An Improved Method for Evaluating Secret Variables and Its Application to WAGE

Weizhe Wang[1], Haoyang Wang[1], and Deng Tang[1(✉)]

Shanghai Jiao Tong University, Shanghai 200240, China
{SJTUwwz, haoyang.wang}@sjtu.edu.cn,dtang@foxmail.com

Abstract. The cube attack is a powerful cryptanalysis technique against symmetric ciphers, especially stream ciphers. The adversary aims to recover secret key bits by solving equations that involve the key. To simplify the equations, a set of plaintexts called a cube is summed up together. Traditional cube attacks use only linear or quadratic superpolies, and the size of cube is limited to an experimental range, typically around 40. However, cube attack based on division property, proposed by Todo et al. at CRYPTO 2017, overcomes these limitations and enables theoretical cube attacks on many lightweight stream ciphers. For a given cube I, they evaluate the set J of secret key bits involved in the superpoly and require $2^{|I|+|J|}$ encryptions to recover the superpoly. However, the secret variables evaluation method proposed by Todo et al. sometimes becomes unresponsive and fails to solve within a reasonable time. In this paper, we propose an improvement to Todo's method by breaking down difficult-to-solve problems into several smaller sub-problems. Our method retains the efficiency of Todo's method while effectively avoiding unresponsive situations. We apply our method to the WAGE cipher, an NLFSR-based authenticated encryption algorithm and one of the second round candidates in the NIST LWC competition. Specifically, we successfully mount cube attacks on 29-round WAGE, as well as on 24-round WAGE with a sponge constraint. To the best of our knowledge, this is the first cube attack against the WAGE cipher, which provides a more accurate characterization of the WAGE's resistance against algebraic attacks.

Keywords: Cube attack · Division property · WAGE · MILP.

1 Introduction

The cube attack, proposed by Dinur and Shamir [6], is an algebraic attack technique against stream ciphers. Its main idea is to obtain simple equations about the key by summing a set of plaintexts, called a cube C_I, and then solve these equations to retrieve partial key information. Cube attacks consist of two phases: the offline phase involves constructing proper cubes and recovering their corresponding superpolies, while the online phase involves encrypting $2^{|I|}$ plaintexts under the real key and summing up them to obtain values of the superpolies. Afterward, the equations are solved, and a brute-force search attack is performed

on the remaining key. It is vital to construct cubes in cube attacks. In traditional cube attacks, superpolies are often required to be linear or quadratic. Both properties can only be probabilistically determined through practical tests. This limits the size of cube to about 40.

Division property is an accurate characterization for the sum property of certain set, which was proposed by Todo [23]. In [24], Todo et al. combined cube attacks with division property. The link between division trail and the algebraic normal form (ANF) of superpoly was established. For a given cube I, they proposed an algorithm to evaluate the set J of secret variables involved in superpoly. A cube attack is available when the restriction $|I|+|J| < n$ is met, where n is the length of key. The combination of cube attacks and division property eliminates the size limitation of the cube and leads to theoretical cube attacks. The capability and applicability of cube attacks were greatly enhanced. Subsequently, more and more improved techniques have been proposed, such as flag technique [26], three-subset division property without unknown set [10], monomial prediction technique [12]. These developments have significantly improved the effectiveness and scope of cube attacks and have made it an important tool for attacking stream ciphers. In current research, cube attacks based on division property have primarily targeted simple stream ciphers such as Trivium [4] and Grain-128a [2]. To trace the propagation of division property effectively, original cryptoanalytic problems are often transformed into mixed-integer linear programming (MILP) problems and solved using optimizers. Some research [20,13,5] has also been conducted on other ciphers with intricate components. However, as the components of targeted cipher become complex, the model to be solved also becomes more complex. Thus, how to handle complex models becomes a crucial aspect in the research of cube attacks based on division property.

Authenticated Encryption with Associated Data (AEAD) is a type of encryption that provides both confidentiality and authenticity of data. It is commonly used to transmit confidential messages over insecure channels. In 2018, the National Institute of Standards and Technology (NIST) called for algorithms to be considered for lightweight cryptographic standards with AEAD and optional hashing functionalities. The aim of the competition was to identify and standardize lightweight cryptographic algorithms suitable for use in constrained environments, such as Internet of Things devices. A total of 57 candidates were submitted to NIST, and after selection, 32 submissions were chosen as Round 2 candidates. WAGE [3], submitted by AlTawy et al., was one of the 32 candidates. The WAGE permutation is used in the unified sponge-duplex mode to achieve the authenticated encryption functionality that provides 128-bit security with at most 2^{64} bits of allowed data per key. In [3], the designers analyzed the diffusion, algebraic, differential, and linear properties of the WAGE permutation and provided a clear security claim against differential and linear attacks. However, the description of WAGE's resistance to algebraic attacks is relatively subjective and lacks specific experiments to support it. In [7] , Fei et al. applied Correlation Power Analysis (CPA) technique to WAGE and recovered the key up to 12 out of 111 rounds.

In this paper, we conduct a detailed study of the secret variables evluation method, and explore the security level of WAGE against cube attacks. Compared to Trivium and Grain-128a, WAGE has a more complex feedback function and a larger state, making its corresponding MILP model more complex and challenging to solve. Applying Todo's method [24] directly to certain rounds of WAGE can result in unresponsiveness, which impedes the process of cube attacks. Thus, we propose an improved method to address this issue and successfully construct some useful cubes, as shown in Table 4. Our implementation is available in the following Github repository: https://github.com/SJTUwwz/WAGE_cube_attack.git. In particular, our contributions can be summarized in the following two aspects:

1. We improve the secret variables evaluation method proposed by Todo et al. [24]. In our improved method, we use a limit for solving time to judge whether the MILP problem is difficult to solve or not. If the time of solving process exceeds the limit, we break down the MILP problem into several sub-problems explicitly. This improvement effectively avoids the issue of unresponsiveness, which can enable higher round attacks.
2. Cube attacks are mounted on 29-round WAGE, as well as on 24-round WAGE with a sponge constraint with time complexity 2^{124}. This is the first cube attack on the initialization phase of WAGE and it provides a clear evaluation of the security level of WAGE by cube attacks.

The remainder of this paper is origanized as follows. In Section 2, we provide a review of the concept of cube attacks, division property, and fully linear integer inequality characterizations. We then introduce Todo's secret variables evaluation method in Section 3.1 and propose our improved method in Section 3.2. Section 4.1 offers a brief description of WAGE. In Section 4.2, we present a detailed description of cube attacks on the initialization phase of WAGE. Finally, Section 5 concludes the paper.

2 Preliminaries

In this paper, we will use the following notations. Let $\mathbb{F}_2$ be the field with two elements $\{0,1\}$ and $\mathbb{F}_2^n$ be the vector space of n-tuple over $\mathbb{F}_2$. We use bold italic letters to represent bit vectors, $\mathbf{0}$ and $\mathbf{1}$ represent bit vectors with all coordinates being 0 and 1, respectively. We denote $\boldsymbol{u} = (u_0, u_1, \ldots, u_{n-1}) \in \mathbb{F}_2^n$ an n-bit vector over $\mathbb{F}_2$. The Hamming weight of $\boldsymbol{u}$ is defined as $wt(\boldsymbol{u}) = \#\{0 \le i \le n-1 : u_i \neq 0\} = \sum_{i=0}^{n-1} u_i$. We use $\oplus$ (resp. $\bigoplus$) to denote the addition (resp. multiple sums) in $\mathbb{F}_2$ or $\mathbb{F}_2^n$. Moreover, $\boldsymbol{u} \succeq \boldsymbol{v}$ represents $u_i \ge v_i$ for all $i \in \{0, 1, \ldots, n-1\}$. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function in $\mathbb{F}_2[x_0, x_1, \ldots, x_{n-1}]/(x_0^2 - x_0, x_1^2 - x_1, \ldots, x_{n-1}^2 - x_{n-1})$ whose algebraic normal form (ANF) is

$$f(\boldsymbol{x}) = f(x_0, x_1, \ldots, x_{n-1}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \boldsymbol{x}^{\boldsymbol{u}},$$

where $a_{\boldsymbol{u}} \in \mathbb{F}_2$ and $\boldsymbol{x}^{\boldsymbol{u}} = \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{n-1} x_i^{u_i}$ is called a monomial.

2.1 Cube Attack

The cube attack [6] was first introduced by Dinur and Shamir at EUROCRYPT 2009, which is the extension of the higher-order differential cryptanalysis [15] and integral cryptanalysis [14]. Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{v} = (v_0, v_1, \ldots, v_{m-1})$ be n secret variables and m public variables, respectively. Then we can represent the symmetric-key cryptosystem as $f(\boldsymbol{x}, \boldsymbol{v})$, where f is a function from $\mathbb{F}_2^{n+m}$ to $\mathbb{F}_2$, $\boldsymbol{x}$ is the secret variable (key), $\boldsymbol{v}$ is the public variable (initialization vector (IV) or nonce). The main idea of cube attacks is to simplify the polynomial $f(\boldsymbol{x}, \boldsymbol{v})$ by computing its higher order differential on public variables $\boldsymbol{v}$. Let $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$ be a subset of public variables. Then function f can be rewritten as

$$f(\boldsymbol{x}, \boldsymbol{v}) = t_I \cdot p_I(\boldsymbol{x}, \boldsymbol{v}) \oplus q_I(\boldsymbol{x}, \boldsymbol{v}),$$

where $t_I = \prod_{v \in I} v$. As noted in [6], after the summation of the 2^d values of f by assigning all the possible values to d variables in I, the value of p_I is computed, that is

$$\bigoplus_{(v_{i_1}, v_{i_2}, \ldots, v_{i_d}) \in C_I} f(\boldsymbol{x}, \boldsymbol{v}) = p_I(\boldsymbol{x}, \boldsymbol{v}).$$

The public variables in I are called cube variables and the remaining public variables are called non-cube variables. The set C_I that contains all 2^d assignments of cube variables is called a cube. The dimension of C_I is d and all the non-cube variables are set to constants, usually all zeros. The simplified polynomial p_I is called the superpoly of C_I in f. In the absence of ambiguity, we would call p_I the superpoly of I in f for convenience.

The cube attack consists of the offline phase and the online phase. In the offline phase, attackers need to find cubes whose superpoly is useful. Useful superpolies can be used to directly recover partial information of key or help to filter out wrong keys. In the online phase, attackers would query the encryption oracle to get the cube summation under the real key for each cube obtained in the offline phase. By solving the equations of superpolies, we can recover some key bits. Finally, a brute-force attack would be applied to recover the whole key.

2.2 Division Property

Division property was first proposed in [23]. Then, the conventional bit-based division property and the bit-based division property using three subsets were introduced in [25]. Three-subset division property without unknown subset was introduced in [10]. In this paper, we will focus on the conventional bit-based division property. The definition of the conventional bit-based division property is as follows.

Definition 1 (Bit-Based Division Property). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$, and $\mathbb{K}$ be a set whose elements take an n-dimensional*

bit vector. When the multiset $\mathbb{X}$ *has the division property* $\mathcal{D}_{\mathbb{K}}^{1^n}$*, it fulfils the following conditions:*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \begin{cases} \textit{unknown}, & \textit{if there are } \boldsymbol{k}\in\mathbb{K} \textit{ s.t. } \boldsymbol{u}\succeq\boldsymbol{k}, \\ 0, & \textit{otherwise.} \end{cases}$$

Assume that $\mathbb{X}$ is the input set with the division property $\mathcal{D}_{\mathbb{K}_0}^{1^n}$. Let $\mathbb{Y}$ be the output set obtained by encrypting r rounds on $\mathbb{X}$. It is difficult to evaluate the division property of $\mathbb{Y}$, denoted by $\mathcal{D}_{\mathbb{K}_r}^{1^n}$, directly. In [25,28], the propagation rules of division property on basic operations were given. Based on those rules, $\mathcal{D}_{\mathbb{K}_0}^{1^n}$ can be computed by iteratively evaluating the propagation on round function. The concept of division trail was first introduced in [28], which facilitates the application of the MILP method to the division property and is defined as follows.

Definition 2 (Division Trail). *Consider the propagation of the division property* $\{\boldsymbol{k}\} = \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r$*. For any vector* $\boldsymbol{k}_i^*$ *in* $\mathbb{K}_i$ $(i \geq 1)$*, there must exist an vector* $\boldsymbol{k}_{i-1}^*$ *in* $\mathbb{K}_{i-1}$ *such that* $\boldsymbol{k}_{i-1}^*$ *can propagate to* $\boldsymbol{k}_i^*$ *by division property propagation rules. Furthermore, for* $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$*, if* $\boldsymbol{k}_{i-1}$ *can propagate to* $\boldsymbol{k}_i$ *for all* $i \in \{1, 2, \ldots, r\}$*, we call* $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r)$ *an* r*-round division trail.*

In [28], the authors described the propagation rules for AND, COPY, and XOR using MILP models. For detailed information, please refer to [28].

In [24], the authors applied the division property to cube attacks. Instead of finding zero-sum integral distinguishers, they used the division property to analyze the ANF coefficients of a Boolean function f. The relation between division property and ANF coefficient is given in the following lemma and proposition.

Lemma 1. *Let* $f(\boldsymbol{x})$ *be a boolean function from* $\mathbb{F}_2^n$ *to* $\mathbb{F}_2$ *and* $a_{\boldsymbol{u}}^f \in \mathbb{F}_2(\boldsymbol{u} \in \mathbb{F}_2^n)$ *be the ANF coefficients. Let* $\boldsymbol{k}$ *be an* n*-dimensional bit vector. Then, assuming there is no division trail such that* $\boldsymbol{k} \xrightarrow{f} 1$*,* $a_{\boldsymbol{u}}^f$ *is always* 0 *for* $\boldsymbol{u} \succeq \boldsymbol{k}$*.*

Proposition 1. *Let* $f(\boldsymbol{x}, \boldsymbol{v})$ *be a boolean function from* $\mathbb{F}_2^{n+m}$ *to* $\mathbb{F}_2$*, where* $\boldsymbol{x}$ *and* $\boldsymbol{v}$ *denote the secret and public variables, respectively. For a set of indices* $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{1, 2, \ldots, m\}$*, let* C_I *be a set of* $2^{|I|}$ *values where the variables in* $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$ *are taking all possible combinations of values. Let* $\boldsymbol{k}_I$ *be an* m*-dimensional bit vector such that* $\boldsymbol{v}^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \cdots v_{i_{|I|}}$*, i.e.* $k_i = 1$ *if* $i \in I$ *and* $k_i = 0$ *otherwise. Assuming there is no division trail such that* $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$*,* x_j *is not involved in the superpoly of the cube* C_I*.*

Based on Lemma 1 and Proposition 1, attackers can find all related secret variables J in the superpoly $p_I(\boldsymbol{x}, \boldsymbol{v})$. Then, the superpoly could be recovered with time complexity $2^{|I|+|J|}$. Degree estimation and term enumeration techniques were introduced in [26]. With these thechniques, the time complexity of recovering the superpoly can be reduced to $2^{|I|} \times \binom{|J|}{\leq d}$, where d is the upper bound of the degree of the superpoly.

2.3 Full Linear Integer Inequality Characterization

To apply MILP to cryptanalytic problems, attackers need to describe the fundamental components of ciphers using linear inequalities. This leads to the concept of full linear integer inequality characterization.

Definition 3 (FLIIC [8]). *Let $S \subset \mathbb{Z}_2^n$ and L be a set of linear integer inequalities:*

$$\begin{cases} a_{0,0}x_0 + a_{0,1}x_1 + \cdots + a_{0,n-1}x_{n-1} + b_0 \geq 0 \\ a_{1,0}x_0 + a_{1,1}x_1 + \cdots + a_{1,n-1}x_{n-1} + b_1 \geq 0 \\ \qquad\qquad\qquad \vdots \\ a_{m-1,0}x_0 + a_{m-1,1}x_1 + \cdots + a_{m-1,n-1}x_{n-1} + b_{m-1} \geq 0 \end{cases},$$

where $a_{i,j}$ and b_i are integers for $0 \leq i \leq m-1$, $0 \leq j \leq n-1$. L is called a full linear integer inequality characterization (FLIIC, in short) of S if the solution set of L on $\mathbb{Z}_2^n$ is S exactly. We also say L fully characterizes S, and m is called the cardinality of L, denoted by $|L|$.

The FLIIC of a complex component like the S-box can be obtained by directly combining basic operations. However, this method would result in a large number of inequalities and variables, making the model inefficient to solve. A better approach is to construct the FLIIC directly based on the given set S. The entire problem can be solved in two steps. The first step involves generating a sufficient number of high-quality inequalities. Then, in the second step, redundancies are removed and a minimal number of inequalities are selected.

In the first step, inequalities can be constructed using the H-representation method proposed in [21], the logical condition method described in [1], or the SuperBall method introduced in [16]. In the second step, we encounter a Set Covering Problem (SCP). In existing research, SCP is typically solved using either a greedy algorithm or MILP techniques [19]. In [8], the authors established a complete theoretical system to solve the problem of fully characterizing a given set with the minimal number of inequalities. They provided an algorithm of enumerating all plain closures for a given S-box, which supports point sets with high dimension up to 18 and is the fastest at present.

3 Evaluating Secret Variables in Superpoly

The critical step in cube attacks is constructing useful cubes during the offline phase. Our objective is to construct cubes that satisfy the condition $|I|+|J| < n$. In this section, we begin by presenting the secret variables evaluation method based on division property, which was initially proposed in [24]. Subsequently, we introduce *our improved secret variables evaluation method* and provide a detailed explanation of why it is an improvement over Todo's method.

3.1 Previous Secret Variables Evaluation Method

Based on the insights provided by Lemma 1 and Proposition 1, Todo et al. [24] introduced a framework that enables the evaluation of secret variables associated with the superpoly of a given cube. Algorithm 1 is the concrete algorithm supported by MILP.

Algorithm 1 Evaluate secret variables based on division property

```
1: procedure DPEval(MILP model M, cube indices I)
2:     Let x be n MILP variables of M corresponding to secret variables.
3:     Let v be m MILP variables of M corresponding to public variables.
4:     M.con ← v_i = 1 for all i ∈ I
5:     M.con ← v_i = 0 for all i ∈ ({0, 1, ..., m − 1} − I)
6:     M.con ← Σ_{i=0}^{n−1} x_i = 1
7:     J = ∅
8:     solve MILP model M
9:     while M is solved do
10:        pick index j ∈ {0, ..., n − 1} s.t. x_j = 1
11:        J = J ∪ {j}
12:        M.con ← x_j = 0
13:        solve MILP model M
14:    end while
15:    return J
16: end procedure
```

The input model M is the MILP model of target cipher by the context of division property and the cube is represented as a set I. In line 4-6, the input division property is contrained to the form $(\boldsymbol{e}_j, \boldsymbol{k}_I)$. Line 7-14 is the core idea of Algorithm 1. When model M is feasible, a satisfying division trail $(\boldsymbol{e}_j, \boldsymbol{k}_I)$ can be found, which means that x_j is involved in the superpoly. Then, the index j is added to the set J, and a new constraint is included to exclude this specific point. The process is repeated until the model becomes infeasible. Finally, the set J containing the indices of the secret variables involved in the superpoly is obtained.

3.2 An improved Secret Variables Evaluation Method

In addition to Todo's method, it is evident that a more intuitive approach exists. This alternative method is to verify $(\boldsymbol{e}_j, \boldsymbol{k}_I)$ input division property for all $0 \leq j \leq m-1$ individually. The details of this intuitive method are presented in Algorithm 2.

When comparing Algorithm 1 and Algorithm 2, it is expected that Algorithm 1 will outperform Algorithm 2, since the former's iteration count is $|J| \leq n$ while the latter's iteration count is n. However, there is another important factor to consider. That is, the MILP problems that need to be solved during the iteration

Algorithm 2 Evaluate each secret variables individually

```
1: procedure INDEval(MILP model M, cube indices I)
2:     Let x be n MILP variables of M corresponding to secret variables.
3:     Let v be m MILP variables of M corresponding to public variables.
4:     M.con ← v_i = 1 for all i ∈ I
5:     M.con ← v_i = 0 for all i ∈ ({0, 1, ..., m − 1} − I)
6:     M.con ← Σ_{i=0}^{n−1} x_i = 1
7:     J = ∅
8:     for j from 0 to n − 1 do
9:         M.con ← x_j = 1
10:        solve MILP model M
11:        if M is solved then
12:            J = J ∪ {j}
13:        end if
14:        M.con → x_j = 1                ▷ "→" means removing constraint.
15:    end for
16:    return J
17: end procedure
```

process are not the same. More precisely, the constraints on input secret variables differ between the two methods. Specially, in the i-th iteration, Todo's method has constraints

$$\sum_{j=0}^{n-1} x_j = 1 \text{ and } x_j = 0, \forall j \in J,$$

while the intuitive method has constraints

$$x_i = 1 \text{ and } x_j = 0, \forall j \neq i, j \in \{0, 1, \ldots, n\}.$$

It is clear that the latter's constraints are stronger. This difference will also have an impact on the efficiency of algorithms. In order to compare the efficiency of the two methods, we conducted a series of experiments. The results indicated that in marjority of cases, Todo's method required less time than the intuitive method. Nevertheless, in certain situations, Todo's method was unable to produce results within a reasonable timeframe. Upon further investigation, we discovered that such issue was due to the difficulty in optimizing the MILP model after eliminating all feasible solutions. Moreover, this problem did not arise when using intuitive method. By combining the strengths of Todo's method and intuitive method, we have developed an improved secret variables evaluation method, which not only maintains the high efficiency, but also avoids the issue of unresponsiveness. Algorithm 3 gives the details of our new method.

Compared to Todo's method, our improved method makes use of the $TimeLimit$ parameter of Gurobi solver. Lines 2-15 correspond to the first part of procedure. In this part, Todo's method is applied directly. If the solver has not stopped when time is up, the model-solving process is forcibly terminated. Then, the unsolved set $\bar{J}$ will be solved by intuitive method one by one. This is the second

Algorithm 3 An improved secret variables evaluation method

procedure NEWEVAL(MILP model M, cube indices I, limit of time t)
  Let $\boldsymbol{x}$ be n MILP variables of M corresponding to secret variables.
  Let $\boldsymbol{v}$ be m MILP variables of M corresponding to public variables.
  $M.TimeLimit \leftarrow t$
  $M.con \leftarrow v_i = 1$ for all $i \in I$
  $M.con \leftarrow v_i = 0$ for all $i \in (\{0, 1, \ldots, m-1\} - I)$
  $M.con \leftarrow \sum_{i=0}^{n-1} x_i = 1$
  $J = \emptyset$
  solve MILP model M
  while M is solved **do**
    pick index $j \in \{0, \ldots, n-1\}$ s.t. $x_j = 1$
    $J = J \cup \{j\}$
    $M.con \leftarrow x_j = 0$
    solve MILP model M
  end while
  if M is not solved within t **then** ▷ The MILP model is difficult to solve.
    $\bar{J} = \{0, 1, \ldots, n-1\} - J$
    for all $j \in \bar{J}$ **do**
      $M.con \leftarrow x_j = 1$
      solve MILP model M
      if M is solved **then**
        $J = J \cup \{j\}$
      end if
      $M.con \rightarrow x_j = 1$
    end for
  end if
  return J
end procedure

part of procedure, corresponding to lines 16-26. Since the MILP model in the intuitive method is simpler, it is hopeful that the problem can be solved within a reasonable time. Actually, we manually divide the original complex MILP model into several simpler MILP models and solve them individually in the second part of the process. A similar idea was used to recover superpoly in [11].

If the MILP model M in the first part can be solved within the time limit, the second part of Algorithm 3 will not be activated. This ensures that our method maintains high efficiency, similar to Todo's method in most cases. In situations where Todo's method does not perform well, our method will continue to work with simpler MILP models. In summary, our new method represents a significant improvement over Todo's method.

4 Experiment and Result

We apply our new method to the lightweight authenticated cipher WAGE [3]. In this section, we will first provide a concise description of WAGE. Subsequently, we will present the details of our experiments and the corresponding results.

4.1 Description of WAGE

The authenticated cipher WAGE-$\mathcal{AE}$-128 is built upon the WAGE permutation, which is specifically designed to be lightweight and hardware-friendly. The WAGE permutation operates on a state size of 259 bits over the finite field $\mathbb{F}_{2^7}$. The design of the WAGE function adopts the structure of the (nonlinear) initialization phase of the WG stream cipher family [18]. The designer of WAGE claims that the permutation achieves full bit diffusion in 28 rounds. Due to the complex nonlinear feedback function, the algebraic degree in WAGE grow rapidly. As a result, WAGE exhibits robust resistance against algebraic attacks, such as integral and cube attacks.

The WAGE permutation The core components of the WAGE permutation include two different S-boxes (WGP and SB) defined over $\mathbb{F}_{2^7}$, a nonlinear feedback, five word-wise XORs, and a pair of 7-bit round constants (rc_1^i, rc_0^i). Fig. 1 presents a high-level overview of the round function of the WAGE permutation. The state consists of 37 7-bit words and is denoted by $\boldsymbol{S}^i = (\boldsymbol{S}_{36}^i, \ldots, \boldsymbol{S}_0^i)$ at the beginning of i-th round. The round function takes as inputs the current state $\boldsymbol{S}^i$ and the round constant tuple (rc_1^i, rc_0^i), and updates the state in a Galois NLFSR fashion with the following three steps:

1. Computing linear feedback. The feedback computation is given by

$$\boldsymbol{fb} = \boldsymbol{S}_{31}^i \oplus \boldsymbol{S}_{30}^i \oplus \boldsymbol{S}_{26}^i \oplus \boldsymbol{S}_{24}^i \oplus \boldsymbol{S}_{19}^i \oplus \boldsymbol{S}_{13}^i \oplus \boldsymbol{S}_{12}^i \oplus \boldsymbol{S}_{8}^i \oplus \boldsymbol{S}_{6}^i \oplus (\omega \otimes \boldsymbol{S}_0^i).$$

where the representation of $\omega \otimes \boldsymbol{x}$ is given by

$$\omega \otimes (x_0, x_1, x_2, x_3, x_4, x_5, x_6) \rightarrow (x_6, x_0 \oplus x_6, x_1 \oplus x_6, x_2 \oplus x_6, x_3, x_4, x_5).$$

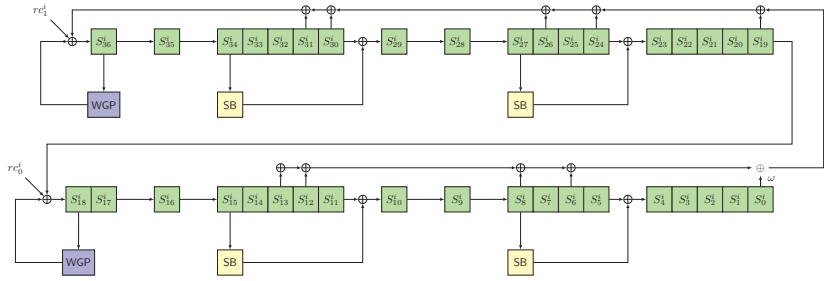

Fig. 1. A block diagram of the WAGE's round function

2. Updating intermediate words and adding round constants.

$$\boldsymbol{S}_5^i \leftarrow \boldsymbol{S}_5^i \oplus SB(\boldsymbol{S}_8^i), \boldsymbol{S}_{11}^i \leftarrow \boldsymbol{S}_{11}^i \oplus SB(\boldsymbol{S}_{15}^i), \boldsymbol{S}_{19}^i \leftarrow \boldsymbol{S}_{19}^i \oplus WGP(\boldsymbol{S}_{18}^i) \oplus rc_0^i$$

$$\boldsymbol{S}_{24}^i \leftarrow \boldsymbol{S}_{24}^i \oplus SB(\boldsymbol{S}_{27}^i), \boldsymbol{S}_{30}^i \leftarrow \boldsymbol{S}_{30}^i \oplus SB(\boldsymbol{S}_{34}^i), \boldsymbol{fb} \leftarrow \boldsymbol{fb} \oplus WGP(\boldsymbol{S}_{36}^i) \oplus rc_1^i.$$

3. Shifting the register contents and update the last word.

$$\boldsymbol{S}_j^{i+1} \leftarrow \boldsymbol{S}_{j+1}^i, j \in 0, 1, \ldots, 35,$$

$$\boldsymbol{S}_{36}^{i+1} \leftarrow \boldsymbol{fb}.$$

The WGP S-box is a unique Welch-Gong permutation that achieves low differential uniformity and high nonlinearity [9,17]. The SB S-box is constructed iteratively using nonlinear transformations and bit permutations. The maximum algebraic degree of both S-boxes is 6. The hexadecimal representations of the WGP and SB S-boxes are provided in Table 1 and Table 2, respectively, in a row-major order. Round constants are generated by an LFSR of length 7 with feedback polynomial $x^7 + x + 1$. The WAGE permutation contains 111 rounds in total.

Table 1. WGP's hexadecimal representation

00	12	0a	4b	66	0c	48	73	79	3e	61	51	01	15	17	0e
7e	33	68	36	42	35	37	5e	53	4c	3f	54	58	6e	56	2a
1d	25	6d	65	5b	71	2f	20	06	18	29	3a	0d	7a	6c	1b
19	43	70	41	49	22	77	60	4f	45	55	02	63	47	75	2d
40	46	7d	5c	7c	59	26	0b	09	03	57	5d	27	78	30	2e
44	52	3b	08	67	2c	5	6b	2b	1a	21	38	07	0f	4a	11
50	6a	28	31	10	4d	5f	72	39	16	5a	13	04	3c	34	1f
76	1e	14	23	1c	32	4e	7b	24	74	7f	3d	69	64	62	6f

Table 2. SB's hexadecimal representation

2e	1c	6d	2b	35	07	7f	3b	28	08	0b	5f	31	11	1b	4d
6e	54	0d	09	1f	45	75	53	6a	5d	61	00	04	78	06	1e
37	6f	2f	49	64	34	7d	19	39	33	43	57	60	62	13	05
77	47	4f	4b	1d	2d	24	48	74	58	25	5e	5a	76	41	42
27	3e	6c	01	2c	3c	4e	1a	21	2a	0a	55	3a	38	18	7e
0c	63	67	56	50	7c	32	7a	68	02	6b	17	7b	59	71	0f
30	10	22	3d	40	69	52	14	36	44	46	03	16	65	66	72
12	0e	29	4a	4c	70	15	26	79	51	23	3f	73	5b	20	5c

The authenticated cipher WAGE-$\mathcal{AE}$-128 WAGE operates in the unified sponge duplex mode to provide authenticated encryption with associated data functionality. The authenticated cipher WAGE-$\mathcal{AE}$-128 supports key, nonce, and tag sizes of 128 bits, and processes 64 bits per call of the WAGE permutation. The 259-bit internal state is represented as a string $(\boldsymbol{X}, \boldsymbol{Y})$, where $\boldsymbol{X}$ and $\boldsymbol{Y}$ denote the 64-bit rate and 195-bit capacity part of the state, respectively. Specifically, the rate part contains 9 words, $\boldsymbol{S}_{8,9,15,16,18,27,28,34,35}$, and the first bit of word $\boldsymbol{S}_{36}$. With a sponge constraint, we can only get the value of the rate part.

During the initialization of WAGE-$\mathcal{AE}$-128, the state is first loaded with a 128-bit nonce $\boldsymbol{N} = (n_0, \ldots, n_{127})$ and a 128-bit key $\boldsymbol{K} = (k_0, \ldots, k_{127})$. The key $\boldsymbol{K}$ is then divided into two key blocks, $\boldsymbol{K}_0 = k_0, \ldots, k_{63}$ and $\boldsymbol{K}_1 = k_{64}, \ldots, k_{127}$. The two key blocks are then absorbed into the state with the WAGE permutation applied each time. The steps of initialization are described as follows:

$$(\boldsymbol{X}, \boldsymbol{Y}) \leftarrow WAGE(load(\boldsymbol{N}, \boldsymbol{K}))$$
$$(\boldsymbol{X}, \boldsymbol{Y}) \leftarrow WAGE(\boldsymbol{X} \oplus \boldsymbol{K}_i, \boldsymbol{Y}), i = 0, 1,$$

The *load* function is explicitly given in Table 3. In Table 3, we denote $k_i, k_{i+1}, \ldots, k_{i+t}$ as k_{i-i+t}. In this paper, we will focus on the attack of the WAGE permutation and the initialization phase of WAGE-$\mathcal{AE}$-128. Consequently, we will not provide a detailed decription of other aspects of WAGE-$\mathcal{AE}$-128. Readers who are interested can refer to [3] for more information.

4.2 Cube Attacks on WAGE Using MILP

As previously mentioned, the cube attack consists of offline phase and online phase. The most challenging part of the attack lies in constructing useful cubes. A cube C_I is deemed *"useful"* when it statisfies the condition $|I|+|J| < n$, where $|I|$ is the dimension of cube, $|J|$ is the number of secret variables involved in its superpoly and n is the length of key. Once a useful cube is obtained, it can be leveraged to construct an effective cube attack. In this section, we will introduce how to construct a theoretical cube attack on WAGE based on our new method. All our experiments were completed on a PC (Intel Core i5-10400 CPU with 6 cores, 16 GB memory, Windows 11). The source code of this work can be found in https://github.com/SJTUwwz/WAGE_cube_attack.git.

Table 3. The $load(\boldsymbol{N}, \boldsymbol{K})$ procedure of WAGE-$\mathcal{AE}$-128.

$\boldsymbol{S}_0$	$\boldsymbol{S}_1$	$\boldsymbol{S}_2$	$\boldsymbol{S}_3$	$\boldsymbol{S}_4$	$\boldsymbol{S}_5$	$\boldsymbol{S}_6$	$\boldsymbol{S}_7$
k_{0-6}	k_{14-20}	k_{28-34}	k_{42-48}	k_{56-62}	k_{71-77}	k_{85-91}	k_{99-105}
$\boldsymbol{S}_8$	$\boldsymbol{S}_9$	$\boldsymbol{S}_{10}$	$\boldsymbol{S}_{11}$	$\boldsymbol{S}_{12}$	$\boldsymbol{S}_{13}$	$\boldsymbol{S}_{14}$	$\boldsymbol{S}_{15}$
$k_{113-119}$	n_{7-13}	n_{21-27}	n_{35-41}	n_{49-55}	n_{64-70}	n_{78-84}	n_{92-98}
$\boldsymbol{S}_{16}$	$\boldsymbol{S}_{17}$	$\boldsymbol{S}_{18}$				$\boldsymbol{S}_{19}$	$\boldsymbol{S}_{20}$
$n_{120-126}$	$n_{106-112}$	$k_{63}, k_{127}, n_{63}, n_{127}, 0, 0, 0$				k_{7-13}	k_{21-27}
$\boldsymbol{S}_{21}$	$\boldsymbol{S}_{22}$	$\boldsymbol{S}_{23}$	$\boldsymbol{S}_{24}$	$\boldsymbol{S}_{25}$	$\boldsymbol{S}_{26}$	$\boldsymbol{S}_{27}$	$\boldsymbol{S}_{28}$
k_{35-41}	k_{49-55}	k_{64-70}	k_{78-84}	k_{92-98}	$k_{106-112}$	$k_{120-126}$	n_{0-6}
$\boldsymbol{S}_{29}$	$\boldsymbol{S}_{30}$	$\boldsymbol{S}_{31}$	$\boldsymbol{S}_{32}$	$\boldsymbol{S}_{33}$	$\boldsymbol{S}_{34}$	$\boldsymbol{S}_{35}$	$\boldsymbol{S}_{36}$
n_{14-20}	n_{28-34}	n_{42-48}	n_{56-62}	n_{71-77}	n_{85-91}	n_{99-105}	$n_{113-119}$

Constructing MILP Model The first step of our attck is to construct MILP model simulating the propagation of division property. Generally, we will construct the MILP model round by round. The round function of WAGE consists of shift, XOR, ω, addition of round constants and S-box operations. The propagation on the first three operations can be fully characterized by two basic rules: COPY and XOR [28]. The corresponding constraint of ω operataion is shown in Algorithm 4. Furthermore, the addition of round constants would not affect the division property. Therefore, our focus is to find the FLIICs of nonlinear components, WGP and SB, to construct the characterization of the entire round function.

Algorithm 4 MILP model for the ω operation in WAGE

procedure ω(MILP model M, input variables $\boldsymbol{x}$, output variables $\boldsymbol{y}$)
    $M.var \leftarrow a_1, a_2, a_3, a_4$ as binary.
    $M.con \leftarrow a_1 + a_2 + a_3 + a_4 = x_6$
    $M.con \leftarrow y_0 = a_0$
    for $i \in \{4, 5, 6\}$ **do**
        $M.con \leftarrow y_i = x_{i-1}$
    end for
    for $i \in \{1, 2, 3\}$ **do**
        $M.con \leftarrow y_i = x_{i-1} + a_i$
    end for
end procedure

Firstly, we calculate the ANFs of WGP and SB using the Möbius transform. With the Algorithm 2 in [28], we can obtain the division trails of WGP and SB, which we denote as S_{WGP} and S_{SB}, respectively. Subsequently, we use SageMath [22] to generate the H-representations and candidate inequalities of S_{WGP} and S_{SB}. During this process, we obtain $3,204$ candidate inequalities for S_{WGP} and $400,781$ for S_{SB}. Given the size of the SCP problem, we employ

different techniques to compute the minimal FLIIC of each set. Specifically, we compute the minimal FLIIC of S_{WGP} with MILP techniques and find a FLIIC with 14 inequalities. For S_{SB}, we use a greedy algorithm and obtain a FLIIC with 46 inequalities. Based on the existing conditions, we can begin building the overall model of WAGE. For ease of description, we will use the notation $OP(inputvariable, outputvariable)$ to represent the specific constraints that need to be added for a given operation. The details of the process are shown in Algorithm 5.

In Algorithm 5, $\boldsymbol{A} - \boldsymbol{H}, \boldsymbol{Z}_1, \boldsymbol{Z}_2$ are temporary 7-bit variables. Lines 4-35 corresponds to the propagation of division trail through round function. The lines 5-15, 16-33, and 34 correspond to the three steps of the round function: linear feedback calculation, intermediate state update, and byte shifting, respectively. Upon this model, we can incorporate additional input constraints and objective functions to enable functionality such as algebraic degree estimation and secret variables evaluation.

Constructing Useful Cubes and Key Recovery In cube attacks, it is important to construct useful cubes with simple superpoly. Usually, superpoly with lower algebraic degrees tends to be simpler and more likely to satisfy $|I|+|J| < n$. A heuristic algorithm of constructing cube with linear superpoly was proposed in [30]. Moreover, they extended a small cube by adding "steep variable" and "gentle variable" properly. However, directly applying their method on WAGE is not practical due to the length of nonce. To address this challenge, we propose the idea of "steep word variable". The definition is given as follow.

Definition 4 (Steep Word Variable). *Let $I = \{\boldsymbol{v}_{i_1}, \boldsymbol{v}_{i_2}, \ldots, \boldsymbol{v}_{i_l}\}$ be a set containing $l \times w$ cube variables, where w is the length of word. Then, a word $\boldsymbol{t} \in B$ is called a steep word variable of I if*

$$ds(I \cup \{\boldsymbol{t}\}) = \min\{ds(I \cup \{\boldsymbol{v}\}) | \boldsymbol{v} \in B\},$$

where $B = \{\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1}\} \backslash I$ and $ds(I)$ is the degree of the superpoly of I.

We opt for the word-based version of "steep variable" for several reasons. Firstly, the nonce length of the WAGE is 128 bits, which leads to an unmanageably large search space if we include bit-by-bit. Secondly, the WAGE itself is word-based and searching at the word level is already sufficiently precise. Finally, from an analysis of division property, when the input division property of an S-box is **1**, its output division property will also be **1**, which facilitates a longer division trails.

In this paper, we would use the degree evaluation method based on division property to determine the steep word variable. Based on the MILP model constructed by Algorithm 5, we present Algorithm 6 to estimate the algebraic degree of superpoly on WAGE.

During the process of constructing cubes, we employ an algorithm similar to depth-first search. Specifically, in each iteration, we add the steep word variable to the cube. This process is similar to the first stage of the Algorithm 3 in [30].

Algorithm 5 MILP model for the R round WAGE permutation

procedure $WAGEModel$(The number of round R, the index of target bit $tindex$)
  Prepare empty MILP model M
  $M.var \leftarrow \boldsymbol{S}^0 = (S_0^0, \ldots, S_{258}^0)$ as binary ▷ $\boldsymbol{S}^0 = (\boldsymbol{S}_0^0, \ldots, \boldsymbol{S}_{36}^0)$
  for $r = 0$ to $R - 1$ **do**
    $M.var \leftarrow \boldsymbol{A} = (a_0, \ldots, a_6)$ as binary
    $M.con \leftarrow \omega(\boldsymbol{S}_0^0, \boldsymbol{A})$
    $XorExpr = [\boldsymbol{A}]$
    for $i \in \{6, 8, 12, 13, 19, 24, 26, 30, 31\}$ **do**
      $M.var \leftarrow \boldsymbol{B} = (b_0, \ldots, b_6), \boldsymbol{C} = (c_0, \ldots, c_6)$ as binary
      $M.con \leftarrow COPY(\boldsymbol{S}_i^r, (\boldsymbol{B}, \boldsymbol{C}))$
      $\boldsymbol{S}_i^r = \boldsymbol{B}$
      $XorExpr.append(\boldsymbol{C})$
    end for
    $M.var \leftarrow \boldsymbol{fb} = (fb_0, \ldots, fb_6)$ as binary
    $M.con \leftarrow XOR(XorExpr, \boldsymbol{fb})$
    for $(i_1, i_2) \in \{(5, 8), (11, 15), (24, 27), (30, 34)\}$ **do**
      $M.var \leftarrow \boldsymbol{D} = (d_0, \ldots, d_6), \boldsymbol{E} = (e_0, \ldots, e_6)$ as binary
      $M.con \leftarrow COPY(\boldsymbol{S}_{i_2}^r, (\boldsymbol{E}, \boldsymbol{F}))$
      $\boldsymbol{S}_{i_2}^r = \boldsymbol{E}$
      $M.var \leftarrow \boldsymbol{G} = (g_0, \ldots, g_6), \boldsymbol{H} = (h_0, \ldots, h_6)$
      $M.con \leftarrow SB(\boldsymbol{S}_{i_1}^r, \boldsymbol{G})$
      $M.con \leftarrow XOR((\boldsymbol{G}, \boldsymbol{F}), \boldsymbol{H})$
      $\boldsymbol{S}_{i_1}^r = \boldsymbol{H}$
    end for
    for $(\boldsymbol{Z}_1, \boldsymbol{Z}_2) \in \{(\boldsymbol{S}_{18}^r, \boldsymbol{S}_{19}^r), (\boldsymbol{fb}, \boldsymbol{S}_{36}^r)\}$ **do**
      $M.var \leftarrow \boldsymbol{D} = (d_0, \ldots, d_6), \boldsymbol{E} = (e_0, \ldots, e_6)$ as binary
      $M.con \leftarrow COPY(\boldsymbol{Z}_2, (\boldsymbol{E}, \boldsymbol{F}))$
      $\boldsymbol{Z}_2 = \boldsymbol{E}$
      $M.var \leftarrow \boldsymbol{G} = (g_0, \ldots, g_6), \boldsymbol{H} = (h_0, \ldots, h_6)$
      $M.con \leftarrow WGP(\boldsymbol{Z_1}, \boldsymbol{G})$
      $M.con \leftarrow XOR((\boldsymbol{G}, \boldsymbol{F}), \boldsymbol{H})$
      $\boldsymbol{Z}_1 = \boldsymbol{H}$
    end for
    $\boldsymbol{S}^{r+1} = (\boldsymbol{S}_1^r, \boldsymbol{S}_2^r, \ldots, \boldsymbol{S}_{36}^r, \boldsymbol{fb})$
  end for
  for i from 0 to 258 **do**
    if $i \neq tindex$ **then**
      $M.con \leftarrow \boldsymbol{S}_i^R = 0$
    else
      $M.con \leftarrow \boldsymbol{S}_i^R = 1$
    end if
  end for
end procedure

Algorithm 6 MILP model for the R round WAGE permutation

procedure $WAGEDeg$(cube indices I, The number of round R, the index of target bit $tindex$)
  $M \leftarrow WAGEModel(R, tindex)$
  $K \leftarrow \{0, 1, \ldots, 62\} \cup \{126, 127\} \cup \{133, 134, \ldots, 195\}$
  $N \leftarrow \{63, 64, \ldots, 125\} \cup \{128, 129\} \cup \{196, 197, \ldots, 258\}$
  $M.con \leftarrow S_i^0 = 1$ for all $i \in I$
  $M.con \leftarrow S_i^0 = 0$ for all $i \in N - I$
  for $i \in \{130, 131, 132\}$ **do**
    $M.con \leftarrow S_i^0 = 0$
  end for
  Set the objective function $M.obj \leftarrow \max \sum_{i \in K} S_i^0$
  Solve MILP model M
  return The solution of M
end procedure

This method will help us find the "path" of fastest algebraic degree descent. Unlike the approach in [30], we start from empty set directly. Table 3 indicates that the nonce words are $\boldsymbol{S}_{9-17}$ and $\boldsymbol{S}_{28-36}$, from which we can select cube variables only. Since WAGE adopts an NLFSR based design, the word at position 0 is mixed at a slower rate slower than others. With a sponge constraint, we can only get the value of $\boldsymbol{S}_{8,9,15,16,18,27,28,34,35}$, and the first bit of word $\boldsymbol{S}_{36}$. With division property, we successfully demonstrate that WAGE achieves full bit diffusion in 28 rounds. Besides, we also demonstrate that $\boldsymbol{S}_8$ achieves full bit diffusion in 23 rounds. Therefore, when constructing cube, we focus output position on $\boldsymbol{S}_0$ and $\boldsymbol{S}_8$. By applying our method, we successfully construct some useful cubes at $\boldsymbol{S}_0$ and $\boldsymbol{S}_8$ in 29 rounds and 24 rounds, respectively. The details of cubes are shown in Table 4.

Table 4. Useful cubes constructed by our method

Position	Round	Cube variables	Index of involved key bit	Deg	$\lvert J\rvert$	$\lvert I\rvert + \lvert J\rvert$
$\boldsymbol{S}_0$	29	$\boldsymbol{S}_{9-16}$	7-13,21-27,35-41,49-55,63-70, 78-84,92-98,106-112,120-127	48	65	121
		$\boldsymbol{S}_{9-15,17}$				
$\boldsymbol{S}_8$	24	$\boldsymbol{S}_{9-16}$				
		$\boldsymbol{S}_{9-15,17}$				

To compared our method with Todo's method, we conducted the same experiments using Todo's method. We found that Todo's method resulted in unresponsiveness when the cube was $\boldsymbol{S}_{9-15,17}$, the round number was 29 and the indices of target bit were 0 and 6. A similar situation occurred when the cube was

$\boldsymbol{S}_{9-16}$ and the indices of target bit were 2 and 5. This phenomenon demonstrates the superiority of our method.

From Table 4, we know that each useful cube satisfies $|I| + |J| = 121 < 128$. Moreover, we can compute 7 superpolies simutaneously because their involved key bits are the same. Therefore, we can obtain 7 superpolies with $2^{|I|} \times 7 \times \binom{|J|}{\leq d} \approx 2^{123.81}$ requests. With the assumption in [24], it is hopeful that we can recover 7 bits secret information from 7 superpolies. Although there is evidence [29,27] suggesting that the assumptions would not hold true in certain cases, it is worth nothing that these cases often occur when the superpoly has low algebraic degree and relates to few secret variables. However, the superpoly we found has high algebraic degree and numerous related secret variables, making it highly unlikely to degenerate into a constant function. Besides, it would be easy to find a non-cube constant that makes the superpoly a balanced function. Hence, we can use these useful cubes to implement cube attacks on WAGE.

For the sake of time complexity, when conducting cube attacks on WAGE, we only need to use one cube from the Table 4. During the offline phase, 7 balanced superpolies are obtained through $2^{123.81}$ requests. In the online phase, we can recover 7 bits secret infromation based on the superpolies and then perform a exhaustive search on the remaining 121 bits key information. The time complexity is $2^{123.81} + 2^{121} \approx 2^{124}$. In summary, we have successfully mounted cube attacks on 29-round WAGE, as well as on 24-round WAGE with a sponge constraint.

5 Conclusion

In this paper, we proposed an improved method for evaluating secret variables in cube attacks. Our method's improvement lies in explicitly breaking down difficult-to-solve problems into sub-problems, which helps to avoid the issue of unresponsiveness to a certain extent. As an application, we used our improved method to attack WAGE and successfully mounted two cube attacks on 29-round WAGE and 24-round WAGE with a sponge constraint. Although this result does not violate WAGE's security claims, it provides a clear security level of WAGE against cube attacks. We also believe that our improved method will facilitate the implementation of cube attacks on other ciphers.

Acknowledgements We are grateful to Xiutao Feng and Shengyuan Xu for their valuable suggestions on FLIIC. We also thank the anonymous reviewers for their helpful comments. The work of Deng Tang was supported in part by the National Key Research and Development Project 2020YFA0712300 and NSFC (No. 62272303).

References

1. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, A.M.: MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR

Trans. Symmetric Cryptol. **2017**(4), 99–129 (2017). https://doi.org/10.13154/tosc.v2017.i4.99-129
2. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. Int. J. Wirel. Mob. Comput. **5**(1), 48–59 (2011). https://doi.org/10.1504/IJWMC.2011.044106
3. AlTawy, R., Gong, G., Mandal, K., Rohit, R.: WAGE: an authenticated encryption with a twist. IACR Trans. Symmetric Cryptol. **2020**(S1), 132–159 (2020). https://doi.org/10.13154/tosc.v2020.iS1.132-159
4. Cannière, C.D., Preneel, B.: Trivium. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_18
5. Derbez, P., Fouque, P.: Increasing precision of division property. IACR Trans. Symmetric Cryptol. **2020**(4), 173–194 (2020). https://doi.org/10.46586/tosc.v2020.i4.173-194
6. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_16
7. Fei, Y., Gong, G., Gongye, C., Mandal, K., Rohit, R., Xu, T., Yi, Y., Zidaric, N.: Correlation power analysis and higher-order masking implementation of WAGE. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) Selected Areas in Cryptography - SAC 2020. LNCS, vol. 12804, pp. 593–614. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-81652-0_23
8. Feng, X., Tian, Y., Wang, Y., Xu, S., Zhang, A.: Full linear integer inequality characterization of set over $\mathbb{Z}_2^n$. CSTR:32003.36.ChinaXiv.202210.00055.V2 (2023), http://www.chinaxiv.org/abs/202210.00055
9. Gong, G., Youssef, A.M.: Cryptographic properties of the welch-gong transformation sequence generators. IEEE Trans. Inf. Theory **48**(11), 2837–2846 (2002). https://doi.org/10.1109/TIT.2002.804043
10. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 466–495. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_17
11. Hu, K., Sun, S., Todo, Y., Wang, M., Wang, Q.: Massive superpoly recovery with nested monomial predictions. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13090, pp. 392–421. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_14
12. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_15
13. Hu, K., Wang, Q., Wang, M.: Finding bit-based division property for ciphers with complex linear layers. IACR Trans. Symmetric Cryptol. **2020**(1), 396–424 (2020). https://doi.org/10.13154/tosc.v2020.i1.396-424
14. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45661-9_9
15. Lai, X.: Higher order derivatives and differential cryptanalysis. Communications and Cryptography: Two Sides of One Tapestry pp. 227–233 (1994)

16. Li, T., Sun, Y.: Superball: A new approach for MILP modelings of boolean functions. IACR Trans. Symmetric Cryptol. **2022**(3), 341–367 (2022). https://doi.org/10.46586/tosc.v2022.i3.341-367
17. Mandal, K., Gong, G., Fan, X., Aagaard, M.D.: Optimal parameters for the WG stream cipher family. Cryptogr. Commun. **6**(2), 117–135 (2014). https://doi.org/10.1007/s12095-013-0091-0
18. Nawaz, Y., Gong, G.: WG: A family of stream ciphers with designed randomness properties. Inf. Sci. **178**(7), 1903–1916 (2008). https://doi.org/10.1016/j.ins.2007.12.002
19. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 185–215. Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_7
20. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5
21. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_9
22. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.1) (2020), `https://www.sagemath.org`
23. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12
24. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_9
25. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_18
26. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 275–305. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_10
27. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 398–427. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_14
28. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_24
29. Ye, C., Tian, T.: Revisit division property based cube attacks: Key-recovery or distinguishing attacks? IACR Trans. Symmetric Cryptol. **2019**(3), 81–102 (2019). https://doi.org/10.13154/tosc.v2019.i3.81-102

30. Ye, C., Tian, T.: A practical key-recovery attack on 805-round trivium. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13090, pp. 187–213. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_7