J Syst Sci Complex (2023) 36: 2234–2254

A New Method for Searching Cubes and Its Application to 815-Round Trivium*

LIU Chen · TIAN Tian · QI Wenfeng

DOI: 10.1007/s11424-023-1497-1
Received: 16 December 2021 / Revised: 24 March 2022

Abstract The cube attack proposed by Dinur and Shamir is one of the most important key-recovery attacks against Trivium. Recently division property based cube attacks have been extensively studied and significantly improved. In particular, the MILP modeling technique for the three-subset division property without unknown subset proposed by Hao, et al. at EUROCRYPT 2020 and the new technique with nested monomial predictions proposed by Hu, et al. at ASIACRYPT 2021 are best techniques to recover exact superpolies in division property based cube attacks. Consequently, at this state of the art, whether a superpoly can be recovered in division property based cube attacks is mainly decided by the scale of the superpoly, that is, the number of terms. Hence the choice for proper cubes corresponding to low-complexity superpolies is more critical now. Some effective cube construction methods were proposed for experimental cube attacks, but not applicable to division property based cube attacks. In this paper, the authors propose a heuristic cube criterion and a cube sieve algorithm, which can be combined with the three-subset division property to recover a number of superpolies. Applied to 815-round Trivium, the authors recovered 417 superpolies from 441 cubes obtained by our algorithm of sizes between 41 and 48. The success rate is 94.56%. There are 165 non-constant superpolies with degree less than 14. In order to demonstrate the significance of the new algorithm, the authors tested the best superpoly recovery technique at EUROCRYPT 2020 using random cubes of similar sizes on 815-round Trivium. The experimental result shows that no cube could be completely recovered within a given period of time because the superpolies for random cubes are too complex.

Keywords Cube attacks, division property, key-recovery attacks, trivium.

1 Introduction

Trivium[1] is one of the eSTREAM hardware-oriented finalists, a bit oriented synchronous stream cipher designed by Cannière and Preneel, and it is also an international standard under ISO/IEC 29192-3:2012. Trivium is composed of a 288-stage quadratic nonlinear feedback shift

LIU Chen · TIAN Tian (Corresponding author) · QI Wenfeng
PLA Strategic Support Force Information Engineering University, Zhengzhou 450001*, China.*
Email: lc_liuchen@126.com; tiantian_d@126.com; wenfeng.qi@263.net.
*This research was supported by the National Natural Science Foundation of China under Grant No. 61672533.
◇ *This paper was recommended for publication by Editor DENG Yingpu.*

register (NFSR) and a linear filtering function. Because of its simple design structure and high level of security, Trivium has attracted extensive attention.

The cube attack is one of the most effective cryptanalytic techniques against Trivium at present, first proposed by Dinur and Shamir at Eurocrypt 2009[2]. The basic idea of a cube attack is as follows. The output bit of a stream cipher can be regarded as a tweakable polynomial $f(\boldsymbol{x}, \boldsymbol{v})$, where $\boldsymbol{x}$ are secret key variables and $\boldsymbol{v}$ are public IV variables. Let I be a subset of IV indices and $t_I = \prod_{i \in I} v_i$. Then, $f(\boldsymbol{x}, \boldsymbol{v})$ can be written as

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

where each term of q_I is not divisible by t_I. In the preprocessing phase, by assigning all possible combinations of 0/1 values to the IV variables indexed by I, $2^{|I|}$ polynomials are obtained from f and their symbolic sum p_I is called the superpoly of I. Consequently, in the online phase, an attacker can build a system of equations on secret key variables $\boldsymbol{x}$ by inquiring the values of all the superpolies obtained in the preprocessing phase. Then some key bits information can be recovered by solving the system of equations. It can be seen that the critical step of cube attacks is to recover a bundle of nonconstant superpolies. In practice, whether a superpoly of a cube can be recovered is decided by two factors. One is the superpoly recovery technique for an iterated tweakable polynomial. The other is the choice of proper cubes. Generally a low-degree superpoly is more easily to be recovered because of sparsity.

Since the cube attack was proposed, a series of improvements made the cube attack play an increasingly important role in the cryptanalysis of stream ciphers, especially in the cryptanalysis of Trivium. According to how to recover superpolies, cube attacks are classified into experimental cube attacks (or traditional cube attacks) and division property based cube attacks. In experimental cube attacks, superpolies are recovered by performing a large number of linearity/quadratic tests, and so their efficiency and effectiveness heavily rely on the sizes of cubes. Here are some benchmarks for experimental cube attacks on Trivium. In [2], where cube attacks were first proposed, there were 35 linear superpolies recovered for 767-round Trivium. In [3], Möbius transformation being introduced to cube attacks with which a set of subcubes can be tested simultaneously, there were 12 linear superpolies and 6 quadratic superpolies recovered for the 799-round Trivium. In [4], an effective method to construct useful cubes for linear superpolies being proposed, there were 42 linear superpolies recovered for 805-round Trivium. The power of the experimental cube attack is mainly restricted by the sizes of cubes which should be within the experimental range, say less than 40. This was overcome by the division property based cube attacks. The bit-based division property was first introduced to cube attacks by Todo, et al. at CRYPTO 2017[5]. With the MILP modeling technique, the division property can exploit superpolies for large cubes and so evaluate the security of a stream cipher against large cubes. In [5, 6], the MILP-aided division property can reveal some key variables not appearing in the superpoly of a given cube for up to 832-round Trivium utilizing a cube of size 72. Later, some technical improvements were given in [7], a constant 0 superpoly was given for 839-round Trivium using a cube of size 78 (The superpoly was later proved to be 0 in [8, 9]). In order to recover exact superpolies in division property based cube attacks, the

authors in [8, 10] tried to build MILP models describing the three-subset division property without unknown subset. Then, a balanced superpoly was recovered for 842-round Trivium with a cube of size 78 in [10]. Then, Hu, et al. also recovered two superpolies for 842-round Trivium with two cubes of size 76 and 77[11]. Very recently, a new framework for recovering the exact ANFs of massive superpolies based on the monomial prediction technique was proposed by Hu, et al. at ASIACRYPT 2021, and the exact ANFs of the superpolies for 843-, 844-, and 845-round Trivium are recovered in [12]. So far 845 is the largest number of rounds for Trivium that a superpoly could be recovered. But the superpolies recovered in [12] are not theoretically balanced. At FSE 2021, Sun proposed a new heuristic algorithm in [13] to search cubes with a balanced secret variable from a preset of candidate cubes. Using the heuristic algorithm, the author recovered a balanced superpoly for 843-round Trivium with a cube of size 78 and presented practical attacks against 806- and 808-round Trivium. Beside the original idea of cube attacks, there are some variants of cube attacks, say correlation cube attacks[14], dynamic cube attacks[15–17], and cube testers[18–23].

Considering the state of the art in cube attacks, both for experimental cube attacks and the division property based cube attacks, there are available techniques for recovering superpolies. In particular, the only reason that a superpoly can not be exactly recovered in the division property based cube attacks is that the algebraic normal form (ANF) of the superpoly is too large. Since randomly selecting cubes is very low efficient, especially as the number of initialization rounds increases, to further improve the effectiveness of cube attacks, constructing proper cubes with low-degree superpolies seems to be critical. Besides, to recover a bundle of superpolies not just one superpoly, cube selecting stage is also important.

Some heuristic methods to construct cubes for key-recovery attacks were proposed in [3, 4, 13, 24]. In [3], the authors proposed a new method to construct a candidate cube by jointing two subcubes satisfying some specific properties. Then, all subcubes of the candidate cube are tested using Möbius transformation to find linear/quadratic superpolies. In particular, in [3], the authors said that the only way linear superpolies have been found for 799-round Trivium was using this method to construct cubes. Thus, it is thought that randomly searching cubes is almost impossible for experimental cube attacks against Trivium with more than 800 initialization rounds. In [24], Ye and Tian put forward a new concept of useful cubes and a method to recover the superpolies of the useful cubes. According to the update function of Trivium, the output function can be represented as a polynomial on the internal state at some time instance t. For a given cube I, by using the numeric mapping method proposed in [22], the algebraic degree of each term can be estimated. If the estimated degree of each term involved in the output function is less than or equal to $|I|$, then the cube I is called a useful cube. In [4], a new heuristic method for constructing cubes used to find linear superpolies is proposed. In this method, the initial cube with a small size is gradually expanded to a large cube according to some greedy strategy. Then, some subcubes of the large cube are tested using Möbius transformation. Later in [13], the author proposed a heuristic algorithm to reject useless cubes from a set of candidate cubes, and the remaining cubes are likely to have balanced superpolies. However, there is no good method in [13] to construct a desirable set of candidate

cubes. For instance, to attack 843-round Trivium, the preset set of candidate cubes is all cubes with dimension 78 and to attack 806- and 808-round Trivium, the preset candidate cubes are from [4].

Up to now the cube construction method proposed in [4] has the highest success rate and was shown to be powerful among experimental cube attacks. However, because it relies on Möbius transformation to test a large cube, the target number of rounds is restricted by an experimental range in [4].

1.1 Our Contributions

In this paper, we devote our attention to the problem of constructing cubes potentially yielding relatively low-degree superpolies compared with random cubes, since the ANF of a low-degree polynomial has small size. We propose a heuristic cube criterion and a new cube search algorithm. First, the new algorithm can construct large cubes whose sizes can be more than 50. Second, the new algorithm will yield a very small set of candidate cubes which can be used in the division property based cube attacks.

The heuristic cube criterion together with the cube sieve algorithm proposed in this paper is mainly inspired by the work in [3]. It is observed that the output function f of Trivium can be recursively expressed as a polynomial with 6 quadratic terms, say $s_{1,1}s_{1,2}, s_{2,1}s_{2,2}, \cdots, s_{6,1}s_{6,2}$, and so the superpoly p_I of I in f is the sum of 6 superpolies $p_{I,1}, p_{I,2}, \cdots, p_{I,6}$ out of these 6 quadratic terms, i.e., $p_I = \bigoplus_{j=1}^{6} p_{I,j}$. Since canceling out terms between $p_{I,1}, p_{I,2}, \cdots, p_{I,6}$ is very unlikely, for a cube I with a low-degree superpoly, it is necessary that for a large number of partitions $\{I_1, I_2\}$ with $I_1 \cup I_2 = I$, the superpolies of I_1 (resp. I_2) in the 12 registers $s_{1,1}, s_{1,2}, s_{2,1}, s_{2,2}, \cdots, s_{6,1}, s_{6,2}$ should all be low-degree. Thus our main idea to sieve cube is dividing a large cube into two disjoint small subcubes and testing the superpolies of two small cubes on 12 related registers. First, in our cube sieve algorithm, this partition process will repeat a sufficient number of times in order to ensure that the final output cubes have the desirable property of low-degree superpolies with a high probability. Second, a new technique based on Möbius transform is proposed to simultaneously test one partition for a number of candidate cubes, that is to say, we simultaneously treat a set of candidate cubes. Third, after experimental testing, the candidate cubes such that the number of disjoint partitions passing through our test exceeds a given threshold are maintained and called valid cubes, which are listed in some order. Valid cubes outputted by our algorithm could be used in the three-subset division property based cube attacks. We remark that though quadratic/linearity tests are used in our algorithm, the size of outputting valid cubes could exceed the experimental range. For instance, by testing subcubes of size 25, we could construct valid cubes of size around 50. Beisdes, before practically recovering the superpolies of valid cubes, their algebraic degrees are undetermined since we only test a part of partitions not all. This implies that there is no strict upperbound on the superpolies recovered from valid cubes, which is good for increasing success probability.

To illustrate the feasibility of the attack strategy, we applied it to 815-round Trivium. In the experiment, we kept 441 valid cubes obtained through our cube sieve algorithm whose

sizes range from 41 to 48 and tried to recover their superpolies using the three-subset division property modeling technique in [10]. We successfully recovered 417 superpolies, among which there are 165 non-constant polynomials. It can be seen that the success rate of valid cubes for recovering superpolies is about 94.56%. Consequently, using these superpolies, we can mount a key-recovery attack against 815-round Trivium by setting up a system of nonlinear equations in key variables. The attack complexity for full-key recovery is about $2^{69}+2^{70}+2^{72}$. At the same time, to verify the effectiveness of this work, we carried out a comparative experiment with random cubes. We used the three-subset division property modeling technique in [10] to recover the superpoly of a random cube. To save time, we set a time limit for each cube, that is, for a cube, if its superpoly can not be restored within 12 hours, the process was stopped and a new cube was randomly selected. As a result, a total of 71 cubes were selected randomly, but none of their superpolies was recovered within the specified time. As a comparison, we summarize the cube attacks based key-recovery attacks against the round-reduced Trivium in Table 1.

Table 1 A summary of key-recovery attacks on Trivium

Attack type	# of rounds	Off-line phase		Total time	ref.
		cube size	# of key bits		
Practical	672	12	63	$2^{18.56}$	[2]
	709	22−23	79	$2^{29.14}$	[25]
	767	28−31	35	$2^{45.00}$	[2]
	784	30−33	42	2^{39}	[3]
	805	32−38	42	$2^{41.40}$	[4]
	806	35−37	45	$2^{39.86}$	[13]
	808	39−41	37	$2^{44.58}$	[13]
Theoretical	799	32−37	18	$2^{62.00}$	[3]
	802	34−37	8	$2^{72.00}$	[26]
	805	28	7	$2^{73.00}$	[14]
	806	34−37	16	2^{64}	[4]
	815	41−48	$10^{\dagger}$	$2^{72.46}$	Subsection 4.2
	835	35	5	$2^{75.00}$	[14]
	832	72	1	$2^{79.01}$	[5, 6, 8]
	832	72	>1	$<2^{79.01}$	[24]
	840	78	1	$2^{79.58}$	[10]
	840	75	3	$2^{77.32}$	[11]
	840	47−62	$3.68^{\ddagger}$	$2^{76.32}$	[12]
	841	78	1	$2^{79.58}$	[10]
	841	76	2	$2^{78.58}$	[11]
	841	56	1^{*}	$2^{78}\star$	[12]
	842	78	1	$2^{79.58}$	[10]
	842	76	2	$2^{78.58}$	[11]

Table 1 (Continued) A summary of key-recovery attacks on Trivium

Attack type	# of rounds	Off-line phase		Total time	ref.
		cube size	# of key bits		
Theoretical	842	56	1*	$2^{78\star}$	[12]
	843	54−57,76	5*	2^{77}	[12]
	843	78	1	$2^{79.58}$	[13]
	844	54−55	2*	2^{78}	[12]
	845	54−55	2*	2^{78}	[12]

† : By using the recovered superpolies, we can extract averagely 10 bits of the key information.
‡ : By using the recovered superpolies in [12], the authors can extract averagely 3.68 bits of the key information.
* : In [12], the recovered superpolies are considered to be balanced by experiments.
⋆ : A superpoly recovered in [12], together with the two superpolies recovered in [11], can obtain 3 key bits, and the total complexity is 2^{78}.

1.2 Organizations

The rest of this paper is organized as follows. In Section 2, we give some basic definitions and concepts. In Section 3, we describe in detail the heuristic cube sieve algorithm. In Section 4, we apply our cube search algorithm to 815-round Trivium and make a comparison with random cubes. Finally, Section 5 concludes this paper.

2 Preliminaries

2.1 Boolean Functions and Algebraic Degree

Let $\mathbb{F}_2$ denote the binary field and $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. The mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$ is called an n-variable Boolean function, and denote by $\mathbb{B}_n$ the set of all n-variable Boolean functions. A Boolean function $f \in \mathbb{B}_n$ can be uniquely represented as a multivariable polynomial over $\mathbb{F}_2$,

$$f(x_1, x_2, \cdots, x_n) = \bigoplus_{\boldsymbol{c}=(c_1,c_2,\cdots,c_n)\in\mathbb{F}_2^n} a_{\boldsymbol{c}} x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n},$$

which is called the algebraic normal form (ANF) of f, and where $a_{\boldsymbol{c}} \in \mathbb{F}_2$. The algebraic degree of f, denoted by $\deg(f)$, is defined as

$$\deg(f) = \max\{\mathrm{wt}(\boldsymbol{c}) | a_{\boldsymbol{c}} \neq 0, \boldsymbol{c} \in \mathbb{F}_2^n\},$$

where $wt(\boldsymbol{c})$ is the Hamming weight of $\boldsymbol{c}$. For ease of description, in this article, we refer to a term with greater degree than 2 as a high-degree term.

2.2 Cube Attacks

Let $x_1, x_2, \cdots, x_n$ be n variables. For a given set of indices $I = \{i_1, i_2, \cdots, i_{|I|}\} \subset \{1, 2, \cdots, n\}$, we can determine a specific term $t_I = x_{i_1} x_{i_2} \cdots x_{i_{|I|}}$, where the variables involved in t_I are called cube variables, and we call I a cube index. Then, according to whether the terms in $f \in \mathbb{B}_n$

are divisible by t_I, f can be uniquely decomposed as

$$f(x_1, x_2, \cdots, x_n) = t_I p_{t_I} \oplus q(x_1, x_2, \cdots, x_n),$$

where each term in $q(x_1, x_2, \cdots, x_n)$ is not divisible by t_I, and p_{t_I} is independent of cube variables. Let C_I, which is called an $|I|$-dimensional cube, be a set of $2^{|I|}$ values where cube variables are taking all possible combinations of values, and all remaining variables (we call noncube variables) are fixed to any value. The sum of f over all values of the cube C_I is

$$\sum_{C_I} f(x_1, x_2, \cdots, x_n) = p_{t_I}.$$

The polynomial p_{t_I} is called the superpoly of C_I in f. For the sake of convenience, the above summation process is called the cube summation, and p_{t_I} is called the superpoly of I in f.

In a cube attack against stream ciphers, the output bits can be regarded as a tweakable Boolean function $f(\boldsymbol{x}, \boldsymbol{v})$ on key variables $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ and the public IV variables $\boldsymbol{v} = (v_1, v_2, \cdots, v_m)$. Because of a number of nonlinear iterations in the initialization stage, the function $f(\boldsymbol{x}, \boldsymbol{v})$ is complicated. The core idea of cube attack is that, by selecting the appropriate cube index I in the offline stage, and performing the cube summation on the output function with noncube variables fixed to some arbitrary values, we can simplify $f(\boldsymbol{x}, \boldsymbol{v})$, obtain the simple expression p_{t_I} on the key variables, and then establish the key equations. The key information is obtained by solving the equations in the online stage.

2.3 The Bit-Based Division Property and Algebraic Degree Evaluation

The Bit-Based Division Property. The concept of division property, a generalization of the integral property, was proposed in [27] at EUROCRYPT 2015. For Sbox-based block ciphers, the authors described the propagation rules of division property for the basic operations such as And, Copy, and Xor. Consequently, according to the specific round function of ciphers, the division property of ciphertext can be figured out by evaluating the propagation of the division property of plaintext. Then, the division property can be used to detect the better integral characteristics for word-oriented cryptographic primitives. Subsequently, at FSE 2016, the bit-based division property was introduced in [28] so that the propagation of integral characteristics can be described in a more precise manner. The definition of the bit-based division property is as follows.

Definition 2.1 (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 $\mathcal{D}_{\mathbb{K}}^{1^n}$, it fulfils the following conditions,

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

where $\boldsymbol{\mu} \succeq \boldsymbol{k}$ if $\mu_i \geq k_i$ for all $i \in \{1, 2, \cdots, n\}$, and $\boldsymbol{x}^{\boldsymbol{\mu}} = \prod_{i=1}^{n} x_i^{\mu_i}$.

Let E_r be an r-round iterative cipher of size n. Assume that $\mathbb{X}$ is the input set with the division property $\mathcal{D}_{\mathbb{K}_0}^{1^n}$. Denote by $\mathbb{Y}$ the corresponding output set created from $\mathbb{X}$ by E_r.

Based on the propagation rules of basic operations proved in [28], the division property of $\mathbb{Y}$, denoted by $\mathcal{D}^{1^n}_{\mathbb{K}_r}$, can be deduced by evaluating the propagation of the division property for every round function. In other words, the division property of $\mathbb{Y}$ can be evaluated as $\mathbb{K}_0 \to \mathbb{K}_1 \to \cdots \mathbb{K}_i \to \cdots \to \mathbb{K}_r$, where $\mathcal{D}^{1^n}_{\mathbb{K}_i}$ is the division property of the internal state after i rounds. However, as r increases, the size of $\mathbb{K}_r$ expands exponentially towards $\mathcal{O}(2^n)$ requiring huge memory resources. Therefore, Xiang, et al. proposed the MILP method and solved the memory crisis in [29].

In order to apply the Mixed Integer Linear Programing (MILP) method to bit-based division property, the author first introduced the concept of division trails at ASIACRYPT 2016, which is defined as follows.

Definition 2.2 (Division Trail) Let us consider the propagation of division property $\mathbb{K}_0 \to \mathbb{K}_1 \to \cdots \mathbb{K}_i \to \cdots \to \mathbb{K}_r$. 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 rule of division property. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r)$ if $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for $i \in \{0, 1, \cdots, r-1\}$, we call $(\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r)$ an r-round division trail.

Xiang, et al. proved that the basic propagation rules Copy, And, Xor of the division property can be translated as some variables and constraints of an MILP model, see [29] for the details. With this method, all possible division trials can be covered with an MILP model $\mathcal{M}$ and the division property of particular output bits can be acquired by analyzing the solutions of $\mathcal{M}$. At CRYPTO 2017, Todo, et al. first applied the bit-based division property to cryptanalysis of stream ciphers, described the propagation rules of division property in stream ciphers, and built the MILP model for stream ciphers[6]. Besides, they proposed the division property based cube attack, which can determine whether a given key variable must not appear in the superpoly of a certain I, by solving the MILP model. Later, Hao, et al. proposed the Flag technique in [7], which can describe the propagation of division property more precisely, to improve the division property based cube attack, with other techniques introduced in [7]. Meanwhile, for a given set of cube variables, the improved method can be used to estimate the degree of the superpoly for Trivium-like ciphers, see [7] for detailed.

The Division Property Based Degree Evaluation. For applying the division property to degree evaluation, the Proposition 4, proposed by Todo et al in paper [5, 6], was generalized in paper [7], and the following proposition was obtained:

Proposition 2.3 *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 $I = \{i_1, i_2, \cdots, i_{|I|}\} \subset \{1, 2, \cdots, m\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1}, v_{i_2}, \cdots, v_{i_{|I|}}\}$ are taking all possible combinations of values. Let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $v^{\boldsymbol{k}_I} = t_I = v_{i_1} v_{i_2} \cdots v_{i_{|I|}}$. Let $\boldsymbol{k}_\Lambda$ be an n-dimensional bit vector. Assuming there is no division trail such that $(\boldsymbol{k}_\Lambda || \boldsymbol{k}_I) \xrightarrow{f} 1$, the monomial $\boldsymbol{x}^{\boldsymbol{k}_\Lambda}$ is not involved in the superpoly of the cube C_I.*

According to Proposition 2.3, the existence of the division trail $(\boldsymbol{k}_\Lambda || \boldsymbol{k}_I) \xrightarrow{f} 1$ is in accordance with the existence of the monomial $\boldsymbol{x}^{\boldsymbol{k}_\Lambda}$ in the superpoly of I. Therefore, the attacker can evaluate the algebraic degree of Trivium-like ciphers by solving the MILP model with the

objective function and some concrete restriction conditions, and Proposition 2.3 guarantees that the estimated degree must be the upper bound of the algebraic degree of the superpoly.

At present, the algebraic degree evaluation based on numeric mapping[22] and division property are the two most effective methods for Trivium-like ciphers. Compared with the method based on numeric mapping, the method based on division property is more accurate, but takes more time. As the initialization rounds r of Trivium-like ciphers increases, it becomes more difficult to return the upper bound of the degree of the superpoly.

2.4 A Method to Recover Superpoly Based on the Milp-Aided Division Property

The cube attack based on division property[6, 7] just tells us whether a given key variable or a given monomial is definitely not in the superpoly. However, it cannot guarantee that the superpoly is a non-constant polynomial, when it tells us that there is a certain monomial involved in the superpoly. For this weakness, Ye and Tian have improved it in [9], and then proposed a new method to recover the superpoly based on the MILP-aided division property. Here, two important lemmas in [9] are simply described.

Lemma 2.4 *For a set of indices $I \subset \{1, 2, \cdots, n\}$, let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $\boldsymbol{v}^{\boldsymbol{k}_I} = \prod_{i \in I} v_i$. Assume that $u = (s^{(t)})^{\boldsymbol{\omega}_u}$, where $s^{(t)} = (s_1^{(t)}, s_2^{(t)}, \cdots, s_N^{(t)})$ is an N-dimensional vector made up of the internal state bits of the target cipher after t rounds of iteration, and $\boldsymbol{\omega}_u = (\omega_u^1, \omega_u^2, \cdots, \omega_u^N) \in \mathbb{F}_2^n$. Namely, u is a term which is the product of some internal state bits of $s^{(t)}$. If there is no division trail such that $(\boldsymbol{0}, \boldsymbol{k}_I) \xrightarrow{s^{(t)}} \boldsymbol{\omega} = (\omega_1, \omega_2, \cdots, \omega_N)$ for each $\boldsymbol{\omega} \preceq \boldsymbol{\omega}_u = (\omega_u^1, \omega_u^2, \cdots, \omega_u^N)$, then the superpoly of I in u is 0-constant, where $\boldsymbol{\omega} \preceq \boldsymbol{\omega}_u$ means that $\omega_i \leq \omega_u^i$ for $1 \leq i \leq N$, and in this case, the term u is called an invalid term.*

Lemma 2.5 *Let I be a set of cube indices. Assume that the output bit f is presented as a polynomial in $s^{(t)}$, i.e., $f = g_t(s^{(t)})$. Then, according to Lemma 2.4, $g_t(s^{(t)})$ could be rewritten as $g_t(s^{(t)}) = g_t^1(s^{(t)}) \oplus g_t^2(s^{(t)})$, where each term u involved in $g_t^2(s^{(t)})$ is an invalid term for I. It can be seen that, the superpoly of I in $f = g_t(s^{(t)})$ is exactly the superpoly of I in $g_t^1(s^{(t)})$.*

According to the above two lemmas, the authors can recover the superpoly of a given cube indexed by the set I. Firstly, the output function f can be represented as a polynomial on the internal state $s^{(t)}$, i.e., $f = g_t(s^{(t)})$. Then, every term u involved in $g_t(s^{(t)})$ is checked with the MILP-aided division property, and the invalid terms need to be discarded. As a result, the authors could obtain a simplified polynomial $g_t^1(s^{(t)})$, and on the basis of Lemma 2.5, the superpoly of I in $g_t^1(s^{(t)})$ equals to the superpoly in $g_t(s^{(t)})$. Next, they further express $g_t^1(s^{(t)})$ as a polynomial on the internal state $s^{(t-t_1)}$, and obtain the simplified polynomial $g_{t-t_1}^1(s^{(t-t_1)})$ by repeating the above process. By performing the above procedure iteratively, they can finally obtain a polynomial $g_0^1(s^{(0)})$ such that the superpoly of I in $g_0^1(s^{(0)})$ is exactly the superpoly in f. Therefore, the superpoly p_{t_I} can be recovered easily according to how $s^{(0)}$ is initialized.

2.5 A Method to Construct Potentially Good Cubes

In [4], for Trivium-like ciphers, combining the idea of GreedyBitSet algorithm with division property[18], Ye and Tian proposed a new algorithm to construct cubes which may have linear

superpoly, through extending a starting cube set iteratively with steep IV variables and gentle IV variables defined by the authors. The algorithm is divided into two stages.

Stage 1 Determine starting cube sets.

By analyzing the concrete structure of Trivium, the authors introduced the concept of the preference bit, which is formally described in the following definitions.

Definition 2.6 (The Preference Bit) Among the six internal state bits in output function of r-round Trivium, the internal state bit which is the most likely to contribute linear superpolies is called the preference bit of r-round Trivium.

Then, the algorithm to predict the preference bit of r-round Trivium was proposed in [4]. For constructing the cubes possibly with linear superpoly, the authors expressed the preference bits of r-round Trivium as a quadratic polynomial with only one quadratic term by using the update function. If there is a cube yielding a linear superpoly on at least one of the state bits involved in the quadratic term, then it is set as a starting cube.

Stage 2 Expand the starting cube with steep IV variables and gentle IV variables.

The authors extended the obtained initial cube index I' by adopting a greedy strategy. In the greedy strategy, the criterion for each selection of IV variables is the estimated degree of the superpoly, and in different cases, they chose one of the steep IV variable and the gentle IV variable, which are defined in [4]. Firstly, the evaluated degree of the superpoly is reduced rapidly by iteratively adding a steep IV variable to I'. They want to construct the cubes which have linear superpoly, therefore, if the degree estimation result is linear, no variables are added, and the cube at this time is set to a candidate cube. If the superpoly becomes a 0-constant polynomial after adding a steep IV variable, then the last added variable will be taken out of I', and the gentle IV variables will be set to the cube variables iteratively, to control the degree of the superpoly to approach the linearity slowly.

For 805-round Trivium, a number of candidate cubes were constructed by using this method, and then for each candidate cubes, the authors detected all subcubes simultaneously by using Möbius transformation, and all of candidate cubes could find a subcube with linear superpoly. As a consequence, we think that, this method is able to construct candidate cubes, including some subcubes which possibly have the linear superpoly with a high probability.

3 A Heuristic Algorithm to Sieve Cubes

With the increase of the number of initialization rounds of Trivium, the dimension of cubes will also increase. To efficiently search cubes with low-degree superpolies, we propose a heuristic cube criterion. Then, we put forward a cube sieve algorithm based on this criterion, which can get valid cubes by sieving all the subcubes of the given cube I, and for convenience, we call such the cube I a target cube in the following.

We will first briefly describe Trivium in Subsection 3.1. Then introduce the heuristic cube criterion in Subsection 3.2. Finally the cube sieve algorithm for Trivium is given in Subsection 3.3.

3.1 Description of Trivium

The main building block of Trivium is a Galois nonlinear feedback shift register. At each clock cycle, there are three bits of the internal state updated by different quadratic feedback functions respectively, and the others are updated by shifting. Trivium is divided into three registers, a total of 288 bits of the internal state. In the initialization phase, an 80-bit secret key and an 80-bit IV are loaded in the internal state of Trivium. After updating the internal state iteratively for 1152 rounds, Trivium starts to output keystream bits. We show the pseudo-code of Trivium in Algorithm 1.

Algorithm 1: TRIVIUM

Input: $\boldsymbol{k} = (k_0, k_1, \cdots, k_{79})$, $\boldsymbol{v} = (v_0, v_1, \cdots, v_{79})$, N
Output: $\boldsymbol{z} = (z_1, z_2, \cdots, z_N)$
$(s_1, s_2, \cdots, s_{93}) = (k_0, k_1, \cdots, k_{79}, 0, \cdots, 0)$
$(s_{94}, s_{95}, \cdots, s_{177}) = (v_0, v_1, \cdots, v_{79}, 0, \cdots, 0)$
$(s_{178}, s_{179}, \cdots, s_{288}) = (0, 0, \cdots, 0, 1, 1, 1)$
for i ***from*** 1 ***to*** $1152 + N$ **do**
    if $i > 1152$ **then**
        Output: $z_{i-1152} = s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288}$
    $t_1 = s_{66} \oplus s_{93} \oplus s_{91}s_{92} \oplus s_{171}$
    $t_2 = s_{162} \oplus s_{177} \oplus s_{175}s_{176} \oplus s_{264}$
    $t_3 = s_{243} \oplus s_{288} \oplus s_{286}s_{287} \oplus s_{69}$
    $(s_1, s_2, \cdots, s_{93}) = (t_3, s_1, \cdots, s_{92})$
    $(s_{94}, s_{95}, \cdots, s_{177}) = (t_1, s_{94}, \cdots, s_{176})$
    $(s_{178}, s_{179}, \cdots, s_{288}) = (t_2, s_{178}, \cdots, s_{287})$

3.2 A Heuristic Cube Criterion

The output function of Trivium is given by $f = s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288}$, which involves 6 internal state bits at the r-th round. According to the update function $t_l = s_i s_{i+1} \oplus s_{i+2} \oplus s_k \oplus s_j$ in Trivium, each bit s_i of the internal state can be recursively expressed as a polynomial with a single degree 2 monomial. For the sake of convenience, denote $s_{1,1}s_{1,2}, s_{2,1}s_{2,2}, \cdots, s_{6,1}s_{6,2}$ by the 6 quadratic terms contained in the output function, and there are 12 bits of the internal state involved in the high-degree part of the polynomial expression of the output bit.

The cube C_I can be determined by the index set I, for convenience of description and without ambiguity, we also call the set I a cube in the following. For a given cube I, we assume that the degree of the term t_I determined by I is greater than the terms outside the high-degree part of the output function f. In this case, the superpoly of I in f is exactly the superpoly in the high-degree part of f, namely, the sum of the superpolies in the 6 quadratic terms contained in f. Let $\{I', I''\} \in Part_I$ be a partition of I, which means that I', I'' satisfy the condition

$I' \cup I'' = I$, where $Part_I$ is the set including all the different partitions of I. Based on the above assumptions, the superpoly of I satisfies the following property.

Property 3.1 For a given cube I, if the degree of the term $t_I = \prod_{i \in I} v_i$ is greater than the terms outside the high-degree part of the output function f, then the superpoly p_{t_I} of I in f satisfies the following equation,

$$p_{t_I} = \bigoplus_{\{I',I''\} \in Part_I} \bigoplus_{i=1}^{6} p'_{i,1} p''_{i,2}, \tag{1}$$

where $p'_{i,j}$ and $p''_{i,j}$ are the superpoly of I' and I'' in $s_{i,j}$ ($i \in \{1, 2, \cdots, 6\}, j \in \{1, 2\}$).

Proof The bit s_i involved in f can be regarded as a polynomial in cube variables, and the ANF of s_i is expressed as

$$s_i = \bigoplus_{I' \subset I} p'_i t_{I'},$$

where p'_i is the coefficient of the ANF, and $t_{I'} = \prod_{i \in I'} v_i$. If $I' = \emptyset$, then $t_{I'} = 1$. Therefore, the quadratic term $s_{i,1} s_{i,2}$ can be represented as

$$s_{i,1} s_{i,2} = \bigoplus_{I', I'' \subset I} p'_{i,1} p''_{i,2} t_{I'} t_{I''}.$$

Evidently, the superpoly of I in $s_{i,1} s_{i,2}$ satisfies

$$p_i = \bigoplus_{\{I',I''\} \in Part_I} p'_{i,1} p''_{i,2}.$$

Since the superpoly p_{t_I} of I in f is the sum of the superpolies in the six quadratic terms, Equation (1) holds. ∎

According to Property 3.1, we present a new detection method called partition test.

Definition 3.2 (Partition Test) For a given cube I, partition I into disjoint two subcubes $\{I', I''\}$. Then quadratic/linearity/constant tests are performed on the 12 bits $s_{i,j}$ ($i \in \{1, 2, \cdots, 6\}, j \in \{1, 2\}$) involved in the high-degree part of the output function f by using subcubes I' and I'' respectively. If for any $l \in \{1, 2, \cdots, 6\}$, there are $\deg_{I'}(p'_{l,1}) + \deg_{I''}(p''_{l,2}) \leq 2$ and $\deg_{I''}(p''_{l,1}) + \deg_{I'}(p'_{l,2}) \leq 2$, then we call the cube I passes one single partition test, where $\deg_{I'}(p'_{i,j})$ denotes the degree of the superpoly $p'_{i,j}$ for I' in $s_{i,j}$ for $i \in \{1, 2, \cdots, 6\}, j \in \{1, 2\}$.

The idea of this partition test is estimating the degree of the polynomial $\bigoplus_{i=1}^{6} p'_{i,1} p''_{i,2}$ by using different partitions $\{I', I''\}$ of I. In order to use partition test for a practical attack, we assume that as the number of initialization rounds r of Trivium increases, in Equation (1), it happens less and less that the maximal degree monomials involved in the polynomial $\bigoplus_{i=1}^{6} p'_{i,1} p''_{i,2}$ will cancel each other in the sum. So, if the given cube I does not pass the single partition test, it does not mean that the superpoly of I must have high-degree terms. However, if the cube fails in many partition tests, we believe that, there are plenty of terms with high-degree in the superpoly, and the superpoly is difficult to be recovered. Based on the above point, we give the following criterion.

Heuristic Cube Criterion. For a given cube I, we perform N different partition tests on it. If the more times the partition test for I fails, the more high-degree terms the superpoly of I involves, then the more difficult it is to be recovered the ANF of the superpoly; In contrast, if the more times the partition test for I is passed, it is considered that the superpoly of I contains fewer high-degree terms, and the easier it is to be recovered.

According to this criterion, we can quickly judge whether the superpoly of I in output function f is easy to be recovered or not, by multiple partition tests for a given cube I. Moreover, since we test the cube I by using its subcubes after split, the number of the cipher operations required by the test is only related to the dimension of the subcubes. In the next subsection, we will give a heuristic cube sieve algorithm.

3.3 A Heuristic Cube Sieve Algorithm

The goal of the algorithm is to efficiently filter out the valid subcubes from the given target cube I. We divide the actual sieving process into three stages.

The primary goal of the first phase is to produce partitions that meets the requirements. We partition I differently for N times, and obtain a set

$$Part_I \triangleq \{\{I'_i, I''_i\} | I'_i \cup I''_i = I \text{ and } I'_i \cap I''_i = \emptyset \text{ and } \big||I'_i| - |I''_i|\big| \leq 1, i \in \{1, 2, \cdots, N\}\}.$$

It can be seen that, we have put forward several requirements for the partitions, and we will explain the reasons for these conditions at the end of this section.

In the second phase, we perform the partition tests for I by using the partitions $\{I', I''\} \in Part_I$. In order to improve detection efficiency, we introduce Möbius transformation into the partition tests. In the single partition test, we first conduct experimental tests on the 12 bits $s_{i,j}$ $(i \in \{1, 2, \cdots, 6\}, j \in \{1, 2\})$ involved in the high-degree part of f by using all subcubes of I' at the same time, and record the degree evaluations of the superpolies of the subcubes with the dimension greater than or equal to $|I'| - m$. For I'', in the same way, we record the degree evaluations of the superpolies of the subcubes with the dimension greater than or equal to $|I''| - m$, where the parameter m determines the minimum dimension of the cube which we need. We denote the result of the test as $\deg_{I'}(p_{k,j}) \in \{-9, 0, 1, 2, 3\}$, where these values in the set represent that the polynomials $p_{k,j}$ are a 0-constant polynomial, a 1-constant polynomial, a linear polynomial, a quadratic polynomial and a polynomial with degree greater than 2. Next we retrieve the results recorded, if there are $I'_j \subset I'_i$ and $I''_l \subset I''_i$ that satisfy the following conditions,

$$\deg_{I'_j}(p'_{k,1}) + \deg_{I''_l}(p''_{k,2}) \leq 2 \quad \text{and} \quad \deg_{I''_j}(p''_{k,1}) + \deg_{I'_l}(p'_{k,2}) \leq 2, \quad \text{for any } k \in \{1, 2, \cdots, 6\}.$$

Then, it indicates that the subcube $I_{j,l} \triangleq I'_j \cup I''_l$ has passed the single partition test, and $I_{j,l}$ is called candidate cube. Next, we establish the set Φ, which contains all candidate cubes which pass a single test,

$$\begin{aligned}\Phi = \{I_{j,l} \triangleq I'_j \cup I''_l \,|\, &\deg_{I'_j}(p'_{k,1}) + \deg_{I''_l}(p''_{k,2}) \leq 2 \text{ and} \\ &\deg_{I''_l}(p''_{k,1}) + \deg_{I'_j}(p'_{k,2}) \leq 2, \text{ for any } k \in \{1, 2, \cdots, 6\}\}.\end{aligned}$$

By reason of above requirements for the cube I when it is split, we can guarantee in a single partition test, for each subcubes with dimension greater than or equal to $|I| - m$, we conduct one and only one partition test on it. Therefore, it is impossible that the same subcube appears multiple times in the Φ. After N partition tests, N sets Φ_i ($i \in \{1, 2, \cdots, N\}$) can be obtained. Next, we construct the multiset $MultiCube = \bigcup_{i=1}^{N} \Phi_i$, count the candidate cube in $MultiCube$, and create a table T_I of candidate cubes based on the order of count from largest to smallest.

At the last stage, the candidate cubes will be filtered according to the preset threshold K. According to the heuristic cube criterion proposed in Subsection 3.2, we believe that the more times the candidate cube passes the test, the easier it is to recover its superpoly. So, in the practice, we will sieve the valid cubes from the candidate cubes by the preset threshold value K. As we know, each subcube of I with degree greater than or equal to $|I| - m$ has been tested N times, and in the table T_I, the count of candidate cubes indicates the number of times that have passed partition tests. If the count of a subcube is less than $N - K$, we think it is difficult to recover its superpoly, and discard it. On the contrary, we save the subcube, and define the priority in order of counting from the largest to the smallest. Finally, we can obtain a collection including valid cubes with a priority.

Now explain why the above requirements are set for the cube split. By observing the update function of Trivium, it can be seen that the quadratic term $s_{i,1}s_{i,2}$ is the product of adjacent state bits. When $s_{i,1}$ and $s_{i,2}$ are regarded as polynomials in key variables and IV variables, they are similar in complexity. So, for a partition $\{I', I''\}$ of I, the dimension difference of I' and I'' is required to be at most 1, in order to avoid 0-constant polynomial in the superpolies $p'_{i,1}$ of I' and $p''_{i,2}$ of I'' in $s_{i,j}$ ($i \in \{1, 2, \cdots, 6\}, j \in \{1, 2\}$) to make $\bigoplus_{i=1}^{6} p'_{i,1}p''_{i,2}$ and $\bigoplus_{i=1}^{6} p''_{i,1}p'_{i,2}$ 0-constant polynomials. Because in a single partition test, if $|I'|$ is much bigger than $|I''|$, then the superpolies of I' in $s_{i,j}$ ($i \in \{1, 2, \cdots, 6\}, j \in \{1, 2\}$) are more likely 0-constant polynomials, and such test is meaningless. The second requirement is I' and I'' are disjoint, which can guarantee that, in a single partition test, if the subsets $I'_j, I'_l \subset I'$ and $I''_h, I''_k \subset I''$ satisfy $I'_j \neq I'_l$ or $I''_h \neq I''_k$, then $I'_j \cup I''_h \neq I'_l \cup I''_k$. Namely, in a single partition test, all detected subcubes are different. Meanwhile, we only record the detection results of the subcubes of I' and I'' with dimension respectively greater than or equal to $|I'| - m$ and $|I''| - m$. So, in a single partition test, we can guarantee all the subcubes with dimension greater than or equal to $|I| - m$ has one and only one partition test. It can be seen that, a single partition test needs $2^{|I'|} + 2^{|I''|}$ cipher operations and $2^{\max\{|I'|,|I''|\}}$ bits memory, in consequence, these two requirements can reduce computational complexity and storage complexity.

3.4 A Simple Instance

In order to make the algorithm easier to understand, we provide a simple instance to illustrate the specific implementation process. Given the output polynomial $h = f \cdot g$ which is a

polynomial on $\boldsymbol{key} = (k_1, k_2, \cdots, k_6)$ and $\boldsymbol{v} = (v_1, v_2, \cdots, v_7)$, where there are

$$\begin{aligned} f(\boldsymbol{key}, \boldsymbol{v}) &= k_1k_3k_6v_1v_2v_4v_7 + k_1v_1v_3v_4 + k_6v_1v_3v_7 + k_1k_2v_1v_4v_7 \\ &\quad + k_5v_1v_5v_7 + k_5v_2v_4v_5 + k_2k_4v_2v_5v_7 + k_2k_3v_4v_5v_7 + k_1v_1v_2 \\ &\quad + k_2k_5v_2v_4 + k_3k_4v_2v_6 + k_1v_3v_5 + k_3v_2 + k_4v_3 + k_5v_6, \\ g(\boldsymbol{key}, \boldsymbol{v}) &= k_2k_4k_5v_3v_5v_6v_7 + k_3k_4v_1v_2v_7 + k_6v_3v_4v_5 + v_1v_3 \\ &\quad + k_1k_6v_1v_4 + k_6v_2v_3 + k_3v_2v_5 + k_3k_5v_2v_7 + k_5k_6v_3v_4 \\ &\quad + k_1k_5v_4v_6 + k_1v_3 + k_4v_5 + k_6v_7. \end{aligned}$$

For the given cube $I = \{1, 2, 3, 4, 5, 7\}$, we use the sieve algorithm to find the subcubes with the low-degree and sparse superpolies. For simplicity, we only consider the subcube whose dimension is greater than or equal to 5, that is, $m = 1$.

Firstly, we will divide I into two 3-dimensional disjoint subcubes, I' and I''. Since the size of I is small, we will detect all the partitions that meet the conditions, a total of 10 different partitions, that is, $N = 10$.

Secondly, we test all the subcubes with dimension greater than 4 by using different partitions. For simplicity, we take $I' = \{1, 3, 4\}, I'' = \{2, 5, 7\}$ as an example to illustrate. Firstly, f and g are detected by using I' and I'', and Table 2 can be obtained. Then search the table to find subcubes meeting both conditions $\deg_{I'_i}(f) + \deg_{I''_i}(g) \leq 2$ and $\deg_{I''_i}(f) + \deg_{I'_i}(g) \leq 2$. Meanwhile, all the subcubes satisfying conditions are

$$\Phi = \{\{1, 2, 3, 4, 5, 7\}, \{1, 2, 3, 4, 5\}, \{1, 3, 4, 5, 7\}, \{1, 2, 3, 5, 7\}\}.$$

Table 2 Detection result by using I' and I''

subcube index I'	$\deg_{I'_i}(f)$	$\deg_{I'_i}(g)$	subcube index I''	$\deg_{I''_i}(f)$	$\deg_{I''_i}(g)$
$\{1, 3, 4\}$	1	-9	$\{2, 5, 7\}$	2	-9
$\{1, 3\}$	-9	0	$\{2, 5\}$	-9	1
$\{1, 4\}$	-9	2	$\{2, 7\}$	-9	2
$\{3, 4\}$	-9	2	$\{5, 7\}$	-9	-9

Finally, after 10 tests, we count the number of different elements in the multiset $MultiCube$, and we show the result in Table 3.

Table 3 Statistical results for elements in the multiset

subcube index	count
$\{1, 2, 3, 4, 5, 7\}$	10
$\{1, 2, 3, 4, 5\}$	10
$\{1, 2, 3, 4, 7\}$	8
$\{1, 2, 3, 5, 7\}$	9
$\{1, 2, 4, 5, 7\}$	8
$\{1, 3, 4, 5, 7\}$	9
$\{2, 3, 4, 5, 7\}$	8

Therefore, it is considered that $\{1,2,3,4,5,7\}$ and $\{1,2,3,4,5\}$ have the highest priority, and in the stage of recovering superpolies, we use them preferentially. It can be seen that $\{1,2,3,5,7\}$ and $\{1,3,4,5,7\}$ also have high priority. Judging from the test results, the superpolies of $\{1,2,3,4,7\}$, $\{1,2,4,5,7\}$ and $\{2,3,4,5,7\}$ are the most difficult to be recovered. Then, we give specific superpolies of all the subcubes, to verify whether this method is valid. The superpolies of all subcubes are shown in Table 4. It can be seen that the complexity of the actual superpolies is the same as the estimate given by the algorithm, indicating that the sieve algorithm is effective.

Table 4 Specific superpolies of I'

subcube index	Superpoly
$\{1,2,3,4,5,7\}$	$k_1k_3k_6$
$\{1,2,3,4,5\}$	$k_1k_3+k_1k_6+k_5$
$\{1,2,3,4,7\}$	$k_1k_3k_5k_6+k_1k_2k_6+k_1k_3k_4+k_1k_3k_5+k_1k_3k_6$
$\{1,2,3,5,7\}$	$k_1k_3k_4+k_2k_4+k_3k_6+k_5k_6$
$\{1,2,4,5,7\}$	$k_1k_2k_4k_6+k_1k_3k_4k_6+k_1k_2k_3+k_1k_3k_6+k_2k_3k_4+k_3k_4k_5$
$\{1,3,4,5,7\}$	$k_1k_2k_6+k_2k_3+k_6$
$\{2,3,4,5,7\}$	$k_2k_4k_5k_6+k_2k_3k_6+k_2k_4k_6$

4 Applications to Trivium

In this section, based on the sieve algorithm given in Subsection 3.3, we propose a new attack strategy, which can efficiently search cubes with low-degree and sparse superpolies. We describe the new strategy in Subsection 4.1, and then present the actual results for 815-round Trivium in Subsection 4.2. Meanwhile, in order to show that the strategy is meaningful, we provided a comparative experiment in Subsection 4.3.

To make the results verifiable, we submitted the codes and the all experimental results to GitHub[†].

4.1 A New Attack Strategy

We divided the whole strategy into three parts, namely, constructing a target cube, sieving valid cubes, and recovering the superpolies according to their priority.

In the phase of constructing a target cube, we adopt the method in paper [4]. It can be seen that, the cube sieve algorithm proposed in this paper is to detect a batch of subcubes of the target cube, and return the subcubes which pass the detection. However, this algorithm is probabilistic, it is more necessary to find the target cube with valid subcubes with a higher probability, so as to ensure that the subcubes we obtained can be used to recover superpolies successfully. For this reason, we chose this method proposed in [4]. Next, we will filter a batch of subcubes of the target cube with the cube sieve algorithm, and obtain some valid subcubes with priority. Finally, we use the method introduced by Ye and Tian to recover the superpolies of the subcubes we filtered, because the superpolies we want to recover are not limited to linear

[†]https://github.com/LiuChen522699/A-new-method-for-searching-cubes-and-its-application-to-815-round-Trivium.git.

and quadratic, and the method of experimental tests is not feasible. In addition, in the cube sieve algorithm, whether the superpolies are easy to be recovered or not, is judged according to the frequency of the occurrence of the high-degree terms, and the valid cubes have fewer such terms. Coincidentally, the method proposed by Ye et al is easier to recover these cubes.

To demonstrate the effectiveness of this strategy, we perform an actual attack on 815-round Trivium.

4.2 Experimental Results On 815-Round Trivium

In this section, we apply the new strategy to 815-round Trivium and give the experimental results.

In the stage of constructing the target cubes, firstly, we found some starting cubes whose dimension are 16 by using the preference bit. Now, the 5 starting cubes used in this paper are shown in Table 5.

Table 5 Starting cubes

$I_1 = \{0, 1, 2, 3, 5, 12, 19, 20, 33, 38, 42, 57, 58, 59, 63, 77\}$
$I_2 = \{1, 4, 5, 6, 13, 23, 30, 36, 39, 41, 46, 47, 65, 67, 73, 75\}$
$I_3 = \{2, 3, 10, 13, 20, 24, 30, 31, 38, 45, 52, 61, 63, 65, 73, 74\}$
$I_4 = \{1, 11, 13, 19, 32, 33, 35, 36, 38, 51, 55, 57, 62, 64, 72, 78\}$
$I_5 = \{5, 6, 7, 10, 11, 18, 26, 27, 29, 32, 34, 36, 48, 67, 71, 75\}$

Then, by using the GreedyBitSet algorithm, the starting cubes were expanded by adding steep IV variables. In the next stage, instead of adding gentle IV variables, we chose appropriate IV variables to construct the target cubes according to the estimated algebraic degree of superpolies of the cubes added different non-cube IV variables, as shown in Table 6.

Table 6 Cubes extended with steep IV variables, and the degree evaluation with further extension

cubes extended with steep IV variables	further extension	degree evaluation
$I'_1 = \{0, 1, 2, 3, 4, 5, 6, 8, 10, 11, 12, 14, 15, 16,$	$I'_1 \cup \{28\}$	3
$17, 18, 19, 20, 23, 25, 27, 29, 31, 33, 36, 38, 40, 42,$	$I'_1 \cup \{49\}$	1
$44, 46, 51, 53, 57, 58, 59, 60, 63, 66, 67, 70, 77, 79\}$	$I'_1 \cup \{34\}$	1
$I'_2 = \{0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 13, 15, 16,$	$I'_2 \cup \{38\}$	2
$17, 18, 19, 21, 23, 25, 27, 29, 30, 31, 32, 33,$	$I'_2 \cup \{62\}$	2
$36, 39, 41, 46, 47, 50, 52, 54, 56, 58, 60, 61,$	$I'_2 \cup \{63\}$	2
$65, 67, 69, 71, 73, 75, 79\}$	$I'_2 \cup \{74\}$	0
$I'_3 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 15, 16, 17,$ $18, 20, 22, 24, 26, 28, 30, 31, 32, 35, 37, 38, 41, 42,$ $45, 48, 50, 52, 54, 61, 63, 65, 67, 70, 73, 74, 78\}$	$I'_3 \cup \{33, 39\}$	2
$I'_4 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 16, 17,$ $18, 19, 22, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35,$ $36, 38, 40, 42, 43, 46, 51, 53, 55, 57, 60, 62, 64, 66,$ $68, 70, 72, 78\}$	$I'_4 \cup \{61\}$	1
$I'_5 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16,$	$I'_5 \cup \{60\}$	1
$17, 18, 1921, 23, 25, 26, 27, 29, 30, 32, 34, 36, 38,$ $40, 46, 48, 50, 53, 56, 58, 63, 65, 67, 69, 71, 75, 78\}$	$I'_5 \cup \{73\}$	1

Finally, we constructed the following target cubes and predefine the dimension of the subcubes to be detected, as shown in Table 7.

Table 7 Target cubes

Target cubes	dim	Parameter m
$I_6 = I'_1 \cup \{34, 49\}$	44	3
$I_7 = I'_1 \cup \{28, 34\}$	44	3
$I_8 = I'_1 \cup \{28, 49\}$	44	3
$I_9 = I'_2 \cup \{38, 62\}$	46	4
$I_{10} = I'_2 \cup \{63, 74\}$	46	4
$I_{11} = I'_3 \cup \{33, 39\}$	44	3
$I_{12} = I'_4 \cup \{61\}$	48	3
$I_{13} = I'_5 \cup \{60, 73\}$	46	4

In the stage of filtering subcubes, $N = 1200$ and $K = 80$ were preset, that is, subcubes which meet the dimension requirement of the target cubes are performed 1200 different partition tests, and the subcubes which pass less than 1120 are discarded. However, in actual experiment, we only recovered the superpolies of the part of subcubes with higher priority, but not all the subcubes that passed the test. For I_5, I_6, and I_7, we preset the parameter $m = 3$, that is, detect all subcubes whose dimension are greater than or equal to 41. Then, a total of 107 subcubes passed the test were selected from I_6, I_7, I_8, and recovered a total of 34 non-constant polynomials. For I_9 and I_{10}, we detected all subcubes whose dimension are greater than or equal to 42, and used the first 186 subcubes with higher priority to recover superpolies, then we obtained 88 non-constant polynomials. For I_{11}, we used the first 35 subcubes with the dimension greater than 41 to recover their superpolies, and 9 non-constant polynomials were obtained. For I_{12}, we used the first 104 subcubes which have the dimension greater than 44 to restore the superpolies, and we got 30 non-constant polynomials. For I_{13}, we used the first 9 subcubes which have the dimension greater than 43 to restore the superpolies, and we got 4 non-constant polynomials. Most of the other polynomials are 0-constant polynomials. Among the detected subcubes, only a few had the superpolies which are difficult to be recovered.

Next, we used the obtained polynomials to establish nonlinear equations. In order to effectively use the recovered superpolies, we simplified the equations by assigning a few key bits. First, when $k_{58} = 1$, we could use 11 balanced polynomials to establish equations, and through analysis, the solution space of the equations was 2^{69}; When $k_{58} = 0$ and $k_{56} = 1$, we could use 10 balanced polynomials to establish equations, and its solution space was 2^{70}; When $k_{58} = 0$ and $k_{56} = 0$, we could use 8 balanced polynomials to establish equations, and the solution space was 2^{72}.

As a consequence, we can restore all the key-bit information, by calling $2^{69} + 2^{70} + 2^{72}$ cipher operations.

4.3 A Comparison with Random Cubes

Recall that Hao, et al. proposed a modeling technique for the three-subset division property in [10], which is one of the best superpoly recovery method using MILP-aided division property. It can be seen that cubes exploited in [10] are of size 78 which is very large. In the following experiment, it is shown that the superpoly recovery method in [10] based on random cubes of moderate sizes did not work on 815-round Trivium.

In this experiment, we randomly selected cubes with the same dimension as the cubes in Subsection 4.2 and used the three-subset division property method in [10] to recover superpolies. Because it depended on MILP solvers, the specific time complexity could not be estimated. Generally, a superpoly had more terms, the MILP model is solved more slowly. Therefore, we required that the time to restore a single superpoly should not exceed 12 hours, so as to avoid spending a large amount of time on extremely complex polynomials. We used a PC with Intel Core i7-8700 @ 3.20GHz CPU. In a month, we tested 71 random cubes, but no superpoly was recovered. Hence, though the division property based cube attacks could recover superpolies for large cubes, selecting cubes with sparse superpolies is also necessary.

5 Conclusions

In this paper, we propose a new method to construct cubes with low-degree superpolies. The core idea is to estimate the frequency of high-degree terms in a superpoly through repeatedly partitioning a large cube into two small disjoint subcubes, so as to judge whether the superpoly is easy to be recovered. Although the partition tests are basically experimental, the finally constructed cube is large. In particular, the new method could be used in the three-subset division property based cube attacks. We applied it to 815-round Trivium for instance on a PC. As a result, we recovered 165 non-constant superpolies whose cube sizes range from 41 to 48. Compared with random cubes of similar sizes, it is very effective in recovering superpolies. We believe that the new method could be used to construct cubes of sizes around 60 with a bit more computation complexity, since subcubes of size 30 in the partition tests are obviously within experimental range and could be tested efficiently.

Observing 165 superpolies recovered for 815-round Trivium, it can be found that many are unbalanced. Whether this is due to the structure of Trivium or our cube sieve algorithm is unclear. Raising the success probability for balanced superpolies will be one subject of our future work.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Cannière C D and Preneel B, Trivium, *New Stream Cipher Designs — The eSTREAM Finalists*, LNCS, Springer, 2008, **4986**: 244–266.

[2] Dinur I and Shamir A, Cube attacks on tweakable black box polynomials, *Proc. Advances in Cryptology — EUROCRYPT* 2009, Germany, 2009, **2009**(LNCS 5479): 278–299.

[3] Fouque P and Vannet T, Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks, *Proc.* 20*th Int. Workshop, FSE* 2013, Singapore, 2013, **2013**(LNCS 8424): 502–517.

[4] Ye C and Tian T, A practical key-recovery attack on 805-round trivium, *Proc. Advances in Cryptology — ASIACRYPT* 2021, Singapore, 2021, **2021**(LNCS 13090): 187–213.

[5] Todo Y, Isobe T, Hao Y, et al., Cube attacks on non-blackbox polynomials based on division property, *Proc. Advances in Cryptology — CRYPTO* 2017, USA, 2017, **2017**(LNCS 10403): 250–279.

[6] Todo Y, Isobe T, Hao Y, et al., Cube attacks on non-blackbox polynomials based on division property, *IEEE Trans. Computers*, 2018, **67**(12): 1720–1736.

[7] Hao Y, Isobe T, Jiao L, et al., Improved division property based cube attacks exploiting algebraic properties of superpoly, *IEEE Trans. Computers*, 2019, **68**(10): 1470–1486.

[8] Wang S, Hu B, Guan J, et al., Milp-aided method of searching division property using three subsets and applications, *Proc. Advances in Cryptology — ASIACRYPT* 2019, Japan, 2019, **2019**(LNCS 11923): 399–427.

[9] Ye C and Tian T, Revisit division property based cube attacks: Key-recovery or distinguishing attacks? *IACR Trans. Symmetric Cryptol.*, 2019, **(3)**: 81–102.

[10] Hao Y, Leander G, Meier W, et al., Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead, *Proc. Advances in Cryptology — EUROCRYPT* 2020, Croatia, 2020, **2020**(LNCS 12105): 466–495.

[11] Hu K, Sun S, Wang M, et al., An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums, *Proc. Advances in Cryptology — ASIACRYPT* 2020, South Korea, 2020, **2020**(LNCS 12491): 446–476.

[12] Hu K, Sun S, Todo Y, et al., Massive superpoly recovery with nested monomial predictions, *Proc. Advances in Cryptology — ASIACRYPT* 2021, Singapore, 2021, **2021**(LNCS 13090): 392–421.

[13] Sun Y, Automatic search of cubes for attacking stream ciphers, *IACR Trans. Symmetric Cryptol.*, **2021**(4): 100–123.

[14] Liu M, Yang J, Wang W, et al., Correlation cube attacks: From weak-key distinguisher to key recovery, *Proc. Advances in Cryptology — EUROCRYPT* 2018, Israel, 2018, **2018**(LNCS 10821): 715–744.

[15] Dinur I, Güneysu T, Paar C, et al., An experimentally verified attack on full grain-128 using dedicated reconfigurable hardware, *Proc. Advances in Cryptology — ASIACRYPT* 2011, South Korea, 2011, **2011**(LNCS 7073): 327–343.

[16] Dinur I and Shamir A, Breaking grain-128 with dynamic cube attacks, *Proc.* 18*th Int. Workshop, FSE* 2011, Denmark, 2011, **2011**(LNCS 6733): 167–187.

[17] Rahimi M, Barmshory M, Mansouri M H, et al., Dynamic cube attack on grain-v1. *IET Inf. Secur.*, 2016, **10**(4): 165–172.

[18] Sarkar S, Maitra S and Baksi A, Observing biases in the state: Case studies with trivium and trivia-sc, *Des. Codes Cryptogr.*, 2017, **82**(1–2): 351–375.

[19] Stankovski P, Greedy distinguishers and nonrandomness detectors, *Proc. Progress in Cryptology — INDOCRYPT* 2010, India, 2010, **2010**(LNCS 6498): 210–226.

[20] Aumasson J, Dinur I, Meier W, et al., Cube testers and key recovery attacks on reduced-round MD6 and trivium, *Proc.* 16*th Int. Workshop, FSE* 2009, Belgium, 2009, **2009**(LNCS 5665): 1–22.

[21] Kesarwani A, Roy D, Sarkar S, et al., New cube distinguishers on nfsr-based stream ciphers, *Des. Codes Cryptogr.*, 2020, **88**(1): 173–199.

[22] Liu M, Degree evaluation of nfsr-based cryptosystems, *Proc. Advances in Cryptology — CRYPTO* 2017, USA, 2017, **2017**(LNCS 10403): 227–249.

[23] Liu M, Lin D, and Wang W, Searching cubes for testing boolean functions and its application to trivium, *Proc. IEEE International Symposium on Information Theory, ISIT* 2015, China, 2015, **2015**(IEEE): 496–500.

[24] Ye C and Tian T, Algebraic method to recover superpolies in cube attacks, *IET Inf. Secur.*, **14**(4): 430–441.

[25] Mroczkowski P and Szmidt J, The cube attack on stream cipher trivium and quadraticity tests, *Fundamenta Informaticae*, 2012, **114**: 309–318.

[26] Ye C and Tian T, A new framework for finding nonlinear superpolies in cube attacks against trivium-like ciphers, *Proc. Information Security and Privacy — 23rd Australasian Conf., ACISP*, 2018, **2018**LNCS 10946: 172–187.

[27] Todo Y, Structural evaluation by generalized integral property, *Proc. Advances in Cryptology — EUROCRYPT* 2015, Bulgaria, 2015, **2015**(LNCS 9056): 287–314.

[28] Todo Y and Morii M, Bit-based division property and application to simon family, *Proc. 23rd Int. Conference, FSE* 2016, Germany, 2016, **2016**(LNCS 9783): 357–377.

[29] Xiang Z, Zhang W, Bao Z, et al., Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers, *Proc. Advances in Cryptology — ASIACRYPT* 2016, Vietnam, 2016, **2016**(LNCS 13090): 648–678.