Revisit Division Property Based Cube Attacks: Key-Recovery or Distinguishing Attacks?

Chen-Dong Ye and Tian Tian

National Digital Switching System Engineering & Technological Research Center, 62 Kexue Road, Zhengzhou, 450001, China. ye_chendong@126.com,tiantian_d@126.com

Abstract. Cube attacks are an important type of key recovery attacks against stream ciphers. In particular, it is shown to be powerful against Trivium-like ciphers. Traditional cube attacks are experimental attacks which could only exploit cubes of size less than 40. At CRYPTO 2017, division property based cube attacks were proposed by Todo et al., and an advantage of introducing the division property to cube attacks is that large cube sizes which are beyond the experimental range could be explored, and so powerful theoretical attacks were mounted to many lightweight stream ciphers.

In this paper, we revisit the division property based cube attacks. There is an important assumption, called Weak Assumption, proposed in division property based cube attacks to support the effectiveness of key recovery. Todo et al. in CRYPTO 2017 said that the Weak Assumption was expected to hold for theoretically recovered superpolies of Trivium according to some experimental results on small cubes. In this paper, based on some new techniques to remove invalid division trails, some best key recovery results given at CRYPTO 2017 and CRYPTO 2018 on Trivium are proved to be distinguishers. First, we build a relationship between the bit-based division property and the algebraic degree evaluation on a set of active variables. Second, based on our algebraic point of view, we propose a new variant of division property which incorporates the distribution of active variables. Third, a new class of invalid division trails are characterized and new techniques based on MILP models to remove them are proposed. Hopefully this paper could give some new insights on accurately evaluating the propagation of the bit-based division property and also attract some attention on the validity of division property based cube attacks against stream ciphers.

Keywords: Division property, cube attacks, MILP, Trivium

1 Introduction

The cube attack is a powerful cryptanalytic technique against stream ciphers proposed by Dinur and Shamir at Eurocrypt 2009 in [1]. Note that for a stream cipher, the first output bit could be described by a tweakable polynomial $f(\boldsymbol{x}, \boldsymbol{v})$ where $\boldsymbol{x}$ represents secrete variables and $\boldsymbol{v}$ represents public variables. Then the main idea of cube attacks is to simplify f by evaluating a set of public variables indexed by I to all their possible values and summing the resultant polynomials. The set C_I containing all the possible values of the public variables indexed by I is called a cube and the symbolic sum of $2^{|I|}$ resultant polynomials obtained from f is called the superpoly of C_I in f. In cube attacks, attackers make use of superpolies to recover secrete key information.

In [1, 2, 3, 4], low-degree superpolies are found by performing a number of linearity/quadraticity tests and the algebraic normal forms of superpolies are also experimentally recovered. Hence, cube attacks in [1, 2, 3, 4] are called experimental cube attacks. The

advantage of an experimental cube attack is that it is easy to verify the correctness of the recovered superpolies since their ANFs are clearly provided. However, testing cubes of size greater than 35 is time consuming. Hence, in experimental cube attacks, the sizes of cubes are typically confined to 40, which greatly restricts the number of attacking rounds.

In [5], by introducing the bit-based division property into cube attacks, Todo et al. could exploit large cube sizes and theoretically evaluate the security of a stream cipher against cube attacks. The division property was first proposed by Todo in [6] as a generalization of integral property used in integral cryptanalysis against block ciphers. In [6], Todo systematically studied propagation rules of division property against Feistel Network and Substitute-Permutation Network (SPN). Later, at FSE 2016, the authors of [7] further proposed the bit-based division property and applied it to SIMON family yielding several new integral distinguishers. In [8], Xiang et al. introduced mixed integer linear programming (MILP) models to evaluate the division propagation which was shown to be more efficient.

In [5], for a cube C_I, by solving an MILP model built according to the propagation rules of the division property, the authors could determine a set J, where the superpoly p_I depends on x_j's ($j \in J$) for arbitrary assignments of non-cube variables. Then, by constructing the truth table of p_I for some randomly chosen assignments of non-cube variables, they attempted to find a proper assignment such that p_I was non-constant. Once a non-constant superpoly p_I was found, a part of the key information could be recovered. Due to the power of MILP solvers, large cubes could be explored. For example, in [5], it was shown that the superpoly of a given 72-dimensional cube was dependent on at most five key variables for the 832-round Trivium. Later in [9], the authors introduced several techniques to improve the division property based cube attacks proposed in [5]. Their techniques focused on finding proper assignments of non-cube variables faster and reducing the complexity of recovering the superpoly. It was shown in [9] that the superpoly of a given 78-dimensional cube was dependent on at most one key variable for the 839-round Trivium. So far this is the best key recovery attack against Trivium regarding the number of rounds.

Besides division property based cube attacks, another two important variants of cube attacks are dynamic cube attacks [10] and correlation cube attacks [11]. Dynamic cube attacks recover secrete key information by exploiting distinguishers on superpolies such as unbalance and constantness. To obtain such distinguishers, the main idea of dynamic cube attacks is to simplify the ANF representation of some intermediate state bits by assigning dynamic constraints to public variables. Dynamic cube attacks were successfully used to break Grain-128 [10, 12]. Although in [13], the authors propose dynamic cube attacks against 721- and 855-round Trivium, quickly the attack against 721-Trivium was experimentally verified to fail and some complexity analysis also indicated that the 855-round attack was questionable in [14]. Correlation cube attacks recover secrete key information by solving a system of probabilistic equations in key variables derived from conditional correlation properties between superpolies and a set of simple key expressions which is a basis of the superpoly. In [11], a correlation cube attack was applied to 835-round Trivium which could recover about 5-bit key information with time complexity 2^{44}, using 2^{45} keystream bits and preprocessing time 2^{51}.

1.1 Motivations

First, we revisit what is guaranteed by a bit-based division trail for an r-round iterative cipher. Second, we briefly discuss how division trails are used in cube attacks against stream ciphers, and we will see the problem. Third, we discuss the countermeasure proposed by previous papers to the problem. Forth, we explain why we focus on Trivium.

Let E be a target r-round n-bit iterated cipher, whose input variables are denoted by $x_1, x_2, \ldots, x_n$ and the output variable is denoted by y, respectively, i.e., $y = E(x_1, x_2, \ldots,$

x_n). If E is a block cipher, then $x_1, x_2, \ldots, x_n$ will represent plaintext bits and y means a ciphertext bit[1]. If E is a stream cipher, then $x_1, x_2, \ldots, x_n$ will represent both IV and key variables and y means the first keystream bit. If there is a division trail $\boldsymbol{k}_0 \xrightarrow{E} k_r = 1$, then attackers do *not* know whether the output bit y is balanced or not. On the other hand, if there is no division trail such that $\boldsymbol{k}_0 \xrightarrow{E} 1$, then it is guaranteed that the output bit y is balanced. The prevail method to check the existence of a division trail like $\boldsymbol{k}_0 \xrightarrow{E} 1$ is using MILP solvers, that is, generating an MILP model to cover all division trails starting from $\boldsymbol{k}_0$, and the feasibility of the model will tell us whether there is a division trail from $\boldsymbol{k}_0$ to 1 or not.

Next let us revisit how the division property is used in a cube attack against a stream cipher $f(\boldsymbol{x}, \boldsymbol{v})$ where $\boldsymbol{x}$ and $\boldsymbol{v}$ denote the secrete key and IV variables, respectively. Given a cube C_I and a secrete key variable x_j where I is a subset of IV indices, generate an MILP model $\mathcal{M}$ that covers all division trails from $(\boldsymbol{e}_j, \boldsymbol{k}_I)$ and evaluate whether there exists a division trail such that $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$, where $\boldsymbol{k}_I$ is the division property of the cube C_I and $\boldsymbol{e}_j$ is the unit vector whose only j-th bit is 1. If there is no division trail such that $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$, then it is guaranteed that the secrete key variable x_j is *not* involved in the superpoly p_I of the cube C_I. But on the other hand, if there is a division trail such that $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$, attackers do not know whether the secrete key variable x_j is involved in p_I or not. Hence what is guaranteed by the division property in a cube attack is a set of secrete key variables not appearing in a target superpoly. However, to mount a key recovery attack, attackers need to know all key variables that are involved in a superpoly.

The countermeasure used in the previous papers on division property based cube attacks is giving the following assumption to support the existence of key variables in a superpoly, namely Weak Assumption.

Assumption 1 (Weak Assumption [5, 15, 9])**.** *For a cube C_I, there are many values in the constant part of IV whose corresponding superpoly is not a constant function.*

Based on this assumption, a division property based cube attack goes roughly like this. Also take $f(\boldsymbol{x}, \boldsymbol{v})$ and the cube C_I for example. First using the division property to rule out a set J_1 of secrete key variables not appearing in the superpoly p_I, which on the other hand implies that secrete key variables not included in J_1 possibly appear in p_I, the set of which is denoted by J_2. That is to say, it is unknown whether a secrete key variable in J_2 appears in p_I or not, but if p_I is not a constant function, then J_2 includes all its variables. Note that it is also possible that none of the key variables in J_2 appears in p_I, for which case p_I is a constant function. Second, based on Assumption 1, it is thought that attackers could easily find an appropriate constant part of IV to make p_I not a constant function. In such case, attackers recover the algebraic normal form (ANF) of p_I on the set J_2 of key variables. Finally, attackers obtain the value of p_I online to build an equation on key variables. Because cube sizes used in division property based cube attacks were very large, best results are theoretical attacks and impossible to experimentally verify. We argue that if Assumption 1 fails, some key recovery attacks claimed in [5, 15, 9] will be distinguishing attacks only. The validity of Weak Assumption was ever briefly discussed in [15, Sect. 7]. Based on some experiments on small cubes, it was concluded in [15] that Weak Assumption was expected to hold in theoretical recovered superpolies for Trivium, and if the assumption did not hold, the recovered superpoly is useful for distinguishing attacks. Although we agree that Assumption 1 should hold with a large probability, they still might fail for a few specific attacks, especially when few key variables are involved in a superpoly, say 1 or 2 key variables, which often happens on Trivium.

[1] Generally, for a block cipher, there are n-bit ciphertexts. Because we focus on stream ciphers, here we simplify the representation of an iterated block cipher to make it look unified with that of a stream cipher.

The validity of Assumption 1 as well as our experimental observation that Assumption 1 fails in some best key recovery attacks on Trivium is the motivation of our work in this paper.

Finally, there are two reasons for us focusing on Trivium in this paper. First, Trivium is an important and typical target for cube attacks. Second, compared with other NFSR-based ciphers, we feel that Weak Assumption is more likely to fail for Trivium since Trivium has quite simple state update function and the recovered superpolies often involve few key variables. So far we do not observe invalid key recovery results for other NFSR-based ciphers.

1.2 Our Contributions

The validity of Assumption 1 is related to the accuracy of division trails. To raise the accuracy, it is necessary to remove as many invalid division trails as possible. In this paper, we propose new methods to identify and remove a class of invalid division trails which could not be aborted by the previous propagation rules of the division property. Consequently, division property based attacks could definitely be improved.

The theoretical basis is our algebraic point of view of the bit-based division property. When an input set of an iterative cipher is just a cube defined by a set of active variables, it is shown that if we treat each output bit as a function on input variables, then the zero-sum property in the definition of the bit-based division property is equivalent to an algebraic degree bounding on active variables. Then we rewrite the definition of the bit-based division property by focusing on algebraic degrees. Furthermore, we propose a new variant of division property with propagation information of each active variables by describing its propagation rules for AND, XOR and COPY. Besides, to facilitate computing division trails of the new variant we also provide MILP models for describing these propagation rules. Then with a division trail of the new variant, we could trace the propagation of *every* active variable along this trail[2]. This is impossible for the conventional division trails. Based on this new variant of division property, it is easy for us to characterize a new class of invalid division trails: if the propagation of active variables given by a division trail is not true, then this trail is invalid. Finally, we show how to efficiently remove invalid division trials with MILP models.

As an application, we apply our new techniques to the round-reduced Trivium. Our main goal is to check whether the known best key-recovery attacks on Trivium are based on invalid division trails. If the existence of all key variables in a superpoly is implied by invalid division trails, then the superpoly is a constant polynomial, and so it is useless in a key recovery attack. Best key-recovery attacks against the round-reduced Trivium based on division property were given in [5, 9]. The best key-recovery attack in [5] was the one mounting to 832-round Trivium which only used one superpoly of some 72-dimensional cube and the superpoly was said to have 5 key variables, and the best key-recovery attack in [9] was the one mounting to 839-round Trivium which only used one superpoly of some 78-dimensional cube and the superpoly was said to have 1 key variable. We checked these two best ones as well as some other key-recovery attacks reported in [9] by detecting and removing invalid division trails defined in this paper. Our results are summarized in Table 1. Superpolies of all the cubes listed in Table 1 are constant functions, and so they could only provide distinguishing attacks. Besides, it is clear that better zero-sum distinguishers might be obtained by removing more invalid division trails. Therefore, we try to find better zero-sum distinguisher for full IV variables by our new method, and it is shown that the first output bit of 840-round Trivium is balanced on full IV variables which tackles one more round than previous best result in [15], see Table 2.

[2]Propagation trails of active variables are not unique. Different division tails implies different propagation trails of active variables.

Table 1: Checking key-recovery attacks on round-reduced Trivium

# of Rounds	Cube	Cube Size	Involved Key Variables	Checking Result
832	I_1	72	$k_{34}, k_{58}, k_{59}, k_{60}, k_{61}$ [5]	no key variable
833	I_2	73	$k_{49}, k_{58}, k_{60}, k_{64}, k_{74}, k_{75}, k_{76}$ [9]	no key variable
833	I_3	74	k_{60} [9]	no key variable
835	I_4	77	k_{57} [9]	no key variable
836	I_5	78	k_{57} [9]	no key variable
839	I_6	78	k_{61} [9]	no key variable

$I_1 = \{1, 2, \ldots, 65, 67, 69, \ldots, 79\}$, $I_2 = \{1, 2, \ldots, 67, 69, 71, \ldots, 79\}$
$I_3 = \{1, 2, \ldots, 69, 71, 73, \ldots, 79\}$, $I_4 = \{1, 2, 3, 4, 6, 7, \ldots, 50, 52, 53, \ldots, 64, 66, 67, \ldots, 80\}$
$I_5 = \{1, \ldots, 11, 13, \ldots, 42, 44, \ldots, 80\}$, $I_6 = \{1, \ldots, 33, 35, \ldots, 46, 48, \ldots, 80\}$

Table 2: Zero-sum distinguishers against Trivium

Applications	# of Rounds	Cube Size	Type	Reference
Trivium	793	80	zero-sum distinguisher	[16]
	837	37	zero-sum distinguisher	[16]
	839	80	zero-sum distinguisher	[15]
	840	80	zero-sum distinguisher	Section 5.6

1.3 Organization

The rest of this paper is organized as follows. Sect. 2 briefly introduces the necessary backgrounds of this paper. In Sect. 3, we view the division property from an algebraic point of view. In Sect. 4, we introduce new techniques to remove invalid division trails. In Sect. 5, we apply our attack framework to the round-reduce Trivium. Finally, Sect. 6 concludes this paper.

2 Preliminaries

2.1 Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP) is a kind of mathematical optimization whose objective function and constraints are linear, and all or some of the variables are constrained to be integers. Generally, there are variables $\mathcal{M}.var$, constraints $\mathcal{M}.con$, and the objective function $\mathcal{M}.obj$ in an MILP model $\mathcal{M}$. If there is no objective function in $\mathcal{M}$, then MILP solvers like Gurobi [17] will return whether $\mathcal{M}$ is feasible.

MILP was first applied to differential and linear cryptanalysis by N. Mouha et al. in [18]. Since then, MILP has been applied to search characteristics in many cryptanalysis techniques such as differential cryptanalysis [19, 20], impossible differential cryptanalysis [21] and integral cryptanalysis based on the division property [8].

2.2 Cube Attacks

The idea of cube attacks was first proposed by Dinur and Shamir in [1]. In a cube attack against stream ciphers, an output bit z is described as a tweakable Boolean function f in key variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and public IV variables $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$, i.e., $z = f(\boldsymbol{x}, \boldsymbol{v})$. Let $I = \{i_1, i_2, \ldots, i_d\}$ be a subset of IV indices. Then f can be rewritten as

$$f(\boldsymbol{x}, \boldsymbol{v}) = t_I \cdot p_I(\boldsymbol{x}, \boldsymbol{v}) \oplus q(\boldsymbol{x}, \boldsymbol{v}), \tag{1}$$

where $t_I = \prod_{i \in I} v_i$, p_I does not contain any variable in $\{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$, and each term in q is not divisible by t_I. It can be seen that the summation of 2^d functions derived from f by assigning all the possible values to d variables indexed by I equals to p_I, that is,

$$\bigoplus_{(v_{i_1}, v_{i_2}, \ldots, v_{i_d}) \in \mathbb{F}_2^d} f(\boldsymbol{x}, \boldsymbol{v}) = p_I(\boldsymbol{x}, \boldsymbol{v}). \tag{2}$$

The public variables in $\{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$ are called *cube variables*, while the remaining public variables are called non-cube variables. The set C_I of all 2^d possible assignments of the cube variables is called a *d-dimensional cube*, and the polynomial p_I is called the *superpoly* of C_I in f.

A cube attack consists of the precessing phase and the online phase. In the preprocessing phase, attackers try to find cubes with low-degree superpolies. In the online phase, the previously found superpolies are evaluated under the real key. By solving a system of low-degree equations, some key variables could be recovered.

2.3 The Bit-Based Division Property

The conventional bit-based division property was introduced in [7]. The authors of [7] also introduced the bit-based division property using three subsets. In this paper, we 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 $\mathbb{F}_2^n$. Let $\mathbb{K}$ be a set whose elements take an n-dimensional bit vector. When the multiset $\mathbb{X}$ has the division property $D_{\mathbb{K}}^{1^n}$, it fulfills the following conditions:

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

where $\boldsymbol{u} \succeq \boldsymbol{k}$ if and only if $u_i \geq k_i$ for all i and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=1}^{n} x_i^{u_i}$.

Let E_r be an r-round iterative cipher of size n, which is initialized with $x_1, x_2, \ldots, x_n$. Assume that $\mathbb{X}$ is the input set with the division property $D_{\mathbb{K}_0}^{1^n}$. Denote by $\mathbb{Y}$ the corresponding output set created from $\mathbb{X}$ by E_r. Generally, it is difficult to evaluate the division property of $\mathbb{Y}$ directly. Based on the propagation rules of basic operations proved in [7, 8], the division property of $\mathbb{Y}$, denoted by $D_{\mathbb{K}_r}^{1^n}$, can be figured out by evaluating the propagation of the division property for every round function. More specifically, when the input set $\mathbb{X}$ is generated by a set of active variables indexed by $I = \{i_1, i_2, \ldots, i_d\}$, where the active variables traverse all $2^{|I|}$ possible combinations while the other variables are assigned to constants. Then, $\mathbb{X}$ has the division property $D_{\boldsymbol{k}}^{1^n}$, where $k_i = 1$ if i in I and $k_i = 0$ otherwise. In this case, the division property of $\mathbb{Y}$ can be evaluated as $\{\boldsymbol{k}\} = \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \cdots \to \mathbb{K}_r$, where $D_{\mathbb{K}_i}^{1^n}$ is the division property of the internal state after i rounds. Moreover, if there does not exist a vector $\boldsymbol{e}_j$ (only the j-th element is 1) in $D_{\mathbb{K}_r}^{1^n}$, then the j-th output bit is balanced.

However, as r increases, $|\mathbb{K}_r|$ would expand rapidly and lead to a high memory complexity [9]. It confines the bit-based division property to be applied to small block ciphers such as SIMON32 and Simeck32 [7]. To avoid the high memory complexity, in [8], the authors applied the MILP methods to the bit-based division property. They first introduced the concept of division trails, which is defined as follows.

Definition 2 (**Division Trail** [8])**.** Let us consider the propagation of the division property $\{\boldsymbol{k}\} = \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \cdots \to \mathbb{K}_r$. Moreover, for any vector $\boldsymbol{k}_{i+1}^* \in \mathbb{K}_{i+1}$, there must exist a vector $\boldsymbol{k}_i^* \in \mathbb{K}_i$ such that $\boldsymbol{k}_i^*$ can propagate to $\boldsymbol{k}_{i+1}^*$ by the propagation rules of division property. 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 $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for $i \in \{0, 1, \ldots, r-1\}$, we call $\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r$ an r-round division trail.

In [8], the authors described the propagation rules for AND, COPY and XOR with MILP models, see [8] for the detailed definition of AND, COPY and XOR. Therefore, they could build an MILP model to cover all the possible division trails generated during the propagation. Besides, in [5, 22], the authors made some simplifications to those MILP models in [8]. In this paper, we describe the propagation rules of division property for AND, COPY and XOR in the following three propositions.

Proposition 1 (**MILP Model for AND**). *Let* $(a_1, a_2, \ldots, a_m) \xrightarrow{\text{AND}} b$ *be a division trail of* AND. *The following inequalities are sufficient to describe the propagation of the division property for* AND.

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \ldots, a_m, b \text{ as binary}, \\ \mathcal{M}.con \leftarrow b = \max(a_1, a_2, \ldots, a_m). \end{cases}$$

Proposition 2 (**MILP Model for XOR**). *Let* $(a_1, a_2, \ldots, a_m) \xrightarrow{\text{XOR}} b$ *be a division trail of* XOR. *The following inequalities are sufficient to describe the propagation of the division property for* XOR.

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \ldots, a_m, b \text{ as binary}, \\ \mathcal{M}.con \leftarrow b = a_1 + a_2 + \cdots + a_m. \end{cases}$$

Proposition 3 (**MILP Model for COPY**). *Let* $a \xrightarrow{\text{COPY}} (b_1, b_2, \ldots, b_m)$ *be a division trail of* COPY. *The following inequalities are sufficient to describe the propagation of the division property for* COPY.

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_1, b_2, \ldots, b_m \text{ as binary}, \\ \mathcal{M}.con \leftarrow a = b_1 + b_2 + \cdots + b_m. \end{cases}$$

2.4 Cube Attacks Combining with the Bit-based Division Property

In [5], the authors applied the bit-based division property to cube attacks. Instead of using the division property to find zero-sum integral distinguishers, in [5], they used the division property to analyze the ANF coefficients of a Boolean function f. Based on the following lemma and proposition, they proposed the division property based cube attacks.

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

Proposition 4. *Let* $f(\boldsymbol{x}, \boldsymbol{v})$ *be a polynomial, where* $\boldsymbol{x}$ *and* $\boldsymbol{v}$ *denote the secret and public variables, respectively. For a set of indices* $I = \{i_1, i_2, \ldots, i_d\} \subset \{1, 2, \ldots, m\}$, *let* C_I *be a set where* $\{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$ *traverse all* $2^{|I|}$ *values and the other public variables are set to constants. Let* $\boldsymbol{k}_I$ *be an* m*-dimensional bit vector such that* $\boldsymbol{v}^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \cdots v_{i_d}$, *i.e.,* $k_i = 1$ *if* $i \in I$ *and* $k_i = 0$ *otherwise. If there is no division trail such that* $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$, *then* x_j *is not involved in the superpoly of the cube* C_I.

When f represents the output bit of the target cipher, based on Proposition 4, for a cube C_I, a set J including all the key variables involved in the superpoly of C_I could be identified by MILP methods. More specifically, if there exists a division trail such that $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{f} 1$, then it is regarded that x_j is in J. After knowing the set J, in [5], division property based cube attacks are described in the following three steps.

1. (Offline phase.) Find a preferable superpoly. Set the non-cube variables to constants randomly. For each possible value of the key variables indexed by J, query the oracle and obtain the summation of all the $2^{|I|}$ output values. Thus, the truth table of $p_I(\boldsymbol{x}, \boldsymbol{v})$ can be constructed with time complexity $2^{|I|+|J|}$. Search for non-constant $p_I(\boldsymbol{x}, \boldsymbol{v})$ by changing the values of non-cube variables.

2. (Online phase.) Set the non-cube variables to previous found values which make the corresponding $p_I(\boldsymbol{x}, \boldsymbol{v})$ non-constant. Query the encryption oracle and get one bit $p_I(\boldsymbol{x}, \boldsymbol{v})$, denoted by a. Then, we obtain an equation $p_I(\boldsymbol{x}, \boldsymbol{v}) = a$ about secrete key variables. The values of key variables which do not satisfy $p_I(\boldsymbol{x}, \boldsymbol{v}) = a$ could be discarded.

3. (Brute-force search phase.) Guess the remaining secrete key variables to recover the entire key.

After the division property based cube attacks were first presented in [5], some improvements were given in [15, 9]. To name a few, in [15], which is the full version of [5], non-cube IV variables could be filled up with 0, later in [9] non-cube IV variables could be filled up with either 0 or 1, and also in [9] with degree bounding and term enumeration techniques, the time complexity to recovery the superpoly could be reduced from $2^{|I|+|J|}$ to $2^{|I|} \times \binom{|J|}{\leq d}$ where d is the degree upper bound of the superpoly.

3 An algebraic point of view of the division property and division trails

Recall that the conventional bit-based division property is defined based on the bit product of an input set $\mathbb{X}$, see Definition 1, which tries to characterize a set of bit products that are balanced on the input set $\mathbb{X}$. To be accordance with the definition, when analyzing an iterative cipher, we need to transform an input set $\mathbb{X}$ into an output set $\mathbb{Y}$ and describe the division property of the output set $\mathbb{Y}$ to find balanced output bits. With the view that each output bit is a function on input variables, we will show that the division property used in cube attacks against stream ciphers is equivalent to algebraic degree bounding in this section. Besides, based on the algebraic point of view, we propose a new variant of division property by which we could trace the propagation of active variables. This new variant of division property will be used to remove invalid division trails.

3.1 Notations

The symbols "+" and "$\oplus$" denote the integer addition and exclusive or, respectively.

The finite field of two elements is denoted by $\mathbb{F}_2$ and for a positive integer n, the n-dimensional vector space over $\mathbb{F}_2$ is denoted by $\mathbb{F}_2^n$. Define a partial order $\preceq$ on $\mathbb{F}_2^n$ by

$$\boldsymbol{a} \preceq \boldsymbol{b} \Leftrightarrow a_i \leq b_i \text{ for all } 1 \leq i \leq n,$$

where $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$. Then for a subset M of $\mathbb{F}_2^n$, we set

- $M^+ = \{\boldsymbol{u} \mid \text{there exists some } \boldsymbol{v} \in M \text{ with } \boldsymbol{v} \preceq \boldsymbol{u}\}$,
- $\overline{M} = \{\boldsymbol{a} \in \mathbb{F}_2^n | \boldsymbol{a} \notin M\}$, i.e., the complementary set of M in $\mathbb{F}_2^n$.
- $M^- = \overline{M^+}$, i.e., the complementary set of M^+ in $\mathbb{F}_2^n$.

Let n be a positive integer. An n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$ is a mapping from $\mathbb{F}_2^n$ into $\mathbb{F}_2$. It is known that a Boolean function $f(x_1, x_2, \ldots, x_n)$ can be uniquely represented as a multivariate polynomial of the form:

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\ldots,\alpha_n)\in\{0,1\}^n} u_{f,\boldsymbol{\alpha}} \cdot \left(\prod_{j=1}^{n} x_j^{\alpha_j} \right),$$

where $u_{f,\boldsymbol{\alpha}} \in \mathbb{F}_2$, which is called the *algebraic normal form* (ANF) of f. The *algebraic degree* of f, denoted by $\deg(f)$, is the global degree of the ANF of f, i.e.,

$$\deg(f) = \max\{\alpha_1 + \alpha_2 + \cdots + \alpha_n \mid u_{f,\boldsymbol{\alpha}} \neq 0\}.$$

Let $I \subseteq \{1, 2, \ldots, n\}$ be a subset of variable indexes. Then the algebraic degree of f restricted on the variables indexed by I is denoted by $\deg_I(f)$, i.e.,

$$\deg_I(f) = \max\{\sum_{i\in I} \alpha_i \mid u_{f,\boldsymbol{\alpha}} \neq 0\}$$

where $\sum$ denotes the integer addition. Take $f = x_1x_2x_3x_5 \oplus x_2x_5 \oplus x_3x_4 \oplus x_4$ and $I = \{2, 3, 4\}$ for an example. We have $\deg(f) = 4$ and $\deg_I(f) = 2$.

A vectorial Boolean function f is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$, denoted by $Y = f(X)$ where $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1(X), y_2(X), \ldots, y_n(X))$. It is clear that each $y_i(X)$ is a Boolean function on X for $1 \leq i \leq n$. In the following we denote the output of r-round iterative action of f on X by $Y_r = (y_{r,1}(X), y_{r,2}(X), \ldots, y_{r,n}(X))$, i.e. $Y_r = f^r(X)$.

The abbreviation **w.r.t.** for "with respect to" will be used in the following.

3.2 Relationship between the bit-based division property and algebraic degree bounding

Let $f : (x_1, x_2, \ldots, x_n) \to (y_1, y_2, \ldots, y_n)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. For an input value $\boldsymbol{a}$, an r-round iteration action of f on $\boldsymbol{a}$ means $\boldsymbol{b} = f^r(\boldsymbol{a})$ where $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{F}_2^n$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{F}_2^n$. For a stream cipher, f means its internal update function. Let us fix a positive integer r. Choose some active input variables which are indexed by $I \subseteq \{1, 2, \ldots, n\}$ with $I \neq \emptyset$. Prepare an input set $\mathbb{X}_I$ where all variables indexed by I take all possible combinations of 0/1 and each variable not indexed by I takes a constant. Then the division property of $\mathbb{X}_I$ is given by $\boldsymbol{k}_I \in \mathbb{F}_2^n$ with $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Assume the propagation of the division property starting from $\boldsymbol{k}_I$ through r rounds is given by $\{\boldsymbol{k}_I\} = \mathbb{K}_0 \xrightarrow{f} \mathbb{K}_1 \xrightarrow{f} \cdots \xrightarrow{f} \mathbb{K}_r$. Then it follows that

$$\bigoplus_{\boldsymbol{b} \in \mathbb{Y}} \left(\prod_{i=1}^{n} b_i^{u_i} \right) = 0 \text{ for } \boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{K}_r^-, \tag{3}$$

where $\mathbb{Y}$ is the output set of the input set $\mathbb{X}_I$ and $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$. Since each output bit is a function on input variables, it is clear that (3) is equivalent to

$$\bigoplus_{\boldsymbol{a} \in \mathbb{X}_I} \left(\prod_{i=1}^{n} y_{r,i}(\boldsymbol{a})^{u_i} \right) = 0 \text{ for } \boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{K}_r^-. \tag{4}$$

Since the propagation rules of the division property, i.e., Propositions 1-3, only distinguish active and non-active variables, which are independent of the specific constants taken by non-active variables, it follows that (4) holds for all possible values of non-active variables. Let us recall the following property of Boolean functions whose proof is omitted here.

Lemma 2. *Let $g(x_1, x_2, \ldots, x_n)$ be an n-variable Boolean function. Let $I \subseteq \{1, 2, \ldots, n\}$ be an nonempty set of active variable indexes. Let $(\mathbb{X}_I, C)$ be an input set such that all variables indexed by I take all possible combinations of 0/1 and the other variables take constants labeled by C where $C \in \mathbb{F}_2^{n-|I|}$. Then $\sum_{\boldsymbol{x} \in (\mathbb{X}_I, C)} g(\boldsymbol{x}) = 0$ holds for all $C \in \mathbb{F}_2^{n-|I|}$ if and only if $\deg_I(g) < |I|$.*

It immediately follows from Lemma 2 that the division property (4) implies that

$$\deg_I \left(\prod_{i=1}^{n} y_{r,i}(X)^{u_i} \right) < |I| \text{ for } \boldsymbol{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{K}_r^-. \tag{5}$$

Therefore, if we only focus on the case that **an input set is a cube defined by a set of active variables**, which is exactly the case used in division property based cube attacks against stream ciphers [5], then the definition of the bit-based division property is equivalent to the following one.

Definition 3. Let $Y = f(X)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and I be a non-empty subset of $\{1, 2, \ldots, n\}$ whose elements are indexes of active variables. Let $\mathbb{K}$ be a subset of $\mathbb{F}_2^n$. If

$$\deg_I \left(\prod_{i=1}^{m} y_i(X)^{u_i} \right) < |I| \text{ for } \boldsymbol{u} = (u_1, u_2, \ldots, u_m) \in \mathbb{K}^-,$$

then we say f has the division property $\mathbb{K}$ w.r.t. I or $\mathbb{K}$ is a division property set of f w.r.t. I.

3.3 Extending division property and trails with active variable distribution

From the algebraic point of view, the division trail could be considered as a propagation trail of active variables. This could be seen more clearly after we extend the conventional division property and trails to include indications of the distribution of active variables.

Definition 4. Let $Y = f(X)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and I be a non-empty subset of $\{1, 2, \ldots, n\}$ whose elements are indexes of active variables. Let

$$\mathbb{M} = \{([k_{j_1}, \lambda_{j_1}], [k_{j_2}, \lambda_{j_2}], \ldots, [k_{j_n}, \lambda_{j_n}]) \mid [k_{j_i}, \lambda_{j_i}] \in \mathbb{F}_2 \times T_I, \prod_{i=1}^{n} \lambda_{j_i}^{k_{j_i}} = \prod_{i \in I} x_i, j \in J\}$$

be a set of n-tuples indexed by a set J where $T_I = \{\prod_{i \in I} x_i^{e_i} | e_i \in \{0, 1\}\}$. Let $\eta : (\mathbb{F}_2 \times T_I)^n \to \mathbb{F}_2^n$ be a mapping[3] given by

$$\eta : ([k_{j_1}, \lambda_{j_1}], [k_{j_2}, \lambda_{j_2}], \ldots, [k_{j_n}, \lambda_{j_n}]) \mapsto (k_{j_1}, k_{j_2}, \ldots, k_{j_n}).$$

If

$$\deg_I \left(\prod_{i=1}^{n} y_i(X)^{u_i} \right) < |I| \text{ for } \boldsymbol{u} \in \mathbb{K}^-,$$

where $\mathbb{K} = \eta(\mathbb{M})$, then we say $\mathbb{M}$ is a division property set of f in $(\mathbb{F}_2 \times T_I)^n$ w.r.t. I.

Remark 1. Without loss of generality, if $k_{j_i} = 0$, then we always set $\lambda_{j_i} = \prod_{i \in I} x_i^0$.

Remark 2. In the following, a division property set $\mathbb{K} \subseteq \mathbb{F}_2^n$ satisfying Definition 3 is called a division property set in $\mathbb{F}_2^n$.

It can be seen from Definition 4 that if $\mathbb{M}$ is a division property set of f in $(\mathbb{F}_2 \times T_I)^n$ w.r.t. I, then $\eta(\mathbb{M})$ is a division property set of f in $\mathbb{F}_2^n$ w.r.t. I. On the contrary, given a division property set $\mathbb{K}$ in $\mathbb{F}_2^n$, there is a corresponding division property set $\mathbb{M}$ in $(\mathbb{F}_2 \times T_I)^n$ such that $\eta(\mathbb{M}) = \mathbb{K}$. We prove this in the following lemma.

Lemma 3. *Let $Y = f(X)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and I be a non-empty subset of $\{1, 2, \ldots, n\}$ whose elements are indexes of active variables. If f has the division property $\mathbb{K}$ in $\mathbb{F}_2^n$ w.r.t. I, then there is a set $\mathbb{M} \subseteq (\mathbb{F}_2 \times T_I)^n$ such that $\eta(\mathbb{M}) = \mathbb{K}$ and f has the division property $\mathbb{M}$ in $(\mathbb{F}_2 \times T_I)^n$ w.r.t. I.*

Proof. Let $\boldsymbol{k} = (k_1, k_2, \ldots, k_n) \in \mathbb{K}$. Then we have $\deg_I(\prod_{i=1}^{n} y_i(X)^{k_1}) \leq |I|$.

If $\deg_I(\prod_{i=1}^{n} y_i(X)^{k_i}) < |I|$, then randomly choose $(\lambda_1, \lambda_2, \ldots, \lambda_n) \in T_I^n$ such that $\prod_{i=1}^{n} \lambda_{j_i}^{k_{j_i}} = \prod_{i \in I} x_i$. Then set $\Gamma_{\boldsymbol{k}} = ([k_1, \lambda_1], [k_2, \lambda_2], \ldots, [k_n, \lambda_n])$.

If $\deg_I(\prod_{i=1}^{n} y_i(X)^{k_1}) = |I|$, then $\prod_{i \in I} x_i$ appears in $\prod_{i=1}^{n} y_i(X)^{k_1}$. It follows that there is at least a decomposition of $\prod_{i \in I} x_i$ given by $\prod_{i \in I} x_i = \prod_{i=1}^{n} \lambda_i^{k_i}$ such that λ_i appears in the ANF of $y_i(X)$ if $k_i = 1$. Hence we set $\Gamma_{\boldsymbol{k}} = ([k_1, \lambda_1], [k_2, \lambda_2], \ldots, [k_n, \lambda_n])$.

Finally it can be seen that the set $\mathbb{M} = \{\Gamma_{\boldsymbol{k}} \mid \boldsymbol{k} \in \mathbb{K}\}$ is the desirable division property set in $(\mathbb{F}_2 \times T_I)^n$ w.r.t. I. □

[3] $(\mathbb{F}_2 \times T_I)^n = \{([k_{j_1}, \lambda_{j_1}], [k_{j_2}, \lambda_{j_2}], \ldots, [k_{j_n}, \lambda_{j_n}]) \mid k_{j_i} \in \mathbb{F}_2, \lambda_{j_i} \in T_I\}$.

It appears that a division property set in $(\mathbb{F}_2 \times T_I)^n$ may include many redundant or nonsense elements. However, by requiring the following propagation rules, only useful elements will be left when analyzing an iterative cipher.

Let $X = (x_1, x_2, \ldots, x_n)$ be a set of input variables and $I \subseteq \{1, 2, \ldots, n\}$ be a non-empty set of active variable indexes. In the following propagation rules, $a(X), b(X), a_i(X)$, $b_i(X)$ are Boolean functions on X.

Propagation rule for COPY. Let $a(X) \xrightarrow{COPY} (b_1(X), b_2(X), \ldots, b_m(X))$, where the division property of $a(X)$ in $\mathbb{F}_2 \times T_I$ w.r.t. I is $[k_a, \lambda_a]$ and the division property of $(b_1(X), b_2(X), \ldots, b_m(X))$ in $(\mathbb{F}_2 \times T_I)^m$ w.r.t. I is $([k_{b_1}, \lambda_{b_1}], [k_{b_2}, \lambda_{b_2}], \ldots, [k_{b_m}, \lambda_{b_m}])$. Then

$$k_{b_1} + k_{b_2} + \cdots + k_{b_m} = k_a, \lambda_{b_i} = \lambda_a^{k_{b_i}}, 1 \leq i \leq n.$$

Propagation rule for XOR. Let $(a_1(X), a_2(X), \ldots, a_m(X)) \xrightarrow{XOR} b(X)$, where the division property of $(a_1(X), a_2(X), \ldots, a_m(X))$ in $(\mathbb{F}_2 \times T_I)^m$ w.r.t. I is $([k_{a_1}, \lambda_{a_1}]$, $[k_{a_2}, \lambda_{a_2}], \ldots, [k_{a_m}, \lambda_{a_m}])$ and the division property of $b(X)$ in $\mathbb{F}_2 \times T_I$ w.r.t. I is $[k_b, \lambda_b]$. Then

$$k_b = k_1 + k_2 + \cdots + k_m, \lambda_b = \left(\prod_{i=1}^{m} \lambda_{a_i}\right)^{k_b}.$$

Propagation rule for AND. Let $(a_1(X), a_2(X), \ldots, a_m(X)) \xrightarrow{AND} b(X)$, where the division property of $(a_1(X), a_2(X), \ldots, a_m(X))$ in $(\mathbb{F}_2 \times T_I)^m$ w.r.t. I is $([k_{a_1}, \lambda_{a_1}]$, $[k_{a_2}, \lambda_{a_2}], \ldots, [k_{a_m}, \lambda_{a_m}])$ and the division property of b in $\mathbb{F}_2 \times T_I$ w.r.t. I is $[k_b, \lambda_b]$. Then

$$k_b = \max(k_{a_1}, k_{a_2}, \ldots, k_{a_m}), \lambda_b = \prod_{i=1}^{m} \lambda_{a_i}.$$

Based on the above propagation rules, we define division trails for an iterative function.

Definition 5. Let $Y = f(X)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and $I \subseteq \{1, 2, \ldots, n\}$ be a set of active variable indexes. Set $(\boldsymbol{k}_0, \boldsymbol{\lambda}_0) = ([k_1, \lambda_1], [k_2, \lambda_2], \ldots, [k_m, \lambda_m])$ with $[k_i, \lambda_i] = [1, x_i]$ if $i \in I$ and $[k_i, \lambda_i] = [0, \prod_{i \in I} x_i^0]$ otherwise. If

$$(\boldsymbol{k}_0, \boldsymbol{\lambda}_0) \xrightarrow{f} (\boldsymbol{k}_1, \boldsymbol{\lambda}_1) \xrightarrow{f} \cdots \xrightarrow{f} (\boldsymbol{k}_r, \boldsymbol{\lambda}_r) \tag{6}$$

satisfies that k_i propagates to k_{i+1} by the three propagation rules defined above, then it is called a division trail over $(\mathbb{F}_2 \times T_I)^n$ of f w.r.t. I.

Remark 3. The starting division property $(\boldsymbol{k}_0, \boldsymbol{\lambda}_0)$ could be seen as the division property of $f(X)^0 = \mathrm{id}$ where id denotes the identity function.

It can be seen that $\boldsymbol{\lambda}_0 \xrightarrow{f} \boldsymbol{\lambda}_1 \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{\lambda}_r$ describes the propagation of active variables along the specific division trial $\boldsymbol{k}_0 \xrightarrow{f} \boldsymbol{k}_1 \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{k}_r$. Then (6) is an indication that $\deg_I(\prod_{i=1}^{n} y_{r,i}(X)^{k_{r,i}})$ may be equal to $|I|$. But it can not immediately deduce from (6) that $\deg_I(\prod_{i=1}^{n} y_{r,i}(X)^{k_{r,i}})$ is equal to $|I|$. This is because the three propagation rules do not cover all algebraic properties of Boolean functions. For example, x_1x_2 disappears in $f \oplus g$ with $f = x_1x_2 \oplus x_2$ and $g = x_1x_2 \oplus x_2$.

4 New Techniques to Remove Invalid Division Trails

In this section, we propose a class of invalid division trails and show how to remove them with MILP models.

Algorithm 1 Checking whether a Division Trail is Invalid

Require: a function f, the set I of active variables indies, a division trail $\boldsymbol{k}_0 \xrightarrow{f} \boldsymbol{k}_1 \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{k}_r$, and an integer N_0;

Compute $(\boldsymbol{k}_0, \boldsymbol{\lambda}_0) \xrightarrow{f} (\boldsymbol{k}_1, \boldsymbol{\lambda}_1) \xrightarrow{f} \cdots \xrightarrow{f} (\boldsymbol{k}_r, \boldsymbol{\lambda}_r)$;
Compute the ANF of $s_i^{(t)}$ for $1 \le i \le n$ and $0 \le t \le N_0$;
Set $end = \min(N_0, r)$;
Check whether $\lambda_i^{(t)}$ appears in $s_i^{(t)}$ for $0 \le t \le end$ and $1 \le i \le n$;
if there exist t and j such that $k_j^{(t)} = 1$ and $\lambda_j^{(t)}$ does not appear in $s_j^{(t)}$ **then**
    return This is an invalid division trail;
else
    return This is a valid division trail;
end if

4.1 A class of invalid division trails

Based on the new variant of the division property introduced in the last section, we propose a new class of invalid division trails.

Definition 6 (Invalid Division Trail). Let $Y = f(X)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and $I \subseteq \{1, 2, \ldots, n\}$ be a set of active variable indexes. Set

$$(\boldsymbol{k}_0, \boldsymbol{\lambda}_0) = ([k_1^{(0)}, \lambda_1^{(0)}], [k_2^{(0)}, \lambda_2^{(0)}], \ldots, [k_n^{(0)}, \lambda_n^{(0)}])$$

with $[k_i^{(0)}, \lambda_i^{(0)}] = [1, x_i]$ if $i \in I$ and $[k_i^{(0)}, \lambda_i^{(0)}] = [0, \prod_{i \in I} x_i^0]$ otherwise. Let

$$(\boldsymbol{k}_0, \boldsymbol{\lambda}_0) \xrightarrow{f} (\boldsymbol{k}_1, \boldsymbol{\lambda}_1) \xrightarrow{f} \cdots \xrightarrow{f} (\boldsymbol{k}_r, \boldsymbol{\lambda}_r) \tag{7}$$

be a division trail over $(\mathbb{F}_2 \times T_I)^n$ of f w.r.t. I. If there exist some t and j such that $\boldsymbol{k}_j^{(t)} = 1$ but $\lambda_j^{(t)}$ does not appear in the ANF of $y_j^{(t)}$, then we call (7) an invalid division trail due to the absence of $\lambda_j^{(t)}$.

Remark 4. If (7) is an invalid division trail due to the absence of $\lambda_j^{(t)}$, then it is known that (7) is a trail indicating that $\deg_I(\prod_{i=1}^n (y_i^{(j)}(X))^{k_i^{(j)}}) < |I|$ for all $t \le j \le r$. Therefore, it should be aborted [4]. We provide a small example to illustrate an invalid division trial more clearly in Appendix, see Example 2.

Next, we propose Algorithm 1 to check whether a division trail is invalid or not. Given a division trail $\boldsymbol{k}_0 \xrightarrow{f} \boldsymbol{k}_1 \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{k}_r$, in Algorithm 1, we first compute the corresponding trail $\boldsymbol{\lambda}_0 \xrightarrow{f} \boldsymbol{\lambda}_1 \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{\lambda}_r$. [5] Then, we check whether there exist some t and i such that $\boldsymbol{k}_i^{(t)} = 1$ but $\boldsymbol{\lambda}_i^{(t)}$ does not appear in $s_i^{(t)}$. To achieve this goal, in Algorithm 1, we compute the ANFs of $s_i^{(t)}$ for $0 \le t \le N_0$ and $1 \le i \le n$.

4.2 Removing Invalid Division Trails

The idea of removing invalid division trails due to the absence of $\lambda_j^{(t)}$ is very straightforward. Let $\lambda_j^{(t)} = T$. We first build an MILP model which covers all the new division trails, and then add a constraint that $\lambda_j^{(t)} \neq T$.

[4] It is guaranteed that $\deg_I(\prod_{i=1}^n (y_i^{(r)}(X))^{k_i^{(r)}}) < |I|$ if and only if all trails are invalid or there is no trail. Otherwise, it is unknown whether $\deg_I(\prod_{i=1}^n (y_i^{(r)}(X))^{k_i^{(r)}}) < |I|$ or not.

[5] For a given division trail $\boldsymbol{k}_0 \xrightarrow{f} \boldsymbol{k}_1 \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{k}_r$, since $\boldsymbol{\lambda}_0$ can be set uniquely according to I, we could recover the $\boldsymbol{\lambda}_i$ according to how $\boldsymbol{k}_{i-1}$ propagates to $\boldsymbol{k}_i$ for $1 \le i \le r$.

4.2.1 MILP models for the propagation rules of the division property in $(\mathbb{F}_2 \times T_I)^n$

To give MILP models that describe the propagation rules of the division property in $(\mathbb{F}_2 \times T_I)^n$, the key point is how to model λ_{j_i} in Definition 4 which is a product of active variables. Note that a product $\prod_{i\in I} x_i^{e_i}$ of active variables could be represented by a binary vector $(e_1, e_2, \ldots, e_{|I|})$. Therefore, in our MILP model, λ_{j_i} is represented by a binary vector of length $|I|$.

Let $X = (x_1, x_2, \ldots, x_n)$ be a set of input variables and $I \subseteq \{1, 2, \ldots, n\}$ be a non-empty set of active variable indexes. Without loss of generality, we assume $I = \{1, 2, \ldots, q\}$. In the following propositions, $a(X), b(X), a_i(X), b_i(X)$ are Boolean functions on X.

Proposition 5 (**MILP Model for COPY**). *Let* $a(X) \xrightarrow{\text{COPY}} (b_1(X), b_2(X), \ldots, b_m(X))$, *where the division property of* $a(X)$ *in* $\mathbb{F}_2 \times T_I$ *w.r.t.* I *is* $[k_a, \lambda_a]$ *with* $\lambda_a = \prod_{j=1}^{q} x_j^{\alpha[j]}$ *and the division property of* $(b_1(X), b_2(X), \ldots, b_m(X))$ *in* $(\mathbb{F}_2 \times T_I)^m$ *w.r.t.* I *is* $([k_{b_1}, \lambda_{b_1}], [k_{b_2}, \lambda_{b_2}], \ldots, [k_{b_m}, \lambda_{b_m}])$ *with* $\lambda_{b_i} = \prod_{j=1}^{q} x_j^{\beta_i[j]}$ *for* $1 \le i \le m$. *The following inequalities*

$$\begin{cases} \mathcal{M}.var \leftarrow k_a, k_{b_1}, k_{b_2}, \ldots, k_{b_m} \text{ as binary} \\ \mathcal{M}.var \leftarrow \alpha[1], \alpha[2], \ldots, \alpha[q] \text{ as binary} \\ \mathcal{M}.var \leftarrow \beta_i[1], \beta_i[2], \ldots, \beta_i[q] \text{ for } i \in \{1, 2, \ldots, m\} \text{ as binary} \\ \mathcal{M}.con \leftarrow k_a = k_{b_1} + k_{b_2} + \cdots + k_{b_m} \\ \mathcal{M}.con \leftarrow \beta_i[j] = \min(\alpha[j], k_{b_i}) \text{ for } 1 \le i \le m \text{ and } 1 \le j \le q \end{cases}$$

are sufficient to describe the propagation of the division property for COPY.

Proposition 6 (**MILP Model for XOR**). *Let* $(a_1(X), a_2(X), \ldots, a_m(X)) \xrightarrow{\text{XOR}} b(X)$, *where the division property of* $(a_1(X), a_2(X), \ldots, a_m(X))$ *in* $(\mathbb{F}_2 \times T_I)^m$ *w.r.t.* I *is* $([k_{a_1}, \lambda_{a_1}], [k_{a_2}, \lambda_{a_2}], \ldots, [k_{a_m}, \lambda_{a_m}])$ *with* $\lambda_{a_i} = \prod_{j=1}^{q} x_j^{\alpha_i[j]}$ *for* $1 \le i \le m$ *and the division property of* $b(X)$ *in* $\mathbb{F}_2 \times T_I$ *w.r.t.* I *is* $[k_b, \lambda_b]$ *with* $\lambda_b = \prod_{j=1}^{q} x_j^{\beta[j]}$. *The following inequalities*

$$\begin{cases} \mathcal{M}.var \leftarrow k_{a_1}, k_{a_2}, \ldots, k_{a_m}, k_b \text{ as binary} \\ \mathcal{M}.var \leftarrow \alpha_i[1], \alpha_i[2], \ldots, \alpha_i[q] \text{ for } i \in \{1, 2, \ldots, m\} \text{ as binary} \\ \mathcal{M}.var \leftarrow \beta[1], \beta[2], \ldots, \beta[q] \text{ as binary} \\ \mathcal{M}.con \leftarrow k_b = k_{a_1} + k_{a_2} + \cdots + k_{a_m} \\ \mathcal{M}.con \leftarrow \beta[j] = \alpha_1[j] + \alpha_2[j] + \cdots + \alpha_m[j] \text{ for } 1 \le j \le q \end{cases}$$

are sufficient to describe the propagation of the division property for XOR.

Proposition 7 (**MILP Model for AND**). *Let* $(a_1(X), a_2(X), \ldots, a_m(X)) \xrightarrow{\text{AND}} b(X)$, *where the division property of* $(a_1(X), a_2(X), \ldots, a_m(X))$ *in* $(\mathbb{F}_2 \times T_I)^m$ *w.r.t.* I *is* $([k_{a_1}, \lambda_{a_1}], [k_{a_2}, \lambda_{a_2}], \ldots, [k_{a_m}, \lambda_{a_m}])$ *with* $\lambda_{a_i} = \prod_{j=1}^{q} x_j^{\alpha_i[j]}$ *for* $1 \le i \le m$ *and the division property of* $b(X)$ *in* $\mathbb{F}_2 \times T_I$ *w.r.t.* I *is* $[k_b, \lambda_b]$ *with* $\lambda_b = \prod_{j=1}^{q} x_j^{\beta[j]}$. *The following inequalities*

$$\begin{cases} \mathcal{M}.var \leftarrow k_{a_1}, k_{a_2}, \ldots, k_{a_m}, k_b \text{ as binary} \\ \mathcal{M}.var \leftarrow \alpha_i[1], \alpha_i[2], \ldots, \alpha_i[q] \text{ for } i \in \{1, 2, \ldots, m\} \text{ as binary} \\ \mathcal{M}.var \leftarrow \beta[1], \beta[2], \ldots, \beta[q] \text{ as binary} \\ \mathcal{M}.con \leftarrow k_b = \max(k_{a_1}, k_{a_2}, \ldots, k_{a_m}) \\ \mathcal{M}.con \leftarrow \beta[j] = \max(\alpha_1[j], \alpha_2[j], \ldots, \alpha_m[j]) \text{ for } 1 \le j \le q \end{cases}$$

are sufficient to describe the propagation of the division property for AND.

Based on Propositions 5, 6 and 7, we can build an MILP model $\mathcal{M}$ which covers all the possible division trails over $(\mathbb{F}_2 \times T_I)^n$ w.r.t I. Then, by adding the constraint requiring $\lambda_j^{(t)} \neq T$ to $\mathcal{M}$, we can remove all the invalid division trails due to the absence of $\lambda_j^{(t)}$.

It can be seen that if an MILP model $\mathcal{M}$ which covers all the conventional division trails contains N variables, then an MILP model $\mathcal{M}'$ which covers all the possible division trails over $(\mathbb{F}_2 \times T_I)^n$ will contain approximately $N + qN$ variables. It is well known that a large number of variables in an MILP model means long solving time. Therefore, we propose a small technique to reduce the number of variables in our MILP models.

4.2.2 Reducing variables in MILP models

Let $Y = f(X)$ be a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and $I \subseteq \{1, 2, \ldots, n\}$ be a set of active variable indexes. Let $(\boldsymbol{k}_0, \boldsymbol{\lambda}_0) \xrightarrow{f} (\boldsymbol{k}_1, \boldsymbol{\lambda}_1) \xrightarrow{f} \cdots \xrightarrow{f} (\boldsymbol{k}_r, \boldsymbol{\lambda}_r)$ be a division trail. Suppose T is a product of some active variables and T does not appear in $s_p^{(t)}$. It is clear that if $\gcd(\lambda_p^{(t)}, T) = T$, that is $\lambda_p^{(t)}$ is divisible by T, then this trail is invalid. Then to simplify our MILP models, we propose division trails with the restriction of T, namely, $\boldsymbol{\lambda}_{0,T} \xrightarrow{f} \boldsymbol{\lambda}_{1,T} \xrightarrow{f} \cdots \xrightarrow{f} \boldsymbol{\lambda}_{r,T}$, where

$$\boldsymbol{\lambda}_{i,T} = \gcd(T, \boldsymbol{\lambda}_i) = (\gcd(T, \lambda_1^{(i)}), \gcd(T, \lambda_2^{(i)}), \ldots, \gcd(T, \lambda_n^{(i)})), 0 \leq i \leq r.$$

Denote the j-th component of $\boldsymbol{\lambda}_{i,T}$ by $\lambda_{j,T}^{(i)}$, i.e., $\lambda_{j,T}^{(i)} = \gcd(T, \boldsymbol{\lambda}_j^{(i)})$, where $1 \leq j \leq n$ and $0 \leq i \leq r$. It can be seen that $\lambda_{j,T}^{(i)}$ is a divisor of T for $0 \leq j \leq n$ and $0 \leq i \leq r$. Moreover, since $\prod_{j=1}^{n} \lambda_j^{(i)} = \prod_{i \in I} x_i$ for $0 \leq i \leq r$, it follows that $\prod_{j=1}^{n} \lambda_{j,T}^{(i)} = T$.

To illustrate our main idea of division trails with the restriction of T more clearly, we provide the a toy example, see Example 3 in the Appendix.

MILP models to describe the propagation rules of the division property with the restriction of T for some product T of active variables are the same as Propositions 5, 6, and 7. But with the restriction of T, q will be quite small, and so the number of variables in an MILP model will be much smaller. For example, in some of our experiments on Trivium, when the number of active variables is more than 70, T only has 2 active variables.

With this new technique, we suggest to remove invalid trails after identifying a target product of variables, see Algorithm 2 for details. In Algorithm 2, we first attempt to find a conventional division trail γ satisfying the previous propagation rules. If the division trail γ is invalid due to the absence of $\lambda_p^{(t)} = T$, then we record (T, t, p) in VT and we remove all the invalid division trails according to the information recorded in VT.

Remark 5. So far in all our experiments on Trivium, we have $|VT| = 1$ for each given cube in Algorithm 2, see Table 3.

5 Experimental Results

In this section, we apply our new attack framework to the round-reduced Trivium. We first show how to build MILP models for Trivium. Then, we perform various experiments on round-reduced Trivium.

5.1 Specification of Trivium

Trivium is a bit oriented synchronous stream cipher designed by Cannière and Preneel, which was selected as one of eSTREAM hardware-oriented portfolio ciphers. The main building block of Trivium is a 288-bit Galois nonlinear feedback shift register. For every clock cycle there are three bits of the internal state updated by quadratic feedback functions and all the other bits of the internal sate are updated by shifting. The internal state of Trivium, denoted by $(s_1, s_2, \ldots, s_{288})$, is initialized by loading an 80-bit secret key and an 80-bit IV into the registers, and setting all the remaining bits to 0 except for the

Algorithm 2 Evaluate involved key variables by MILP models

procedure IdentifyInvolvedKeyVariables(cube indices I)
  Declare an empty MILP model $\mathcal{M}$;
  Let $\boldsymbol{x}$ be n MILP variables of $\mathcal{M}$ corresponding to secret variables;
  Let $\boldsymbol{v}$ be m MILP variables of $\mathcal{M}$ corresponding to public variables;
  $\mathcal{M}.con \leftarrow v_i = 1$ for all $i \in I$;
  $\mathcal{M}.con \leftarrow v_i = 0$ for all $i \in (\{1, 2, \ldots, m\} \setminus I)$;
  $\mathcal{M}.con \leftarrow \sum_{i=1}^{n} x_i = 1$;
  Build an MILP model $\mathcal{M}$ which covers all the possible conventional division property trails propagating through the target cipher;
  Set J and VT to $\emptyset$;
  Solve the MILP model $\mathcal{M}$;
  while $\mathcal{M}$ is feasible **do**
    Check the validity of the reported conventional division trail (denoted by γ);
    if γ is invalid **then**
      Set $VT = VT \cup \{(T, t, p)\}$, where γ is invalid due to that T does not appear in $s_p^{(t)}$;
      Rebuild the MILP model $\mathcal{M}$ covering all the new division trails with the restriction of T;
      Add the constraint requiring $\lambda_p^{(t)} \neq T$ to $\mathcal{M}$ for every $(T, t, p) \in VT$;
      $\mathcal{M}.con \leftarrow v_i = 1$ for all $i \in I$;
      $\mathcal{M}.con \leftarrow v_i = 0$ for all $i \in (\{1, 2, \ldots, m\} \setminus I)$;
      $\mathcal{M}.con \leftarrow \sum_{i=1}^{n} x_i = 1$;
      $\mathcal{M}.con \leftarrow x_j = 0$ for all $j \in J$;
      Solve $\mathcal{M}$;
    else
      Set $J = J \cup j$, where $j \in \{1, 2, \ldots, n\}$ s.t. $x_j = 1$;
      $\mathcal{M}.con \leftarrow x_j = 0$ for all $j \in J$;
      Solve $\mathcal{M}$;
    end if
  end while
  return J
end procedure

Algorithm 3 Pseudo-code of Trivium

$(s_1, s_2, \ldots, s_{93}) \leftarrow (x_1, x_2, \ldots, x_{80}, 0, \ldots, 0)$;
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (v_1, v_2, \ldots, v_{80}, 0, \ldots, 0)$;
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$;
for i from 1 to N **do**
  $t_1 \leftarrow s_{66} \oplus s_{93} \oplus s_{91} \cdot s_{92} \oplus s_{171}$;
  $t_2 \leftarrow s_{162} \oplus s_{177} \oplus s_{175} \cdot s_{176} \oplus s_{264}$;
  $t_3 \leftarrow s_{243} \oplus s_{288} \oplus s_{286} \cdot s_{287} \oplus s_{69}$;
  if $i > 1152$ **then**
    $z_{i-1152} \leftarrow s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288}$;
  end if
  $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$;
  $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$;
  $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$;
end for

last three bits of the third register. Then, the algorithm would not output any keystream bit until the internal state is updated $4 \times 288 = 1152$ rounds. This is described by the pseudo-code shown in Algorithm 3. For more details, please refer to [23].

5.2 Reducing MILP Models with Degree Evaluations

Let I be non-empty set whose elements are indices of active variables. Denote by $\mathbb{X}_I$ the input set where all active variables indexed by I take all possible combinations of 0/1 and the remaining variables take constants.

In our attacks, when targeting z_r, we should check the existence of a conventional division trail such that $\boldsymbol{k}_I = \boldsymbol{k}_0 \xrightarrow{z_r} 1$, where $\boldsymbol{k}_I$ is the division property of $\mathbb{X}_I$ and z_r is the first output bit of an r-round iterative cipher E_r. In previous papers, to check the

existence of a such conventional division trail, it needs to build and solve an MILP model $\mathcal{M}_r$ covering all the possible conventional division trails propagating through r rounds and the output function. Since many variables and constraints are added to describe the propagation of division property through each round, $\mathcal{M}_r$ usually consists of a large amount of variables and constraints and is difficult to solve. In this section, we try to divide such a large MILP model into several smaller ones, and the time complexity could be reduced with the help of degree evaluation method in [16].

Let $f(\boldsymbol{x})$ be a polynomial from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Assume the ANF of $f(x)$ is represented as

$$f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}}^f \cdot \boldsymbol{x}^{\boldsymbol{u}}.$$

where $a_{\boldsymbol{u}}^f \in \mathbb{F}_2$ denotes the ANF coefficient of x^u. Then the following Lemma is an apparent property of the division property which was ever used in line 13 of Algorithm 2 in [5].

Lemma 4. *Let $\boldsymbol{k}$ be an n-dimensional bit vector. If for any $\boldsymbol{u}$ with $a_{\boldsymbol{u}}^f = 1$ there is no division trail such that $\boldsymbol{k} \xrightarrow{\boldsymbol{x}^{\boldsymbol{u}}} 1$, then there is no division trial such that $\boldsymbol{k} \xrightarrow{f} 1$.*

For some $0 \leq t < r$, z_r can be written as a polynomial on $s^{(t)} = (s_1^{(t)}, s_2^{(t)}, \ldots, s_n^{(t)})$ uniquely, i.e., $z_r = g_t(s^{(t)})$, where $s^{(t)}$ is the internal state of E_r after t rounds. Based on Lemma 4, we know whether there exists a division trail $\boldsymbol{k}_0 \xrightarrow{z_r} 1$ by checking whether there exists a division trail $\boldsymbol{k}_0 \xrightarrow{u} 1$ for each term $u = \prod_{j=1}^{h} s_{i_j}^{(t)}$ of g_t. Thus, for each term u, we only need to build an MILP model which covers all the possible trails propagating through t rounds and check the existence of division trail $\boldsymbol{k}_0 \xrightarrow{u} 1$.

However, one problem to apply Lemma 4 is that there may be too many terms in g_t. Our solution is using the degree evaluation method based on the numeric mapping. For the chosen set I, if $\deg_I(u) < |I|$, then it is clear that $\bigoplus_{\mathbb{X}_I} u = 0$. Namely, for such a term u, there would be no division trail $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{u} 1$, where $\boldsymbol{e}_j$ is a binary vector whose only the j-th element is 1. Thus, by utilizing the degree evaluation method, we could easily remove a large amount of terms of g_t, and so the number of small MILP models needed to be solved could be reduced dramatically. Due to the high efficiency of degree evaluation method, the total time would be much less than that of solving a large MILP model.

Combining all the above techniques, we propose a new attack framework in Algorithm 4. In Algorithm 4, RT is the set of terms whose evaluated degrees reach $|I|$. For each term u in RT, we call Algorithm 2 to determine a set J_u, where the superpoly of C_I in u is only related to x_j's ($j \in J_u$) for arbitrary assignments of non-cube variables. After knowing J_u for each term u, the key variables involved in the superpoly of C_I in z_r can be determined by $J = \bigcup_{u \in RT} J_u$.

Remark 6. Since MILP solvers are used, it is hard to estimate the complexities of Algorithm 4 and the previous algorithms accurately. However, in our experiments (targeting more than 830-round Trivium with large cubes), when checking the existence of a certain division trail, the total time of our algorithms is less than the time of solving a large MILP model directly. This is because that many terms of g_t are discarded by the degree evaluation method efficiently and small MILP models are much easier to solve. Furthermore, with Algorithm 4, the existence of a certain division trail can be checked in parallel.

5.3 MILP Models for Trivium

In this subsection, for Trivium, we would show how to build MILP models with respect to the conventional division property and our new division property.

Assume that z_r is expressed as $z_r = g_t(s^{(t)})$ for some $0 \leq t < r$. Due to Lemma 4, we only need to consider the propagation of the division property through t rounds. In

Algorithm 4 New attack framework

Require: The set of cube indices I
Express the output bit z_r as $z_r = g_t(s^{(t)})$ for some t;
Set J to $\emptyset$;
Compute the set $RT = \{u | \mathrm{DEG}_I(u) \geq |I|\}$, where $\mathrm{DEG}_I(u)$ is the evaluated degree of u w.r.t. I;
for each term u in RT **do**
    Figure out the set J_u with Algorithm 2, where the superpoly of C_I in u is only related to x_j's ($j \in J_u$) for arbitrary assignments of non-cube variables;
    Set $J = J \cup J_u$;
end for
return J;

Algorithm 5 Original MILP model of division property for the round function of Trivium

procedure TriviumCoreOriginal($\mathcal{M}$,$\boldsymbol{x}$,$\boldsymbol{W}$,i_1,i_2,i_3,i_4,i_5,$\boldsymbol{flag}$,i)
    $\mathcal{M}.var \leftarrow y_{5\cdot i+j}(1 \leq j \leq 5), z_{4\cdot i+j}(1 \leq j \leq 4), a_i$ as binary;
    $\mathcal{M}.con \leftarrow y_{5\cdot i+j} = x_{i_j} - z_{4\cdot i+j}(1 \leq j \leq 4)$;
    $\mathcal{M}.con \leftarrow a_i = \max(z_{4\cdot i+3}, z_{4\cdot i+4})$;
    if either $flag_{i_3} = 0$ or $flag_{i_4} = 0$ **then**
        $\mathcal{M}.con \leftarrow a_i = 0$;
    end if
    $\mathcal{M}.con \leftarrow y_{5\cdot i+5} = x_{i_5} + a_i + z_{4\cdot i+1} + z_{4\cdot i+2}$;
    Set $w_j = x_j$ for all $j \in \{1, 2, \ldots, 288\}$ w/o i_1, i_2, i_3, i_4, i_5;
    $w_{i_j} = y_{5\cdot i+j}$ for $1 \leq j \leq 5$;
    return $\boldsymbol{w}$
end procedure

Algorithm 6 MILP models of the division property over $(\mathbb{F}_2 \times T_I)^n$ with restriction of T for the round function of Trivium

procedure TriviumCoreDV($\mathcal{M}$,$\boldsymbol{x}$,$\boldsymbol{W}$,i_1,i_2,i_3,i_4,i_5,$\boldsymbol{flag}$,i)
    $\mathcal{M}.var \leftarrow y_{5\cdot i+j}(1 \leq j \leq 5), z_{4\cdot i+j}(1 \leq j \leq 4), a_i$ as binary;
    for $1 \leq L \leq q$ **do**
        $\mathcal{M}.var \leftarrow y^L_{5\cdot i+j}(1 \leq j \leq 5), z^L_{4\cdot i+j}(1 \leq j \leq 4), a^L_i$ as binary;
    end for
    $\mathcal{M}.con \leftarrow y_{5\cdot i+j} = x_{i_j} - z_{4\cdot i+j}$ for all $1 \leq j \leq 4$;
    $\mathcal{M}.con \leftarrow a_i = \max(z_{4\cdot i+3}, z_{4\cdot i+4})$;
    $\mathcal{M}.con \leftarrow y_{5\cdot i+5} = x_{i_5} + a_i + z_{4\cdot i+1} + z_{4\cdot i+2}$;
    ▷ Describe the propagation of our new division property for COPY.
    $\mathcal{M}.con \leftarrow y^L_{5\cdot i+j} = \min(\boldsymbol{W}[i_j][L], y_{5\cdot i+j})$ for $1 \leq j \leq 4$ and $1 \leq L \leq q$;
    $\mathcal{M}.con \leftarrow z^L_{4\cdot i+j} = \min(\boldsymbol{W}[i_j][L], z_{4\cdot i+j})$ for $1 \leq j \leq 4$ and $1 \leq L \leq q$;
    ▷ Describe the propagation of our new division property for AND.
    $\mathcal{M}.con \leftarrow a^L_i = \max(z^L_{4\cdot i+3}, z^L_{4\cdot i+4})$ for $1 \leq L \leq q$;
    ▷ Describe the propagation of our new division property for XOR.
    $\mathcal{M}.con \leftarrow y^L_{5\cdot i+5} = W^L_{i_5} + a^L_i + z^L_{4\cdot i+1} + z^L_{4\cdot i+2}$ for $1 \leq L \leq q$;
    if either $flag_{i_3} = 0$ or $flag_{i_4} = 0$ **then**
        $\mathcal{M}.con \leftarrow a_i = 0$;
        $\mathcal{M}.con \leftarrow a^L_i = 0$ for $1 \leq L \leq q$;
    end if
    for all $j \in \{1, 2, \ldots, 288\}$ w/o i_1, i_2, i_3, i_4, i_5 **do**
        $w_j = x_j$;
        $\boldsymbol{U}[j][L] = \boldsymbol{W}[j][L]$ for $1 \leq L \leq q$;
    end for
    Set $w_{i_j} = y_{5\cdot i+j}$ for $1 \leq j \leq 5$;
    Set $\boldsymbol{U}[i_j][L] = y^L_{5\cdot i+j}$ for $1 \leq j \leq 5$ and $1 \leq L \leq q$;
end procedure
return $(\mathcal{M}, \boldsymbol{w}, \boldsymbol{U})$

Algorithm 5, we describe the the propagation of the conventional division property for the round function of Trivium. Hence, we could build MILP models for each term u of g_t to check the existence of the division trail $\boldsymbol{k}_I \xrightarrow{u} 1$, where $\boldsymbol{k}_I$ is the initial division property.

Once we find an invalid division trail due to that T does not appear in $s_p^{(r_0)}$, the next

Algorithm 7 MILP models of the new division property with restriction of T of a target term u against Trivium

procedure TriviumEval(round t where z_r is expressed as $z_r = g_t(s^{(t)})$, the term $T = \prod_{j=1}^{q} v_{i_j}$ which does not appear in $s_p^{(r_0)}$, the target term $u = \prod_{i=1}^{h} s_{j_i}^{(t)}$ of g_t)
  Declare an empty MILP model $\mathcal{M}$;
  $\mathcal{M}.var \leftarrow \boldsymbol{s}_i^0$ for all $i \in \{1, 2, \ldots, 288\}$;
  Set $flag_i = \delta$ for $i \in \{1, 2, \ldots, 80, 94, 95, \ldots, 173\}$;
  Set $flag_i = 0_c$ for $i \in \{81, 82, \ldots, 93, 174, 175, \ldots, 285\}$;
  Set $flag_i = 1_c$ for $i \in \{286, 287, 288\}$;
  $\mathcal{M}.var \leftarrow \boldsymbol{W}[0][i][j]$ for all $i \in \{1, 2, \ldots, 288\}$ and all $j \in \{1, 2, \ldots, q\}$;
  $\mathcal{M}.con \leftarrow \boldsymbol{W}[0][93 + i_j][j] = 1$ for all $j \in \{1, 2, \ldots, q\}$;
  Add the constraints that require all the remaining variables in $\boldsymbol{W}[0]$ equal to 0;
  for $i = 1$ to r_0 **do**
    $(\mathcal{M}, \boldsymbol{x}, \boldsymbol{A})$=**TriviumCoreDV**$(\mathcal{M}, \boldsymbol{s}^{i-1}, \boldsymbol{W}[i-1], 66, 171, 91, 92, 93, \boldsymbol{flag}, 3 \cdot i + 1)$;
    $(\mathcal{M}, \boldsymbol{y}, \boldsymbol{B})$=**TriviumCoreDV**$(\mathcal{M}, \boldsymbol{x}, \boldsymbol{A}, 162, 264, 175, 176, 177, \boldsymbol{flag}, 3 \cdot i + 2)$;
    $(\mathcal{M}, \boldsymbol{z}, \boldsymbol{C})$=**TriviumCoreDV**$(\mathcal{M}, \boldsymbol{y}, \boldsymbol{B}, 243, 69, 286, 287, 288, \boldsymbol{flag}, 3 \cdot i + 3)$;
    Set $\boldsymbol{s}^i = \boldsymbol{z} >>> 1$;
    UpdateFlag($\boldsymbol{flag}$); ▷ See the Appendix for the detail specification;
    Set $\boldsymbol{W}[i][j][L] = \boldsymbol{C}[j-1][L]$, for $2 \le j \le 288$ and $1 \le L \le q$;
    Set $\boldsymbol{W}[i][1][L] = \boldsymbol{C}[288][L]$ for $1 \le L \le q$;
  end for
  for $i = r_0 + 1$ to t **do**
    $(\mathcal{M}, \boldsymbol{x})$=**TriviumCoreOriginal**$(\mathcal{M}, \boldsymbol{s}^{i-1}, 66, 171, 91, 92, 93, \boldsymbol{flag}, 3 \cdot i + 1)$;
    $(\mathcal{M}, \boldsymbol{y})$=**TriviumCoreOriginal**$(\mathcal{M}, \boldsymbol{x}, 162, 264, 175, 176, 177, \boldsymbol{flag}, 3 \cdot i + 2)$;
    $(\mathcal{M}, \boldsymbol{z})$=**TriviumCoreOriginal**$(\mathcal{M}, \boldsymbol{y}, 243, 69, 286, 287, 288, \boldsymbol{flag}, 3 \cdot i + 3)$;
    Set $\boldsymbol{s}^i = \boldsymbol{z} >>> 1$;
    UpdateFlag($\boldsymbol{flag}$);
  end for
  $\mathcal{M}.con \leftarrow s_i^t = 0$ for all $i \in \{1, 2, \ldots 288\}$ w/o $\{j_1, j_2, \ldots, j_h\}$;
  $\mathcal{M}.con \leftarrow \sum_{i=1}^{h} s_{j_i}^t \le h$; ▷ Add the constraint corresponding to u;
  $\mathcal{M}.con \leftarrow \sum_{L=1}^{q} \boldsymbol{W}[r_0][p][L] - q < 0$; ▷ Add the constraint requiring $\lambda_{p,T}^{(r_0)} \neq T$;
  return $\mathcal{M}$;
end procedure

step of our attack is to remove all the invalid division trails due to the absence of T. Based on Lemma 4, for each term u of g_t, we only need to build an MILP model which covers all the new possible division trails with restriction of T propagating through t rounds. Hence, in Algorithm 7, for a target term $u = \prod_{j=1}^{h} s_{i_j}^{(t)}$, we introduce how to build the MILP model which covers all the new division trails with the restriction of T, where the procedure **TriviumCoreDV**, presented in Algorithm 6 formally, is used to describe the propagation of our new division property with the restriction of T for the round function of Trivium. Note that, to remove all the invalid division trails due to the absence of T, we only need to know whether $\lambda_{p,T}^{(r_0)} = T$. Hence, in Algorithm 7, we call Algorithm 6 for the first r_0 rounds with respect to the propagation of the new division property with the restriction of T and call Algorithm 5 to consider the propagation of the conventional division property for the $(r_0 + 1)$-th round to the t-th round.

Remark 7. The flag technique in [9] is used to remove some invalid conventional division trails by considering the effect of specific values of the non-active variables on the propagation of the division property. Note that the main idea of our new division property is to record the distribution of active variables for a conventional division trail. We can still use our techniques to record the distribution of active variables of the division trails which are valid under the flag technique.

5.4 Experimental Verification

In this subsection, we provide a small example to illustrate our attacks in detail and verify our techniques.

Example 1. We choose v_1, v_2, v_{45}, v_{46} as active variables and the output bit after 308 rounds as a target, i.e., $z^{(308)} = s_{66}^{(308)} \oplus s_{93}^{(308)} \oplus s_{162}^{(308)} \oplus s_{177}^{(308)} \oplus s_{243}^{(308)} \oplus s_{288}^{(308)}$.

First, we compute the polynomial representation of $z^{(308)}$ on the internal state bits after 239-round initialization. That is

$$\begin{aligned} z^{(308)} &= s_{283}^{(239)} s_{284}^{(239)} \oplus s_{285}^{(239)} \oplus s_{240}^{(239)} \oplus s_{27}^{239} \oplus s_{91}^{(239)} s_{92}^{(239)} \oplus s_{93}^{(239)} \\ &\quad \oplus s_{177}^{(239)} \oplus s_{108}^{(239)} \oplus s_{172}^{(239)} s_{173}^{(239)} \oplus s_{174}^{(239)} \oplus s_{159}^{(239)} \oplus s_{261}^{(239)} \oplus s_{219}^{(239)}. \end{aligned}$$

Second, by using the degree evaluation method, we get a set of terms $RT = \{s_{91}^{(239)} s_{92}^{(239)}\}$, where the evaluated degree of each term in RT reaches 4.

Third, for each term u in RT, we check whether there exists a conventional division trail $\boldsymbol{k} \xrightarrow{u} 1$, where $\boldsymbol{k} \in \mathbb{F}_2^{288}$, $k_{93+i} = 1$ for $i \in \{1, 2, 45, 46\}$, $k_i = 0$ otherwise. For simplicity, we denote by FRT the set of terms satisfying the above condition. In this case, we have $FRT = \{s_{91}^{(239)} s_{92}^{(239)}\}$. For $s_{91}^{(239)} s_{92}^{(239)}$, we obtain a conventional division trail $\boldsymbol{k} = \boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \ldots \to \boldsymbol{k}_{239} \to 1$, denoted by γ, where $k_{239}[i] = 1$ for $i \in \{91, 92\}$ and $k_{239}[i] = 0$ otherwise.

Fourth, we recover $[\boldsymbol{k}_0, \boldsymbol{\lambda}_0] \to [\boldsymbol{k}_1, \boldsymbol{\lambda}_1] \to \cdots \to [\boldsymbol{k}_{239}, \boldsymbol{\lambda}_{239}] \to [1, v_1 v_2 v_{45} v_{46}]$ according to γ and check whether this trail is valid or not. The experiment shows that γ is invalid due to $\lambda_1^{(149)} = v_{45} v_{46}$ not appearing in s_1^{149}.

Finally, we build an MILP model which covers all our new division trials with the restriction of $T = v_{45} v_{46}$ and requires that $\lambda_{1,T}^{(149)} \neq v_{45} v_{46}$. Then we check whether there exists a new division trail $[\boldsymbol{k}_0, \boldsymbol{\lambda}_0] \to [\boldsymbol{k}'_{239}, \boldsymbol{\lambda}'_{239}] \to [1, v_1 v_2 v_{45} v_{46}]$. Our experimental result shows that there does not exist such a division trail. This indicates that $v_1 v_2 v_{45} v_{46}$ does not appear in $z^{(308)}$.

To verify our result, we practically compute the ANF of $z^{(308)}$, and it is shown that $v_1 v_2 v_{45} v_{46}$ does not appear in $z^{(308)}$, which is in accordance with our result.

5.5 Checking Key Recovery Attacks on Trivium

In [5, 9], for a cube C_I where I is an index set of active IV variables, the authors used division property to identify a set J of key variables which includes all possible key variables that are involved in the superpoly of C_I. Based on Assumption 1, it was previously regarded that the resultant superpoly could be nonconstant by assigning appropriate values to non-cube IV variables and some key information could be recovered once the resultant superpoly was recovered. However, note that it is also possible that the resultant superpoly is only a constant function and in such case no key information would be recovered. Therefore, we propose that key recovery attacks given by division property based cube attacks should be verified.

We applied our new attack framework to verify the key-recovery attacks against Trivium variants with more than 830 initialization rounds reported in [5, 9], it was shown that all these key-recovery attacks were only distinguishing attacks. Some important parameters used in our experiments are presented in Table 3, where the expressions of $u_1, u_2 \ldots, u_8$ are listed in Appendix. Take the first row of Table 3 for example. Let I be the cube index corresponding to the first row of Table 3. First, when the output bit z_{832} is considered as a polynomial on the internal state $s^{(312)}$, i.e., $z_{832} = g_{312}(s^{(312)})$, there is only one term u_1 satisfying that a division trail $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{u_1} 1$ exists for $j = 34$[6]. Let

[6]Here we did not directly compute the ANF of $z_{832} = g_{312}(s^{(312)})$ but used Lemma 4 recursively.

Table 3: Details of the checking key recovering attacks

# of rounds	$\|I\|$	Involved key variables	RT	VT/internal state bit	checking result
832	72	$k_{34}, k_{58}, k_{59}, k_{60}, k_{61}$[5]	u_1	$v_{37}v_{38}/s_{94}^{(205)}$	no key variable
833	73	$k_{49}, k_{58}, k_{60}, k_{64}$, k_{74}, k_{75}, k_{76}[9]	u_2 u_3	$v_{45}v_{46}/s_{1}^{(149)}$ $v_{23}v_{37}v_{38}/s_{94}^{(249)}$	no key variable
833	74	k_{60}[9]	u_4	$v_{29}v_{30}/s_{1}^{(165)}$	no key variable
835	77	k_{57}[9]	u_5 u_6	$v_{72}v_{73}/s_{1}^{(122)}$ $v_{50}v_{51}/s_{1}^{(169)}$	no key variable
836	78	k_{57}[9]	u_7	$v_{71}v_{72}/s_{1}^{(123)}$	no key variable
839	78	k_{61}[9]	u_8	$v_{61}v_{75}v_{76}/s_{94}^{(262)}$	no key variable

Table 4: Details on finding zero-sum distinguishers

# of rounds	RT	VT/internal state bit	RT	VT/internal state bit
840	u_9 u_{10}	$v_{30}v_{31}/s_{1}^{(164)}$	u_{11} u_{12}	$v_{74}v_{75}v_{76}/s_{94}^{(262)}$

us denote this division trail by γ_{u_1}. Second, γ_{u_1} is invalid due to $v_{37}v_{38}$ does not appear in $s_{94}^{(205)}$. Third, after removing all the invalid division trails due to the absence of $v_{37}v_{38}$ in $s_{94}^{(205)}$, there is no division trail such that $(\boldsymbol{e}_j, \boldsymbol{k}_I) \xrightarrow{u_1} 1$ for any $j \in \{34, 58, 59, 60, 61\}$. Therefore, no key variables are involved in the superpoly of the cube I.

5.6 Improvements on Finding Zero-Sum Distinguishers

Although invalid trails would not affect the correctness of distinguishers found with the division property, we may find better distinguishers if some invalid division trails can be removed. In [15], it was reported that the first output bit of 839-round Trivium is balanced when all the IV variables are active. However, there may be some invalid division trails not being removed in the previous works. Therefore, in this subsection, we attempt to find distinguishers for Trivium variants with more initialization rounds than 839.

We perform the experiments in a similar way to that used in the previous subsection. As a result, we prove that the degree of the first output bit of 840-round Trivium is balanced when all the IV variables are active. The detailed information of our experiments is shown in Table 4, and the detailed expressions of $u_9, u_{10}, u_{11}, u_{12}$ can be found in Appendix.

6 Conclusion

In this paper we revisit the key recovery attacks given by the division property based cube attacks against stream ciphers. Weak Assumption is one of fundamental assumptions used in this type of attacks. Based on Weak Assumption, it was thought that if a subset J of the secret key variables was suggested to be involved in a target superpoly by the MILP modelled division property, then the superpoly would be a non-constant function by choosing an appropriate constant part of IV and so it could be used for recovering key information. However, our observations show that Weak Assumption does not always hold for Trivium and the target superpoly could be a constant function for all choices of the constant part of IV in practice. It is shown in this paper that some best key recovery attacks on Trivium are distinguishing attacks only. To attain this goal, we proposed a new variant of division property from an algebraic point of view and some new techniques to remove invalid division trails by MILP models. For the state of art, division property based cube attacks could only guarantee distinguishing attacks, and all key recovery results need further verification.

References

[1] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 278–299, 2009.

[2] Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 502–517, 2013.

[3] Piotr Mroczkowski and Janusz Szmidt. Corrigendum to: The cube attack on stream cipher Trivium and quadraticity tests. *IACR Cryptology ePrint Archive*, 2011:32, 2011.

[4] Chen-Dong Ye and Tian Tian. A new framework for finding nonlinear superpolies in cube attacks against trivium-like ciphers. In Willy Susilo and Guomin Yang, editors, *Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings*, volume 10946 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2018.

[5] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, pages 250–279, 2017.

[6] Yosuke Todo. Structural evaluation by generalized integral property. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 287–314, 2015.

[7] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, pages 357–377, 2016.

[8] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In *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*, pages 648–678, 2016.

[9] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 275–305, 2018.

[10] Itai Dinur and Adi Shamir. Breaking grain-128 with dynamic cube attacks. In *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, pages 167–187, 2011.

[11] Meicheng Liu, Jingchun Yang, Wenhao Wang, and Dongdai Lin. Correlation cube attacks: From weak-key distinguisher to key recovery. In *Advances in Cryptology*

- EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II, pages 715–744, 2018.

[12] Itai Dinur, Tim Güneysu, Christof Paar, Adi Shamir, and Ralf Zimmermann. An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 327–343, 2011.

[13] Ximing Fu, Xiaoyun Wang, Xiaoyang Dong, and Willi Meier. A key-recovery attack on 855-round trivium. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 160–184, 2018.

[14] Yonglin Hao, Lin Jiao, Chaoyun Li, Willi Meier, Yosuke Todo, and Qingju Wang. Observations on the dynamic cube attack of 855-round trivium from crypto'18. Cryptology ePrint Archive, Report 2018/972, 2018. https://eprint.iacr.org/2018/972.

[15] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. *IEEE Trans. Computers*, 67(12):1720–1736, 2018.

[16] Meicheng Liu. Degree evaluation of NFSR-Based cryptosystems. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, pages 227–249, 2017.

[17] Zonghao Gu, Edward Rothberg, and Robert Bixby. Gurobi optimizer. http://www.gurobi.com/.

[18] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, pages 57–76, 2011.

[19] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 158–178, 2014.

[20] Meiqin Wang Peng Wang Kexin Qiao Xiaoshuang Ma Danping Shi Ling Song Kai Fu Siwei Sun, Lei Hu. Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747, 2014. https://eprint.iacr.org/2014/747.

[21] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 185–215, 2017.

[22] Ling Sun, Wei Wang, and Meiqin Wang. Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. Cryptology ePrint Archive, Report 2016/811, 2016. https://eprint.iacr.org/2016/811.

[23] Christophe De Cannière and Bart Preneel. Trivium. In *New Stream Cipher Designs - The eSTREAM Finalists*, pages 244–266. 2008.

Appendix

The Toy Example I

Example 2. Let nf be a toy NFSR-based stream cipher of size 8. Denote the state of nf at time clock t by $s^{(t)} = (s_1^{(t)}, s_2^{(t)}, \cdots, s_8^{(t)})$. Assume that nf is initialized with $x_1, x_2, \cdots, x_8$, i.e., $s^{(0)} = (x_1, x_2, \ldots, x_8)$. The state of nf is updated as

$$\begin{aligned} s_1^{(t+1)} &= s_5^{(t)} \oplus s_6^{(t)} s_7^{(t)} \oplus s_8^{(t)} \\ s_i^{(t+1)} &= s_{i-1}^{(t)} \text{ for } i \in \{2,3,4,5,6,7,8\} \end{aligned}$$

Furthermore, the output bit after r rounds is defined as $z_r = s_1^{(r)} s_3^{(r)}$.

We choose x_2, x_3, x_4, x_5 as active variables. Then, according to the propagation rules of the conventional division property, the following conventional division trail would exist.

round 0	$(0,1,1,1,1,0,0,0)$
round 1	$(1,0,1,1,1,0,0,0)$
round 2	$(1,1,0,1,1,0,0,0)$
round 3	$(0,1,1,0,1,1,0,0)$
round 4	$(0,0,1,1,0,1,1,0)$
round 5	$(1,0,0,1,1,0,0,0)$
round 6	$(0,1,0,0,1,1,0,0)$
round 7	$(0,0,1,0,0,1,1,0)$
round 8	$(1,0,0,1,0,0,0,0)$

For the above conventional division trail, based on our ideas, we could know the distribution of active variables through this division trail. We show the corresponding division trail over $(\mathbb{F}_2, T_I)^8$ in the following.

round 0	$([0,1],[1,x_2],[1,x_3],[1,x_4],[1,x_5],[0,1],[0,1],[0,1])$
round 1	$([1,x_5],[0,1],[1,x_2],[1,x_3],[1,x_4],[0,1],[0,1],[0,1])$
round 2	$([1,x_4],[1,x_5],[0,1],[1,x_2],[1,x_3],[0,1],[0,1],[0,1])$
round 3	$([0,1],[1,x_4],[1,x_5],[0,1],[1,x_2],[1,x_3],[0,1],[0,1])$
round 4	$([0,1],[0,1],[1,x_4],[1,x_5],[0,1],[1,x_2],[1,x_3],[0,1])$
round 5	$([1,x_2x_3],[0,1],[0,1],[1,x_4],[1,x_5],[0,1],[0,1],[0,1])$
round 6	$([0,1],[1,x_2x_3],[0,1],[0,1],[1,x_4],[1,x_5],[0,1],[0,1])$
round 7	$([0,1],[0,1],[1,x_2x_3],[0,1],[0,1],[1,x_4],[1,x_5],[0,1])$
round 8	$([1,x_4x_5],[0,1],[0,1],[1,x_2x_3],[0,1],[0,1],[0,1],[0,1])$

The above division trail indicates that

$$\bigoplus_{(x_2,x_3,x_4,x_5)\in\mathbb{F}_2^4} z_8$$

is unknown, where $z_8 = s_1^{(8)} s_3^{(8)}$. However, by computing the real ANFs of $s^{(t)}$ for $0 \leq t \leq 8$, we know that x_4x_5 does not appear in the $s_1^{(8)}$ and $x_2x_3x_4x_5$ does not appear in z_8. It indicates that the above division trail is invalid and should be aborted. Furthermore, it can be seen that with our techniques, we could identify some invalid division trails which could not be found previously.

The Toy Example II

Example 3. Let nf be defined as Example 2. When choosing x_2, x_3, x_4, x_5 as active variables, as shown in Example 2, there would one division trail. Since $T = x_4x_5$ does not appear in $s_1^{(8)}$, we would focus on the division trails with the restriction of T. In the following, for the division trail shown in Example 2, we present its corresponding division trail with the restriction of T.

round 0	$([0,1],[1,1],[1,1],[1,x_4],[1,x_5],[0,1],[0,1],[0,1])$
round 1	$([1,x_5],[0,1],[1,1],[1,1],[1,x_4],[0,1],[0,1],[0,1])$
round 2	$([1,x_4],[1,x_5],[0,1],[1,1],[1,1],[0,1],[0,1],[0,1])$
round 3	$([0,1],[1,x_4],[1,x_5],[0,1],[1,1],[1,1],[0,1],[0,1])$
round 4	$([0,1],[0,1],[1,x_4],[1,x_5],[0,1],[1,1],[1,1],[0,1])$
round 5	$([1,1],[0,1],[0,1],[1,x_4],[1,x_5],[0,1],[0,1],[0,1])$
round 6	$([0,1],[1,1],[0,1],[0,1],[1,x_4],[1,x_5],[0,1],[0,1])$
round 7	$([0,1],[0,1],[1,1],[0,1],[0,1],[1,x_4],[1,x_5],[0,1])$
round 8	$([1,x_4x_5],[0,1],[0,1],[1,1],[0,1],[0,1],[0,1],[0,1])$

According to the above division trail with the restriction of T, we know that $\lambda_{1,T}^{(8)} = x_4x_5$. While x_4x_5 does not appear in $s_1^{(8)}$, equivalently to Example 2, we know that this division trail is invalid and should be aborted.

The Specification of the Procedure UpdateFlag

In [9], the authors defined $=, \oplus$ and $\times$ operations for the elements of set $\{0_c, 1_c, \delta\}$. The $=$ operation tests whether two elements are equal (naturally $1_c = 1_c, 0_c = 0_c$ and $\delta = \delta$). The $\oplus$ operation follows the rules:

$$\begin{cases} 1_c \oplus 1_c = 0_c \\ 0_c \oplus x = x \oplus 0_c = x \text{ for arbitrary } x \in \{1_c, 0_c, \delta\} \\ \delta \oplus x = x \oplus \delta = \delta \end{cases} \tag{8}$$

The $\times$ operation follows the rules:

$$\begin{cases} 1_c \times x = x \times 1_c = x \\ 0_c \times x = x \times 0_c = 0_c \text{ for arbitrary } x \in \{1_c, 0_c, \delta\} \\ \delta \times \delta = \delta \end{cases} \tag{9}$$

Based on these rules, we give the algorithm to update the flag vector for the round function of Trivium.

Algorithm 8 Update the Flag Vector

procedure UpdateFlag(*flag*)
  Set $flag_{93} = flag_{66} \oplus flag_{171} \oplus flag_{91} \times flag_{92} \oplus flag_{93}$;
  Set $flag_{177} = flag_{162} \oplus flag_{264} \oplus flag_{175} \times flag_{176} \oplus flag_{177}$;
  Set $flag_{288} = flag_{243} \oplus flag_{69} \oplus flag_{286} \times flag_{287} \oplus flag_{288}$;
  $\boldsymbol{flag} = \boldsymbol{flag} >>> 1$;
  return *flag*;
end procedure

Expressions for $u_1, u_2, \ldots, u_{12}$

Here we give the specific expressions for $u_1, u_2, \ldots, u_{12}$ in Subsections 5.5 and 5.6.

$$u_1 = s^{(312)}_{101} s^{(312)}_{113} s^{(312)}_{114} s^{(312)}_{147} s^{(312)}_{148} s^{(312)}_{155} s^{(312)}_{156} s^{(312)}_{161} s^{(312)}_{175} s^{(312)}_{176} s^{(312)}_{185} s^{(312)}_{186} s^{(312)}_{193} s^{(312)}_{194} s^{(312)}_{201} s^{(312)}_{202}$$
$$u_2 = s^{(312)}_{120} s^{(312)}_{121} s^{(312)}_{149} s^{(312)}_{151} s^{(312)}_{152} s^{(312)}_{163} s^{(312)}_{164} s^{(312)}_{166} s^{(312)}_{165} s^{(312)}_{184} s^{(312)}_{185} s^{(312)}_{195} s^{(312)}_{196} s^{(312)}_{200} s^{(312)}_{210} s^{(312)}_{211} s^{(312)}_{244} s^{(312)}_{245}$$
$$u_3 = s^{(312)}_{102} s^{(312)}_{114} s^{(312)}_{115} s^{(312)}_{148} s^{(312)}_{149} s^{(312)}_{156} s^{(312)}_{157} s^{(312)}_{161} s^{(312)}_{175} s^{(312)}_{176} s^{(312)}_{184} s^{(312)}_{185} s^{(312)}_{192} s^{(312)}_{193} s^{(312)}_{200} s^{(312)}_{201}$$
$$u_4 = s^{(312)}_{102} s^{(312)}_{114} s^{(312)}_{115} s^{(312)}_{148} s^{(312)}_{149} s^{(312)}_{156} s^{(312)}_{157} s^{(312)}_{161} s^{(312)}_{175} s^{(312)}_{176} s^{(312)}_{184} s^{(312)}_{185} s^{(312)}_{192} s^{(312)}_{193} s^{(312)}_{200} s^{(312)}_{201}$$
$$u_5 = s^{(304)}_{87} s^{(304)}_{88} s^{(304)}_{105} s^{(304)}_{106} s^{(304)}_{138} s^{(304)}_{139} s^{(304)}_{145} s^{(304)}_{159} s^{(304)}_{160} s^{(304)}_{166} s^{(304)}_{172} s^{(304)}_{173} s^{(304)}_{174} s^{(304)}_{178} s^{(304)}_{182} s^{(304)}_{183} s^{(304)}_{193}$$
$$u_6 = s^{(314)}_{102} s^{(314)}_{114} s^{(314)}_{115} s^{(314)}_{148} s^{(314)}_{149} s^{(314)}_{155} s^{(314)}_{156} s^{(314)}_{161} s^{(314)}_{175} s^{(314)}_{176} s^{(314)}_{184} s^{(314)}_{185} s^{(314)}_{192} s^{(314)}_{193} s^{(314)}_{201} s^{(314)}_{202}$$
$$u_7 = s^{(305)}_{105} s^{(305)}_{106} s^{(305)}_{138} s^{(305)}_{139} s^{(305)}_{146} s^{(305)}_{147} s^{(305)}_{151} s^{(305)}_{160} s^{(305)}_{165} s^{(305)}_{166} s^{(305)}_{172} s^{(305)}_{173} s^{(305)}_{182} s^{(305)}_{183} s^{(305)}_{190} s^{(305)}_{191} s^{(305)}_{193}$$
$$u_8 = s^{(303)}_{46} s^{(303)}_{106} s^{(303)}_{107} s^{(303)}_{134} s^{(303)}_{135} s^{(303)}_{137} s^{(303)}_{151} s^{(303)}_{152} s^{(303)}_{154} s^{(303)}_{168} s^{(303)}_{169} s^{(303)}_{180} s^{(303)}_{181} s^{(303)}_{185} s^{(303)}_{193} s^{(303)}_{194} s^{(303)}_{229} s^{(303)}_{230}$$
$$u_9 = s^{(309)}_{87} s^{(309)}_{88} s^{(309)}_{101} s^{(309)}_{102} s^{(309)}_{137} s^{(309)}_{138} s^{(309)}_{146} s^{(309)}_{147} s^{(309)}_{152} s^{(309)}_{166} s^{(309)}_{167} s^{(309)}_{171} s^{(309)}_{172} s^{(309)}_{174} s^{(309)}_{183} s^{(309)}_{184} s^{(309)}_{192} s^{(309)}_{193}$$
$$u_{10} = s^{(309)}_{102} s^{(309)}_{103} s^{(309)}_{137} s^{(309)}_{138} s^{(309)}_{146} s^{(309)}_{147} s^{(309)}_{152} s^{(309)}_{158} s^{(309)}_{166} s^{(309)}_{167} s^{(309)}_{172} s^{(309)}_{173} s^{(309)}_{183} s^{(309)}_{184} s^{(309)}_{190} s^{(309)}_{192} s^{(309)}_{193}$$
$$u_{11} = s^{(319)}_{121} s^{(319)}_{122} s^{(319)}_{150} s^{(319)}_{151} s^{(319)}_{152} s^{(319)}_{164} s^{(319)}_{165} s^{(319)}_{166} s^{(319)}_{167} s^{(319)}_{184} s^{(319)}_{185} s^{(319)}_{194} s^{(319)}_{195} s^{(319)}_{201} s^{(319)}_{208} s^{(319)}_{209} s^{(319)}_{243} s^{(319)}_{244}$$
$$u_{12} = s^{(319)}_{121} s^{(319)}_{122} s^{(319)}_{150} s^{(319)}_{151} s^{(319)}_{152} s^{(319)}_{164} s^{(319)}_{165} s^{(319)}_{166} s^{(319)}_{167} s^{(319)}_{184} s^{(319)}_{185} s^{(319)}_{194} s^{(319)}_{195} s^{(319)}_{200} s^{(319)}_{208} s^{(319)}_{209} s^{(319)}_{244} s^{(319)}_{245}$$

The Comparison with the 3-Subset Division Property

Compared with the 2-subset division property, both the 3-subset division property and our method with the distribution of active variables could find more accurate integral characteristics by removing invalid division trails. However, the two methods are quite different and the sets of invalid division trials removed by the two methods are often distinct.

First, the 3-subset division property removes invalid division trails by recording 1-sum property. In the 3-subset division property, division vectors are recorded in K and L where a division vector u in L satisfies 1-sum property, that is, the summation of the bit product $y(X)^u$ is 1 over an input set. During the propagation of 3-subset division property, a division vector appearing even times in L are removed. As a result, the attackers could find more accurate integral characteristics. So far there is no MILP method to compute division trials for 3-subset division property.

Our method removes invalid division trails by finding a term $t = x^u$, which is a product of some active variables involved in a division trail, vanished in the ANF of some intermediate state bit (any multiple of t does not appear). MILP solvers could be easily utilized in our method.

In addition, the 3-subset division property could be used to find 1-sum integral characteristic beside 0-sum integral characteristic, but our method could only be used to find 0-sum integral characteristic.

Finally, it is known that 3-subset division property is only useful for small state ciphers. For stream ciphers like Trivium, it is impossible to compute 3-subset division property because of large internal state.