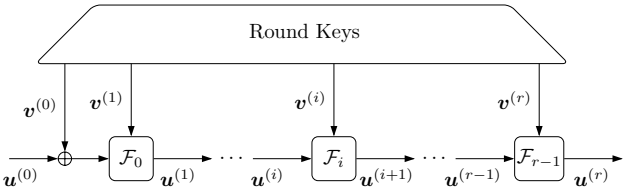


**Fig. 1** Variables in MILP and SAT model for AO ciphers.

**Initial constraint and stopping rule.** Given $U, V \in \mathbb{F}_{2^n}^t$, to verify the existence of the monomial $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \cdot \pi_{\boldsymbol{v}^{(i)}}(\boldsymbol{x}^{(i)})$ in the $j$-th word of the $r$-round AO cipher output $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, we define $\boldsymbol{u}^{(0)} = U$ and $\boldsymbol{u}^{(r)} = \boldsymbol{e}_j$ in the model, with $\boldsymbol{v}^{(i)}$ as the free variables, where $0 \le i \le r-1, 0 \le j \le t-1$. To evaluate the maximum algebraic degree of $r$-round AO ciphers, we categorize the initial constraint into the single-variable setting and the multi-variable setting. In the single-variable setting, we set the initial constraint as $\boldsymbol{u}^{(0)} = (u_0, \ldots, u_{t-1}) = (u_0, 0, \ldots, 0)$, where $u_0 \in \mathbb{F}_{2^n}$ is a variable. On the other hand, in the multi-variable setting, the initial constraint is established as $\boldsymbol{u}^{(0)} = (u_0, \ldots, u_{t-1})$, where the number of variables in $\boldsymbol{u}^{(0)}$ depends on the chosen variable count. As for the stopping rule, we generally set $V$ as $\boldsymbol{e}_1$.

**Objective function**:

*MILP model*: According to the Proposition 1, the algebraic degree of a multivariate function $\boldsymbol{F}$ is defined as $d_{\boldsymbol{F}} = \max\left\{\sum_{i=0}^{t-1} \mathrm{H}(u_i) : u_i \in [0, 2^n - 1], \phi(\boldsymbol{u}) \neq 0\right\}$. To evaluate the maximum algebraic degree of the monomial $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ in the output of an $r$-round AO cipher, given the initial constraint and stopping rule, we set the objective function of the MILP model as follows

$$
\begin{aligned}
&\text{Maximum}\left(\sum_{0 \le i \le t-1} \mathrm{H}(u_i^{(0)})\right) \\
&\quad = \text{Maximum}\left(\sum_{0 \le i \le t-1, 0 \le j \le n-1} u_{i,j}^{(0)}\right).
\end{aligned}
$$

In particular, when operating in a single-variable setting where all variables except $u_0$ are set to 0, the objective function becomes

$$\text{Maximum}\left(\mathrm{H}(u_0^{(0)})\right) = \text{Maximum}\left(\sum_{0\le i\le n-1} u_{0,i}^{(0)}\right).$$

*SAT model*: The Boolean satisfiability problem (SAT) is the problem of determining whether there exists an evaluation for the binary variables such that the value of the given Boolean formula equals one. The SAT method cannot directly compute the Boolean cardinality constraint in the same manner as the MILP method, providing its maximum value. After setting the initial constraint and stopping rules, if we aim to check whether the output monomials contain a monomial $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}) \cdot \pi_{\boldsymbol{v}^{(i)}}(\boldsymbol{k})$ with an algebraic degree of $\delta$. We add the constraint

$$\mathrm{H}(\boldsymbol{u}^{(0)}) = \sum_{0\le i\le t-1, 0\le j\le n-1} u_{i,j}^{(0)} = \delta,$$

or

$$\mathrm{H}(u_0^{(0)}) = \sum_{0\le i\le n-1} u_{0,i}^{(0)} = \delta, \tag{6}$$

to the SAT model. This is challenging in the SAT model, so we introduce the sequential encoding method [27]. These kinds of constraints can be abstracted as the Boolean cardinality constraint $\sum_{j=0}^{n-1} x_j \le k$, where $x_j$' are Boolean variables, and $k$ is a non-negative integer. Following the approaches in [33], we improve the conversion of the Equation 6 into CNF formulas.

*Sequential Encoding Method.* The sequential encoding method introduces $(n-1)\cdot k$ auxiliary variables $s_{i,j}$ $(0 \le i \le n-2, 0 \le j \le k-1)$ to compute the partial sum $s_i = \sum_{j=0}^{i} x_j$ for the cardinality constraints. Specifically, for each partial sum $s_i \le k$, it is represented as a unary $s_{i,0}||s_{i,1}||\ldots||s_{i,k-1}$ using $k$ auxiliary variables, where $s_i = m$ implies $s_{i,0} = \cdots = s_{i,m-1}$ and $s_{i,m} = \ldots s_{i,k-1} = 0$, ensuring $s_i = \sum_{j=0}^{k-1} s_{i,j} = m$. By employing the constraints from $s_{i-1}$ and $x_i$, the SAT model for the Boolean cardinality constraint $\sum_{j=0}^{n-1} x_j \le k$ can be established using the sequential encoding method. Implementing the sequential encoding method requires the addition of $2\cdot kn + n - 3\cdot k - 1$ clauses.

$$\begin{cases} \neg x_0 \vee s_{0,0} = 1, \neg s_{0,j} = 1 \\ \neg x_i \vee s_{i,0} = 1, \neg s_{i-1,0} \vee s_{i,0} = 1, \\ \neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j} = 1, \\ \neg s_{i-1,j} \vee s_{i,j} = 1, \\ \neg x_i \vee \neg s_{i-1,k-1} = 1, \\ \neg x_{n-1} \vee \neg s_{n-2,k-1} = 1, \end{cases} \tag{7}$$

where $1 \le j \le k-1, 1 \le i \le n-2$.

To transform $\sum_{i=0}^{n-1} x_i = k$ into CNF, we need to add some additional clauses to the CNF 7. We observe that the first constraint, $\neg x_i \vee s_{0,0} = 1$ , only determines $s_{0,0}$ to be 1. When $x_0 = 0$, $s_{0,0}$ remains undetermined. Therefore, we need to add a clause for when $x_0 = 0$, yielding $s_{0,0} = 0$. This leads us to include the clause

$$x_0 \vee \neg s_{0,0} = 1.$$

Similarly, we need to add clauses after the third and fourth clauses of the CNF 7. Here, $s_{i,0}$ is determined by $x_i$ and $s_{i-1,0}$. If both $x_i$ and $s_{i,0}$ are 0, then $s_{i,0} = 0$. Therefore, we include the clause

$$x_i \vee s_{i-1,0} \vee \neg s_{i,0} = 1.$$

For the fifth and sixth clauses of the CNF 7, we add clauses to determine $s_{i,j}$. We need to include the following clauses:

$$x_i \vee s_{i-1,j} \vee \neg s_{i,j} = 1, s_{i-1,j-1} \vee s_{i-1,j} \vee \neg s_{i,j} = 1.$$

By following this approach, we need to add $1 + (1 + 2\cdot(k-1))\cdot(n-2)$ clauses to convert the cardinality constraint $\sum_{j=0}^{n-1} x_j \le k$ to $\sum_{i=0}^{n-1} x_i = k$. Overall, a total of $4\cdot k\cdot n - 7\cdot k + 2$ clauses are required, as shown below.

$$\begin{cases} \neg x_0 \vee s_{0,0} = 1, x_0 \vee \neg s_{0,0} = 1, \\ \neg s_{0,j} = 1, \neg x_i \vee s_{i,0} = 1, \\ \neg s_{i-1,0} \vee s_{i,0} = 1, \\ x_i \vee s_{i-1,0} \vee \neg s_{i,0} = 1, \\ \neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j} = 1, \\ \neg s_{i-1,j} \vee s_{i,j} = 1, \\ x_i \vee s_{i-1,j} \vee \neg s_{i,j} = 1, \\ s_{i-1,j-1} \vee s_{i-1,j} \vee \neg s_{i,j} = 1, \\ \neg x_i \vee \neg s_{i-1,k-1} = 1, \\ \neg x_{n-1} \vee \neg s_{n-2,k-1} = 1, \end{cases}$$

where $1 \le j \le k-1, 1 \le i \le n-2$.

**Algorithms to evaluate the upper bound of algebraic degree.** To denote the monomials corresponding to the input of the $i$th round, we use $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)})$. Given a monomial $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)}) \cdot \pi_{\boldsymbol{v}^{(i)}}(\boldsymbol{k}^{(i)})$ uniquely specified by $\boldsymbol{u}^{(i)}$ and $\boldsymbol{v}^{(i)}$. Our MILP and SAT models are described based on variables in $\boldsymbol{u}^{(i)}$ and $\boldsymbol{v}^{(i)}$, for $0 \le i \le r$, as shown in Figure 1.

After setting the initial constraint, stopping rules, and objective function, we use off-the-shelf solvers like Gurobi[34] to solve our MILP model. As described in Algorithm 1, we can obtain the upper bound of algebraic degree for $r$ rounds. Based on our experiments, evaluating the upper bound of the algebraic degree using the SAT method is generally more effective, but it requires us to input a test algebraic degree. Therefore, we can combine Algorithm 1 and Algorithm 2. The algebraic degree obtained from Algorithm 1 serves as the input test algebraic degree for Algorithm 2, and solvers like CryptoMiniSat[35] or CaDiCaL[28] are used to solve the SAT model and obtain a more accurate upper bound for the algebraic degree. For efficiency enhancement, we can bypass the results obtained from Algorithm 1 and instead utilize the coefficient grouping technique proposed by Liu et al.[26] to efficiently evaluate the algebraic degree, acting as the test algebraic degree input for Algorithm 2.

## 4. Applications

In this section, we apply the search tool based on MILP and

**Algorithm 1:** Evaluate the upper bound on algebraic degree $d$ using MILP method

**Input:** The $r$-round MILP model $\mathcal{M}^r_{MILP}$, the initial constraint $U$ and stopping rule $V$.
**Output:** The algebraic degree upper bound $d$.

$\mathcal{M} \leftarrow \mathcal{M}^r_{\text{MILP}}$;
$d \leftarrow 0$;
**for** $i = 0$ *to* $t-1$ **do**
  $\mathcal{M}.con \leftarrow u_i^{(0)} = U[i]$;
  $\mathcal{M}.con \leftarrow u_i^{(r)} = V[i]$;
$\mathcal{M}.obj \leftarrow \text{Maxmium}(\sum_{i=0}^{t} \text{H}(u_j^{(0)}))$;
$\mathcal{M}.optimize()$;
$d \leftarrow \mathcal{M}.obj$;
**return** $d$;

**Algorithm 2:** Evaluate the upper bound on algebraic degree $d$ using SAT method

**Input:** The $r$-round SAT model $\mathcal{M}^r_{SAT}$, the test upper bound of algebraic degree $\delta$, the initial constraint $U$ and stopping rule $V$.
**Output:** The upper bound of algebraic degree $d$.

$\mathcal{M} \leftarrow \mathcal{M}^r_{\text{SAT}}$;
$d \leftarrow 0$, $flag \leftarrow 0$;
**for** $i = 0$ *to* $t-1$ **do**
  $\mathcal{M}.con \leftarrow u_i^{(0)} = U[i]$;
  $\mathcal{M}.con \leftarrow u_i^{(r)} = V[i]$;
$\mathcal{M}.con \leftarrow \sum_{i=0}^{t} \text{H}(u_j^{(0)}) = \delta$;
solve the $r$-round SAT model $\mathcal{M}$;
**if** *the problem is satisfiable* **then**
  $flag \leftarrow 1$;
$\mathcal{M}.con \leftarrow remove\left(\sum_{i=0}^{t} H(u_j^{(0)}) = \delta\right)$;
**if** $flag = 1$ **then**
  **for** $\delta = \delta + 1$ *to* $n$ **do**
    $\mathcal{M}.con \leftarrow \sum_{i=0}^{t} H(u_j^{(0)}) = \delta$;
    solve the $r$-round SAT model $\mathcal{M}$;
    **if** *the problem is unsatisfiable* **then**
      $d \leftarrow \delta - 1$;
      **return** $d$;
  **return** $n$;
**else**
  **for** $\delta = \delta - 1$ *to* $0$ **do**
    $\mathcal{M}.con \leftarrow \sum_{i=0}^{t} H(u_j^{(0)}) = \delta$;
    solve the $r$-round SAT model $\mathcal{M}$;
    **if** *the problem is satisfiable* **then**
      $d \leftarrow \delta$;
      **return** $d$;

SAT introduced in the previous section to several AO ciphers and their variants, including **Chaghri**, **MiMC** with different affine layers, and **Ciminion**. We obtain upper tighter bounds on the algebraic degree for these ciphers under different parameter settings.

### 4.1 Application to **Chaghri** and **MiMC** with Complex Affine Layer

**Specification of Chaghri.** **Chaghri** is an AO block cipher designed for Fully Homomorphic Encryption (FHE) applications and is defined over $\mathbb{F}_{2^{63}}$. It consists of a total of eight rounds, with each round divided into two steps. The state of **Chaghri** is denoted as $z = (z_0, z_1, z_2) \in \mathbb{F}^3_{2^{63}}$. Since **Chaghri** is designed for FHE schemes, the choice of the security rounds during its design primarily considers the security in the decryption direction and low multiplication depth. In the encryption direction of **Chaghri**, the algebraic degree of its S-boxes is remarkably high. After a few rounds, the algebraic degree in the encryption direction reaches its upper bound and no longer increases. Additionally, the affine layers become very dense. Therefore, our focus is mainly on evaluating the algebraic degree in the decryption direction of **Chaghri**.

In **Chaghri**, the round key $RK_i = (RK_{i,0}, RK_{i,1}, RK_{i,2}) \in \mathbb{F}^3_{2^{63}}$ is derived from the master key $MK = (MK_0, MK_1, MK_2) \in \mathbb{F}^3_{2^{63}}$. Before entering the round function, there is a whitening key $RK_0 = (RK_{0,0}, RK_{0,1}, RK_{0,2})$. In the following, we explain each component used in the round function, namely $\mathcal{G}$, $\mathcal{B}$, and $\mathcal{M}$.

- **The nonlinear function** $\mathcal{G}(x) : \mathbb{F}_{2^{63}} \rightarrow \mathbb{F}_{2^{63}}$. $\mathcal{G}(x)$ is defined as $\mathcal{G}(x) = x^{2^{32}+1}$.
- **The affine transform** $\mathcal{B}(x) : \mathbb{F}_{2^{63}} \rightarrow \mathbb{F}_{2^{63}}$. $\mathcal{B}(x)$ is defined as $\mathcal{B}(x) = c_1 x^{2^8} + c_2 x^{2^2} + c_3 x + c_4$, where $c_1, c_2, c_3 \in \mathbb{F}_{2^{63}} \backslash \{0\}$, $c_4 \in \mathbb{F}_{2^n}$ are constants.
- **The linear transform** $\mathcal{M} : \mathbb{F}^3_{2^{63}} \rightarrow \mathbb{F}^3_{2^{63}}$. $\mathcal{M}$ is a $3 \times 3$ MDS matrix. The designers do not specify a concrete choice for $\mathcal{M}$ and they claim any MDS matrix is suitable. We note here that our attacks apply to any choice of $\mathcal{M}$.

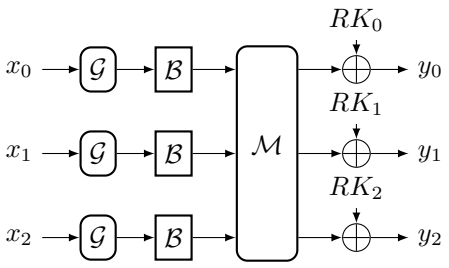


**Fig. 2** The step function of **Chaghri**.

The decryption round function $\mathcal{F}$ for Chaghri consists in iterating two step functions, with the step function described in Figure 2. According to the description of the step function, we can obtain the round function of **Chaghri** as $\mathcal{F}(z) = \mathcal{AK} \circ \mathcal{M} \circ \mathcal{B} \circ \mathcal{G} \circ \mathcal{AK} \circ \mathcal{M} \circ \mathcal{B} \circ \mathcal{G}(z)$. Therefore, the **Chaghri** algorithm with 8 rounds can be defined as $\textbf{Chaghri}(z) = \mathcal{F}_6 \circ \cdots \circ \mathcal{F}_0 \circ \mathcal{AK}(z)$.

**Specification of MiMC. MiMC** is an iterated key-alternating $n$-bit block cipher, where each round consists of a key addition with the key $k$, the addition of a round constant $c_i \in \mathbb{F}_{2^n}$. The nonlinear component of the construction is the evaluation of the cube function $\mathcal{F}(x) = x^3$ over $\mathbb{F}_{2^n}$. The ciphertext is finally produced by adding the key $k$ again to the output of the last round. Additionally, different round constants are used to break symmetries, with the first round constant being 0. Therefore, the round function is described as $\mathcal{F}_i(x) = \mathcal{F}(x \oplus k \oplus c_i)$, where $c_0 = c_r = 0$. The encryption process of **MiMC** is defined as $\mathbf{MiMC}(n, r) := \mathcal{F}_{r-1} \circ \mathcal{F}_{r-2} \circ \cdots \circ \mathcal{F}(x) \oplus k$.

The algebraic degree of **MiMC** has been widely investigated in prior research, as examined in [19]. The algebraic degree of **MiMC** demonstrates linear growth. In Crypto 2023, Liu et al. [26] made an attempt to incorporate affine layers into **MiMC**, aiming to achieve exponential growth in its algebraic degree. They also provided specific parameters for the affine layers. Therefore, the evaluated algebraic degrees of **MiMC** in this paper are conducted with the inclusion of affine layers and compared with the evaluation results under the same parameters set by Liu et al. The affine function $\mathcal{B}(x) = c_0 + \sum_{i=1}^{w} c_i x^{2^{h_i}}$ is characterized by a set of parameters $h = \{h_1, \ldots, h_w\}$, where $c_1, \ldots, c_w \in \mathbb{F}_{2^n} \backslash \{0\}$.

**Model for the general monomial trail of Chaghri and MiMC with affine layer.** Denote $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ as the initial state of **Chaghri**, and let $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)})$ and $\pi_{\boldsymbol{u}^{(i+1)}}(\boldsymbol{x}^{(i+1)})$ represent the input and output of the $i$-th step function in **Chaghri**, respectively. Assuming an analysis of $r$ steps of **Chaghri**, $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ represents the monomials of the plaintext, while $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ represents the monomials of the ciphertext. Assuming that the round keys for each round are independent, denoted by $\pi_{\boldsymbol{v}^{(i)}}(\boldsymbol{k}^{(i)})$. The model for the general monomial trail of **Chaghri** is described based on the variables $\boldsymbol{u}^{(i)}$ and $\boldsymbol{v}^{(i)}$ as shown in Figure 1. Additionally, auxiliary variables need to be introduced in the model based on the modelling approach described in Section 3.2. The general monomial trail models of **Chaghri** and **MiMC**, denoted as $\mathcal{M}_{\mathbf{Chaghri}}$ and $\mathcal{M}_{\mathbf{MiMC}}$ respectively, both with parameters $(h_1, \ldots, h_w)$, are described as follows.

$$\mathcal{M}_{\mathbf{MiMC}} \leftarrow \begin{cases} (u^{(i)}, v^{(i)}) \xrightarrow{\textbf{XOR}} (y^{(i)}), \\ (y^{(i)}) \xrightarrow{\textbf{w-COPY}} (y_1^{(i)}, \ldots, y_w^{(i)}), \\ (y_1^{(i)}) \xrightarrow{\textbf{POWER+AFFINE}} (py_1^{(i)}), \\ \vdots \\ (y_w^{(i)}) \xrightarrow{\textbf{POWER+AFFINE}} (py_w^{(i)}), \\ (py_1^{(i)}, py_w^{(i)}) \xrightarrow{\textbf{XOR}} (u^{(i+1)}), \end{cases}$$

where $0 \le i < r$.

$$\mathcal{M}_{\mathbf{Chaghri}} \leftarrow \begin{cases} (u_i^{(j)}, v_i^{(j)}) \xrightarrow{\textbf{XOR}} (y_i^{(j)}), \\ (y_i^{(j)}) \xrightarrow{\textbf{w-COPY}} (y_{i,1}^{(j)}, \ldots, y_{i,w}^{(j)}), \\ (y_{i,1}^{(j)}) \xrightarrow{\textbf{POWER+AFFINE}} (py_{i,1}^{(j)}), \\ \vdots \\ (y_{i,w}^{(j)}) \xrightarrow{\textbf{POWER+AFFINE}} (py_{i,w}^{(j)}), \\ (py_{i,1}^{(j)}, \ldots, py_{i,w}^{(j)}) \xrightarrow{\textbf{w-XOR}} (z_i^{(j)}), \\ (z_i^{(j)}) \xrightarrow{\textbf{3-COPY}} (z_{i,0}^{(j)}, z_{i,1}^{(j)}, z_{i,2}^{(j)}), \\ (z_{0,0}^{(j)}, z_{1,0}^{(j)}, z_{2,0}^{(j)}) \xrightarrow{\textbf{3-XOR}} (u_0^{(j+1)}), \\ (z_{0,1}^{(j)}, z_{1,1}^{(j)}, z_{2,1}^{(j)}) \xrightarrow{\textbf{3-XOR}} (u_1^{(j+1)}), \\ (z_{0,2}^{(j)}, z_{1,2}^{(j)}, z_{2,2}^{(j)}) \xrightarrow{\textbf{3-XOR}} (u_2^{(j+1)}), \end{cases}$$

where $0 \le i \le 2, 0 \le j < r$.

**Algebraic degree of Chaghri and MiMC with different affine layers.** For **Chaghri**, we evaluate its algebraic degree under the single-variable setting. The $i$-th output word of $r$ rounds of **Chaghri** constitutes a function of the initial state $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$. As described in Section 3.3, we set the initial constraint as $U = (u_0, 0, 0) \in \mathbb{F}_{2^n}^3$, where $u_0$ is a variable. The stopping rule is denoted as $V = \boldsymbol{e}_1 \in \mathbb{F}_{2^n}^3$. Considering that **MiMC**'s state is $t = 1$, analogous to **Chaghri**, we set the initial constraint as $U = (u_0)$, where $u_0 \in \mathbb{F}_{2^n}$ is a variable, and the stopping rule is $V = \boldsymbol{e}_1 \in \mathbb{F}_{2^n}$. By calling Algorithm 1 or Algorithm 2, we can obtain the upper bounds for the algebraic degree of $r$ rounds of **Chaghri** and **MiMC**, as shown in Table 3 and Table 4.

### 4.2 Application to **Ciminion**

**Specification of Ciminion. Ciminion** is a nonce-based stream ciphers proposed by Dobraunig et al. at EUROCRYPT 2021 [4]. Its objective is to minimize the number of multiplications in large fields $\mathbb{F}_p$ with $p \ge 2^{64}$ or $\mathbb{F}_{2^n}$ with $n \ge 64$. Unlike **Chaghri** and **MiMC** which use a power map as the non-linear layer, **Ciminion** employs Toffoli gates $(x, y, z) \mapsto (x, y, xy + z)$. Additionally, **Ciminion** uses a very lightweight linear layer instead of an MDS matrix.

**Encryption scheme.** As shown in Figure 3, the scheme takes a nonce $\mathcal{N}$ along with two subkey elements $K_1$ and $K_2$ as input, and processes the input with a permutation $P_C$ to output an intermediate state. Then this intermediate state is used as the input of a permutation $P_E$. The output state is truncated to two elements, which are used to encrypt two plaintext elements $P_1$ and $P_2$. If more elements need to be encrypted, the intermediate state can be expanded by repeatedly performing an addition of two subkey elements, then followed by a call to the rolling function *rol*. After each call to the rolling function *rol*, the output state is used as the input of the corresponding permutation $P_E$. In this way, two more plaintext elements $P_{2i}$ and $P_{2i+1}$ are encrypted by the truncated elements of the resulting state.

- **Permutation. Ciminion** has two permutations, $P_C$ and $P_E$. They act on a state of triples $(a, b, c) \in \mathbb{F}_q^3$, where

**Table 4** The new algebraic degree upper bound of the **MiMC** with different affine layers.

| $(n, d, h_1, h_2)$ | References | $r$ rounds | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\leq 3$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| $(129, 3, 0, 6)$ | [26] | $2^r$ | 15 | 27 | 40 | 49 | 57 | 65 | 73 | 81 | 89 | 97 | 105 | 112 | 129 | - | - | - | - | - | - | - |
| | Section 4.1 | $2^r$ | **12** | **22** | **32** | **42** | **52** | **60** | **68** | **74** | **82** | **90** | **98** | **106** | **112** | **120** | **128** | **129** | - | - | - | - |
| $(129, 3, 0, 9)$ | [26] | $2^r$ | 15 | 27 | 48 | 66 | 78 | 89 | 100 | 111 | 122 | 129 | - | - | - | - | - | - | - | - | - | - |
| | Section 4.1 | $2^r$ | **12** | **22** | **32** | **48** | **66** | **82** | **94** | **104** | **116** | **126** | **128** | **129** | - | - | - | - | - | - | - | - |
| $(129, 3, 0, 63)$ | [26] | $2^r$ | 15 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 | 104 | 112 | 120 | 128 | 129 | - | - | - | - |
| | Section 4.1 | $2^r$ | **12** | **18** | **26** | **32** | **38** | **46** | **52** | **58** | **64** | **70** | **76** | **82** | **88** | **94** | **100** | **106** | **112** | **118** | **124** | **129** |

$q = 2^n$ or a prime number $q = p$ of approximately $n$ bits ($\log_2(p) \approx n$). Both permutations are constructed from the same round function $f$, with $N$ rounds and $R$ rounds, respectively. The round function is illustrated in Figure 4.

- **Round Function.** Let the round function of the $i$-th round be denoted as $f_i$. Then the round functions of $P_C$ and $P_E$ are $f_i$ and $f_{i+NR}$, respectively. The round function $f_i$ requires four round constants $(rc_{i,1}, rc_{i,2}, rc_{i,3}, rc_{i,4})$ which are generated with Shake-256 [36], [37], and $rc_{i,4}$ is not equal to 0 or 1. Let the input state of $f_i$ be $(a_{i-1}, b_{i-1}, c_{i-1})$, then the output state is $(a_i, b_i, c_i)$, and their relationship is as follows

$$\begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} := \begin{bmatrix} 0 & 0 & 1 \\ 1 & rc_{i,4} & rc_{i,4} \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{i-1} \\ b_{i-1} \\ c_{i-1} + a_{i-1} \cdot b_{i-1} \end{bmatrix} + \begin{bmatrix} rc_{i,3} \\ rc_{i,2} \\ rc_{i,1} \end{bmatrix}.$$

- **The Rolling Function.** The rolling function, denoted as *rol*, is a simple Non-Linear Feedback Shift Register (NLFSR). It takes three field elements as input, denoted as $(l_a, l_b, l_c)$. The output is also three field elements, denoted as $(w_a, w_b, w_c)$. Their relationship is given as $w_= l_c + l_a \cdot l_b$, $w_b = l_a$, and $w_c = l_b$.

**Subkey Generation.** The subkey $K_i$ is generated from two master keys $(MK_1, MK_2)$. The author employs a sponge structure [38], instantiated with the permutation $P_C$, to expand the master key and generate subkey $K_i$. In our model, we assume that each subkey $K_i$ is mutually independent; thus, the key generation algorithm is not considered.

**Ciminion** is designed to operate over prime fields and binary fields. In this paper, our focus lies specifically on the binary field variant of **Ciminion**. More precisely, we primarily analyze the algebraic degree of the permutation $P_C$.

**Model for the general monomial trail of Ciminion permutation $P_C$.** Denote $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ as the initial monomial of **Ciminion** permutation $P_C$, and let $\pi_{\boldsymbol{u}^{(i)}}(\boldsymbol{x}^{(i)})$ and $\pi_{\boldsymbol{u}^{(i+1)}}(\boldsymbol{x}^{(i+1)})$ represent the input and output of the $i$-th round function in $P_C$, respectively. Assuming an analysis of $r$ rounds of $P_C$, $\pi_{u_0^{(0)}}(x^{(0)})$ represents the monomials of the Nonce $\mathcal{N}$, while $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ represents the monomials of the output of $P_C$. The round keys $K_1, K_2$ are treated as independent variables denoted by $\pi_{u_1^{(i)}}(k_1^{(i)}), \pi_{u_2^{(i)}}(k_2^{(i)})$. The

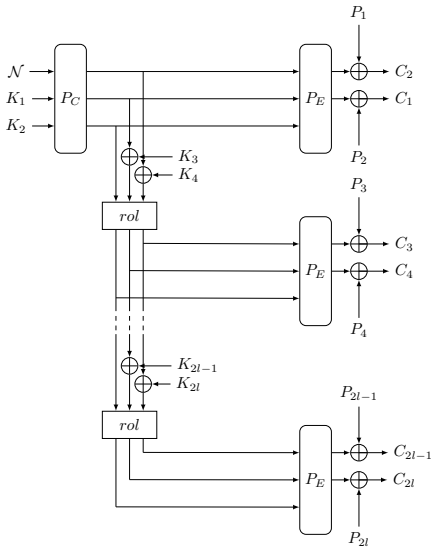


**Fig. 3** Encryption with **Ciminion** over $\mathbb{F}_{2^n}$. The construction is similar over $\mathbb{F}_p$ ($\oplus$ is replaced by $+$, the addition module $p$).

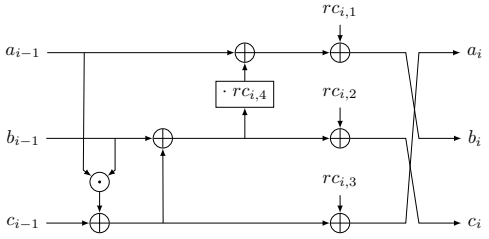


**Fig. 4** **Ciminion** Round function $f_i$.

model for the general monomial trail of $P_C$ is described based on the variables $(u_0^{(i)}, u_1^{(i)}, u_2^{(i)})$. Additionally, auxiliary variables need to be introduced in the model based on the modelling approach described in Section 3.2. According to the round function of **Ciminion**, the general monomial trail model of $P_C$, denoted as $\mathcal{M}_{\text{Ciminion}}$.

**Algebraic degree of the permutation $P_C$.** The $i$-th output word of $r$ rounds of **Ciminion** constitutes a function of the initial state $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$. As described in Section 3.3, we set the initial constraint as $U = (u_0, u_1, u_2) \in \mathbb{F}_{2^n}^3$, where $u_0, u_1, u_2$ are variables. The stopping rule is denoted as $V = \boldsymbol{e}_1 \in \mathbb{F}_{2^n}^3$. By calling Algorithm 1 or Algorithm 2,

we can obtain the upper bounds for the algebraic degree of $r$ rounds of $P_C$, as shown in Table 5. Compared to the claimed upper bounds in [29], we achieve tighter results, thus according to the analysis in [29], we can similarly construct a higher-order differential distinguisher with an one more round.

**Table 5** The new algebraic degree upper bound of the first two branches of the **Ciminion** $P_C$.

| Algebraic degree | References | $r$ rounds | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | ... | 30 | ... | 64 | 65 | 66 |
| $d_r$ | [29] | 1 | 1 | 2 | 3 | ... | 29 | ... | 63 | 64 | 65 |
| | Section 4.2 | 1 | 1 | 2 | **2** | ... | **28** | ... | **62** | **63** | **64** |

$$\mathcal{M}_{\textbf{Ciminion}} \leftarrow \begin{cases} (u_0^{(r)}) \xrightarrow{\textbf{COPY}} (u_{0,0}^{(r)}, u_{0,1}^{(r)}), \\ (u_1^{(r)}) \xrightarrow{\textbf{COPY}} (u_{1,0}^{(r)}, u_{1,1}^{(r)}), \\ (u_{0,1}^{(r)}, u_{1,1}^{(r)}) \xrightarrow{\textbf{AND}} (y^{(r)}), \\ (y^{(r)}, u_2^{(r)}) \xrightarrow{\textbf{XOR}} (z_0^{(r)}), \\ (z_0^{(r)}) \xrightarrow{\textbf{COPY}} (z_{0,0}^{(r)}, z_{0,1}^{(r)}), (z_{0,0}^{(r)}) \rightarrow (u_0^{r+1}), \\ (z_{0,1}^{(r)}, u_{1,0}^{(r)}) \xrightarrow{\textbf{XOR}} (z_1^{(r)}), \\ (z_1^{(r)}) \xrightarrow{\textbf{COPY}} (z_{1,0}^{(r)}, z_{1,1}^{(r)}), (z_{1,0}^{(r)}) \rightarrow (u_2^{r+1}), \\ (z_{1,1}^{(r)}, u_{0,0}^{(r)}) \xrightarrow{\textbf{XOR}} (z_2^{(r)}), (z_2^{(r)}) \rightarrow (u_1^{r+1}), \end{cases}$$

where $0 \leq r < R$.

### 4.3 Testing the actual algebraic degree of **Ciminion**, **Chaghri** and **MiMC** with different affine layers

To verify the accuracy of our evaluation results, we calculated the actual algebraic degree of reduced-round **Ciminion**, **Chaghri** and **MiMC** with different affine layers. For **Chaghri**, we used an arbitrary invertible matrix for the linear transformation matrix. The actual algebraic degrees are shown in the Table 6. By calculating the actual algebraic degrees, we verified the correctness of our model's evaluation of the algebraic degrees and demonstrated the accuracy of our model's evaluation of the upper bounds of the algebraic degrees. All the code for calculating the actual algebraic degrees is in `https://github.com/minionsjay/GMP_SAT`.

## 5. Conclusion

In this paper, we present an automatic search tool that utilizes both MILP and SAT to evaluate the upper bounds on the algebraic degree of `AO` ciphers. Our tool builds upon the general monomial prediction technique and is applied to a range of `AO` ciphers over $\mathbb{F}_{2^n}$. Through our approach, we are able to obtain upper tighter bounds on the algebraic degree. While our tool may not match the efficiency of Liu et al.'s coefficient group technique in evaluating the algebraic degree, it enables us to achieve upper tighter bounds. This represents a trade-off between efficiency and accuracy. However, our tool contributes significantly to enhancing the understanding of `AO` ciphers among cryptographic designers and analysts. While our method can derive tighter upper bounds on the algebraic degree, as noted in Liu et al.'s paper, the monomial prediction technique offers limited insight into factors driving the growth of the algebraic degree.

**Table 6** Comparison of the algebraic degree upper bounds obtained by the methods in this paper with the actual algebraic degrees.

| **Chaghri** | | | | | | |
|---|---|---|---|---|---|---|
| $(n, d, h_1, h_2)$ | Methods | $r$ steps | | | | |
| | | 1 | 2 | 3 | 4 | 5 |
| $(63, 32, 3)$ | our bound | 2 | 3 | 5 | 7 | 9 |
| | actual degree | 2 | 3 | 5 | 7 | 9 |
| $(63, 32, 0, 3)$ | our bound | 2 | 4 | 8 | 14 | 24 |
| | actual degree | 2 | 4 | 8 | 14 | 24 |
| $(63, 32, 0, 6)$ | our bound | 2 | 4 | 8 | 14 | 24 |
| | actual degree | 2 | 4 | 8 | 14 | 24 |
| $(63, 32, 0, 36)$ | our bound | 2 | 4 | 8 | 14 | 24 |
| | actual degree | 2 | 4 | 8 | 14 | 24 |

**Discussion of the MILP, SMT, and SAT methods.** We think it difficult to give a comprehensive comparison between MILP, SMT, and SAT methods. However, we attempted to use different modeling methods on the same `AO` ciphers with the same computational resources and recorded the time taken. The time comparisons can be found in Table 2. From the experiments, we observed that when using the SMT method to evaluate `AO` ciphers featuring complex affine layers, such as **Chaghri**, the model solving time increases significantly starting from six rounds. Therefore, we believe that the complex affine layers might be affecting the solving time of the SMT model in this study. Both SMT and SAT methods have shorter solving times than the MILP method, largely because the POWER and COPY operations involve modular addition operations. This seems to indicate that SMT and SAT models are more suitable for handling ciphers with modular addition operations in the propagation rule modeling. Furthermore, recent studies [39], [40] have shown that automated modeling methods based on SMT and SAT are better suited for handling ciphers with modular addition operations.

An intriguing future direction is to explore an efficient yet highly accurate method applicable to all `AO` ciphers over $\mathbb{F}_{2^n}$. This understanding has the potential to facilitate the construction of higher-order differential attacks, which are known as conditional higher-order differential attacks.

**Acknowledgments**
This work is supported by the National Key Research and Development Program of China (2022YFB2701900), the National Natural Science Foundation of China (Grants 62472172 and 62072181), and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

**References**

[1] M.R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity," Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I, ed. J.H. Cheon and T. Takagi, Lecture Notes in Computer Science, vol.10031, pp.191–219, 2016.

[2] M.R. Albrecht, C. Cid, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, and M. Schofnegger, "Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC," Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III, ed. S.D. Galbraith and S. Moriai, Lecture Notes in Computer Science, vol.11923, pp.371–397, Springer, 2019.

[3] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec, "Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols," IACR Trans. Symmetric Cryptol., vol.2020, no.3, pp.1–45, 2020.

[4] C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters, "Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields," Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II, ed. A. Canteaut and F. Standaert, Lecture Notes in Computer Science, vol.12697, pp.3–34, Springer, 2021.

[5] L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch, "Reinforced Concrete: A Fast Hash Function for Verifiable Computation," Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, ed. H. Yin, A. Stavrou, C. Cremers, and E. Shi, pp.1323–1335, ACM, 2022.

[6] T. Ashur, M. Mahzoun, and D. Toprakhisar, "Chaghri - A FHE-friendly Block Cipher," Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022, ed. H. Yin, A. Stavrou, C. Cremers, and E. Shi, pp.139–150, ACM, 2022.

[7] C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems, "New design techniques for efficient arithmetization-oriented hash functions: ttanemoi permutations and ttjive compression mode," Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III, ed. H. Handschuh and A. Lysyanskaya, Lecture Notes in Computer Science, vol.14083, pp.507–539, Springer, 2023.

[8] L. Grassi, S. Onofri, M. Pedicini, and L. Sozzi, "Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over Fnp Application to Poseidon," IACR Trans. Symmetric Cryptol., vol.2022, no.3, pp.20–72, 2022.

[9] L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch, "From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications," Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV, ed. C. Hazay and M. Stam, Lecture Notes in Computer Science, vol.14007, pp.255–286, Springer, 2023.

[10] T. Ashur and S. Dhooghe, "MARVELlous: a STARK-Friendly Family of Cryptographic Primitives." Cryptology ePrint Archive, Paper 2018/1098, 2018. https://eprint.iacr.org/2018/1098.

[11] M.R. Albrecht, L. Grassi, L. Perrin, S. Ramacher, C. Rechberger, D. Rotaru, A. Roy, and M. Schofnegger, "Feistel Structures for MPC, and More," Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II, ed. K. Sako, S.A. Schneider, and P.Y.A. Ryan, Lecture Notes in Computer Science, vol.11736, pp.151–171, Springer, 2019.

[12] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger, "On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy," Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II, ed. A. Canteaut and Y. Ishai, Lecture Notes in Computer Science, vol.12106, pp.674–704, Springer, 2020.

[13] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems," 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, ed. M. Bailey and R. Greenstadt, pp.519–535, USENIX Association, 2021.

[14] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings, ed. A. Menezes and S.A. Vanstone, Lecture Notes in Computer Science, vol.537, pp.2–21, Springer, 1990.

[15] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings, ed. T. Helleseth, Lecture Notes in Computer Science, vol.765, pp.386–397, Springer, 1993.

[16] X. Lai, Higher Order Derivatives and Differential Cryptanalysis, pp.227–233, Springer US, Boston, MA, 1994.

[17] L.R. Knudsen, "Truncated and Higher Order Differentials," Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings, ed. B. Preneel, Lecture Notes in Computer Science, vol.1008, pp.196–211, Springer, 1994.

[18] T. Jakobsen and L.R. Knudsen, "The Interpolation Attack on Block Ciphers," Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings, ed. E. Biham, Lecture Notes in Computer Science, vol.1267, pp.28–40, Springer, 1997.

[19] M. Eichlseder, L. Grassi, R. Lüftenegger, M. Øygarden, C. Rechberger, M. Schofnegger, and Q. Wang, "An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC," Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I, ed. S. Moriai and H. Wang, Lecture Notes in Computer Science, vol.12491, pp.477–506, Springer, 2020.

[20] C. Cid, L. Grassi, A. Gunsing, R. Lüftenegger, C. Rechberger, and M. Schofnegger, "Influence of the Linear Layer on the Algebraic Degree in SP-Networks," IACR Trans. Symmetric Cryptol., vol.2022, no.1, pp.110–137, 2022.

[21] C. Bouvier, A. Canteaut, and L. Perrin, "On the algebraic degree of iterated power functions," Des. Codes Cryptogr., vol.91, no.3, pp.997–1033, 2023.

[22] J. Cui, K. Hu, M. Wang, and P. Wei, "On the Field-Based Division Property: Applications to MiMC, Feistel MiMC and GMiMC," Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III, ed. S. Agrawal and D. Lin, Lecture Notes in Computer Science, vol.13793, pp.241–270, Springer, 2022.

[23] K. Hu, S. Sun, M. Wang, and Q. Wang, "An Algebraic Formulation of the Division Property: Revisiting Degree Evaluations, Cube Attacks, and Key-Independent Sums," Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I, ed. S. Moriai and H. Wang, Lecture Notes in Computer Science, vol.12491,

pp.446–476, Springer, 2020.
[24] T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer, "Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems," Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III, ed. D. Micciancio and T. Ristenpart, Lecture Notes in Computer Science, vol.12172, pp.299–328, Springer, 2020.
[25] F. Liu, R. Anand, L. Wang, W. Meier, and T. Isobe, "Coefficient Grouping: Breaking Chaghri and More," Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV, ed. C. Hazay and M. Stam, Lecture Notes in Computer Science, vol.14007, pp.287–317, Springer, 2023.
[26] F. Liu, L. Grassi, C. Bouvier, W. Meier, and T. Isobe, "Coefficient Grouping for Complex Affine Layers," Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III, ed. H. Handschuh and A. Lysyanskaya, Lecture Notes in Computer Science, vol.14083, pp.540–572, Springer, 2023.
[27] C. Sinz, "Towards an Optimal CNF Encoding of Boolean Cardinality Constraints," Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings, ed. P. van Beek, Lecture Notes in Computer Science, vol.3709, pp.827–831, Springer, 2005.
[28] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions, ed. T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, Department of Computer Science Report Series B, vol.B-2020-1, pp.51–53, University of Helsinki, 2020.
[29] L. Zhang, M. Liu, S. Li, and D. Lin, "Cryptanalysis of Ciminion," Information Security and Cryptology - 18th International Conference, Inscrypt 2022, Beijing, China, December 11-13, 2022, Revised Selected Papers, ed. Y. Deng and M. Yung, Lecture Notes in Computer Science, vol.13837, pp.234–251, Springer, 2022.
[30] C. Carlet, P. Charpin, and V.A. Zinoviev, "Codes, Bent Functions and Permutations Suitable For DES-like Cryptosystems," Des. Codes Cryptogr., vol.15, no.2, pp.125–156, 1998.
[31] Y. Hao, G. Leander, W. Meier, Y. Todo, and Q. Wang, "Modeling for Three-Subset Division Property Without Unknown Subset - Improved Cube Attacks Against Trivium and Grain-128AEAD," Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I, ed. A. Canteaut and Y. Ishai, Lecture Notes in Computer Science, vol.12105, pp.466–495, Springer, 2020.
[32] H. Hadipour and M. Eichlseder, "Integral cryptanalysis of WARP based on monomial prediction," IACR Trans. Symmetric Cryptol., vol.2022, no.2, pp.92–112, 2022.
[33] L. Sun, W. Wang, and M. Wang, "More accurate differential properties of LED64 and midori64," IACR Trans. Symmetric Cryptol., vol.2018, no.3, pp.93–123, 2018.
[34] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.
[35] M. Soos, K. Nohl, and C. Castelluccia, "Extending sat solvers to cryptographic problems," International Conference on Theory and Applications of Satisfiability Testing, pp.244–257, Springer, 2009.
[36] G. Bertoni, J. Daemen, M. Peeters, and G.V. Assche, "The Keccak SHA-3 submission." Submission to NIST (Round 3), 2011.
[37] M.J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.
[38] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "On the indifferentiability of the sponge construction," Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp.181–197, Springer, 2008.
[39] Y. Li, F. Liu, and G. Wang, "New records in collision attacks on SHA-2," Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I, ed. M. Joye and G. Leander, Lecture Notes in Computer Science, vol.14651, pp.158–186, Springer, 2024.
[40] Y. Li, F. Liu, and G. Wang, "Automating collision attacks on RIPEMD-160," IACR Trans. Symmetric Cryptol., vol.2023, no.4, pp.112–142, 2023.

**Jianqiang Ni** is currently working toward the PhD degree with the School of Software Engineering Institute, East China Normal University. His research interests include symmetric cryptanalysis and design.

**Gaoli Wang** received the B.S. degree in fundamental mathematics and the Ph.D. degree in information security from Shandong University, Jinan, China. She is currently a Professor with the Software Engineering Institute, East China Normal University. Her research interests include cryptography, computer, and network security.

**Yingxin Li** is currently working toward the PhD degree with the School of Software Engineering Institute, East China Normal University. His research interests include symmetric cryptanalysis and design.

**Siwei Sun** received the B.S. degree in Beijing University Of Technology and the Ph.D. degree in University of Chinese Academy of Sciences, Beijing, China. He is currently a Professor with the School of Cryptology, University of Chinese Academy of Sciences. His research interests include automation of symmetric cryptographic algorithm design and analysis, optimization and secure implementation of cryptographic algorithms, and symmetric cryptanalysis based on quantum computing.