An Experimentally Verified Attack on 820-Round Trivium (Full Version)

Cheng Che and Tian Tian(✉)

PLA Strategic Support Force Information Engineering University, Zhengzhou 450001, China. `che_cheng@126.com,tiantian_d@126.com`

Abstract. The cube attack is one of the most important cryptanalytic techniques against Trivium. As the method of recovering superpolies becomes more and more effective, another problem of cube attacks, i.e., how to select cubes corresponding to balanced superpolies, is attracting more and more attention. It is well-known that a balanced superpoly could be used in both theoretical and practical analyses. In this paper, we present a novel framework to search for valuable cubes whose superpolies have an independent secret variable each, i.e., a linear variable not appearing in any nonlinear term. To control online complexity, valuable cubes are selected from very few large cubes. New ideas are given on the large cube construction and the subcube sieve.

For the verification of this new algorithm, we apply it to Trivium. For 815-round Trivium, using one cube of size 47, we obtain more than 200 balanced superpolies containing 68 different independent secret variables. To make a trade-off between the number of cubes and computation complexity, we choose 35 balanced superpolies and mount a key-recovery attack on 815-round Trivium with a complexity of $2^{47.32}$. For 820-round Trivium, using two cubes of size 52, we obtain more than 100 balanced superpolies, which contain 54 different independent secret variables. With 30 balanced superpolies, we mount a key-recovery attack on 820-round Trivium with a complexity of $2^{53.17}$. Strong experimental evidence shows that the full key-recovery attacks on 815- and 820-round Trivium could be completed within six hours and two weeks on a PC with two RTX3090 GPUs, respectively.

Keywords: Cube Attacks· Key-Recovery Attacks· Division Property· Trivium.

1 Introduction

Cube Attack: The cube attack is a new method of analyzing symmetric-key cryptosystems proposed by Dinur and Shamir in [1], which absorbs the ideas of higher-order differential attacks, saturation attacks, and chosen IV statistical attacks. 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 $\boldsymbol{x} = (x_1, x_2, ..., x_n)$ are secret key variables and $\boldsymbol{v} = (v_1, v_2, ..., v_m)$ are public

IV variables. For a randomly chosen set $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\} \subset \{v_1, v_2, ..., v_m\}$, $f(\boldsymbol{x}, \boldsymbol{v})$ can be represented uniquely as

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

where $t_I = v_{i_1} \cdots v_{i_{|I|}}$, and q_I misses at least one variable in I. The IV variables in I are set as active, that is, all possible combinations of $0/1$ are taken, and the rest of the IV variables are inactive and set to constants. The set of these values is called a cube, and the sum of $f(\boldsymbol{x}, \boldsymbol{v})$ over all values of the cube is evaluated, which is the value of $p_I(\boldsymbol{x}, \boldsymbol{v})$ for a given secret key. The polynomial $p_I(\boldsymbol{x}, \boldsymbol{v})$ is called the superpoly of the cube, and it is much simpler than $f(\boldsymbol{x}, \boldsymbol{v})$. Once an attacker recovers a certain number of superpolies, he could build a system of equations on secret key variables $\boldsymbol{x}$ by inquiring the values of all the superpolies. Then some information about the secret variables can be retrieved. It can be seen that the critical step of cube attacks is to recover a bundle of nonconstant superpolies.

In [1,2], $f(\boldsymbol{x}, \boldsymbol{v})$ is regarded as a blackbox polynomial and analyzed experimentally because the algebraic normal form of $f(\boldsymbol{x}, \boldsymbol{v})$ is too complicated. However, such experimental analysis has significant drawbacks, e.g., the cube size is limited to the experimental range.

Division Property: In [3], the division property was proposed, which enables cube attacks to analyze cryptographic algorithms as a non-blackbox polynomial, and thus significantly improves the analytical ability of cube attacks. The division property is a generalization of integral property, originally for block ciphers. It can exploit the algebraic structure of block ciphers to construct integral distinguishers even if the block cipher has non-bijective or bit-oriented structures. In [4], the division property was first introduced to cube attacks on stream ciphers, and it could be used to identify the secret variables not involved in the superpoly efficiently. To improve the effectiveness of cube attacks based on division property, some new techniques were given in [5,6].

However, the traditional division property only confirms that a specific monomial does not appear in the superpoly if the division property cannot propagate to the output bit. If the division property can propagate to the output bit, it is not clear whether the corresponding monomial appears. This inaccuracy of the traditional division property makes many previous key-recovery attacks, e.g., [7,5], degenerate to distinguishing attacks [8,6]. This was finally resolved by Hao et al. in [9], where the model for three-subset division property without unknown subset was proposed.

Trivium: Trivium [10] is a bit-oriented synchronous stream cipher designed by De Cannière and Preneel, which is one of the eSTREAM hardware-oriented finalists and an International Standard under ISO/IEC 29192-3:2012. Because of its simple structure and high level of security, Trivium has attracted extensive attention.

When the cube attack was first proposed, a key-recovery attack on 767-round Trivium was given, in which 35 linear superpolies were recovered by linearity tests [1]. Next, key-recovery attacks on 784- and 799-round of Trivium were given in

[2]. Recently, an effective method to construct cubes for linear superpolies was proposed in [11], and a practical attack against 805-round Trivium was given. On the other hand, cube attacks based on division property theoretically evaluate the security of Trivium by targeting a very high round number. In [9], Hao et al. accurately recovered the superpolies of 840-, 841-, and 842-round Trivium by using three-subset division property without unknown subset. Meanwhile, Hu et al. in [12] described the propagation of monomials from a pure algebraic perspective and proposed monomial prediction technique, and thus accurately recovered the superpolies of 840-, 841-, and 842-round Trivium. Very recently, Hu et al. [13] combined the monomial prediction technique with the backtracking method in [8] and presented a new framework for recovering the exact ANFs of massive superpolies, recovering the superpolies for 843-, 844- and 845-round Trivium. At FSE 2021, Sun proposed a new heuristic method in [14] to reject cubes without independent secret variables from a preset of candidate cubes. Using the heuristic algorithm, the author of [14] recovered a balanced superpoly for 843-round Trivium and presented practical attacks against 806- and 808-round Trivium.

Motivation: Our work aims to enhance the ability of a practical key-recovery attack on Trivium. Firstly, in our attacks, we use a special class of balanced superpolies. It is well known that a balanced superpoly could recover 1-bit key information. However, n balanced superpolies do not always provide n-bit key information. For a superpoly p, if p could be decomposed into $p(\boldsymbol{k}) = p'(\boldsymbol{k}) \oplus k_i$, where k_i does not appear in p', then we say that p has an **independent variable** k_i. To facilitate the key recovery process, we only consider superpolies with an independent variable. This is because a superpoly with an independent variable is balanced. If we obtain many such superpolies, it is easy to select n superpolies to provide n-bit key information. In the following paper, we say a cube is **valuable** if its superpoly has at least one independent secret variable. Secondly, to reduce the complexity of the online phase, the existing practical attacks [11,14] all selected a large cube I and searched the subcubes of I to recover balanced superpolies. So, the attacker only needs $2^{|I|}$ times to query the encryption oracle. In the following paper, a desirable large cube is called a **mother cube**. Therefore, a good mother cube and a method of searching for valuable subcubes are critical for practical attacks. This is different from the theoretical attack on high-round Trivium.

Our work is motivated by the heuristic method of rejecting useless cubes in [14]. When the attacker recovers the superpoly, the whole system can be split into several subsystems based on the divide-and-conquer strategy. The algorithm proposed in [14] simultaneously solves a set of cubes in a subsystem. Sun considered that for a cube, if the subsystem is solved to obtain the high-degree term, the whole system will obtain the term with a high probability, so it can be considered that the superpoly of the cube does not contain the independent secret variables appearing in the term. Therefore, for a given secret variable k_i, useless cubes can be rejected from a preset of candidate cubes by solving a subsystem. When multiple subsystems are tested in this way, the remaining cubes are more likely to have independent secret variables k_i. There are two obvious drawbacks of this search algorithm. On one hand, both rejection and acceptance of this method may be wrong, and it is difficult for practical attacks to control the size of cube if some valuable cubes are rejected. On the other hand, when there are too many subcubes in the test, it cannot terminate within an acceptable period of time. The reason why the search cannot be completed in a short time may be that some complicated subcubes are difficult to solve.

Instead of rejecting useless cubes, we consider choosing cubes that are more likely to be valuable. Our work is based on two observations. Firstly, we observe that low-degree superpolies are easier to recover and more likely to contain independent variables. Moreover, for many low-degree balanced superpolies, it is easier to select n balanced superpolies to provide n-bits of key information.Therefore, we consider giving a batch of subcubes with low-degree superpolies through the algebraic degree evaluation. For simplicity, **the degree of a cube** refers to the degree of the superpoly corresponding to the cube, and a cube with a low-degree superpoly is called a **low-degree cube**. Secondly, we note that the existence of linear terms is necessary for a superpoly to have independent secret variables. We find that many superpolies do not have linear terms in experimental tests. It is obvious that these superpolies without linear terms cannot contain independent secret variables. Based on this observation, we can determine whether the cube was rejected by recovering the linear terms.

Our Contribution: This paper is devoted to practical key-recovery attacks against Trivium. We present a novel framework to search for valuable subcubes from a mother cube, which is experimentally verified to be quite effective. It consists of the following three aspects.

1. We modify the algorithm for constructing cubes targeting linear superpolies presented in [11]. We aim to construct a mother cube with many low-degree subcubes rather than several low-degree cubes unrelated to each other. Therefore, we modify the end of the first stage in order to construct a potentially good mother cube.
2. We propose an efficient method to search for low-degree subcubes. For a mother cube, it is not practical to enumerate the algebraic degrees of all subcubes. We use a deep-first-search strategy. We first enumerate the degrees of all the subcubes with one less variable for a given mother cube. Then, for the subcubes with degree less than 5, we continually enumerate the subcubes with one less variable until there is no subcube with degree less than 5. As a result, we can identify most of the low-degree subcubes efficiently.
3. We propose a method for searching for valuable subcubes. We note that the existence of linear terms is necessary for a superpoly to have independent secret variables. Moreover, since linear terms account for only a small part of the superpoly, it is efficient to recover all linear terms of a cube. Therefore, we recover the linear terms of the low-degree subcubes and reject the subcubes without linear terms, or all the linear terms have been covered by some simple superpolies. Experimental data on 820-Trivium show that only about

20% of the superpolies are left after the rejection. Finally, we recover the superpolies of the remaining cubes by solving the subsystem.

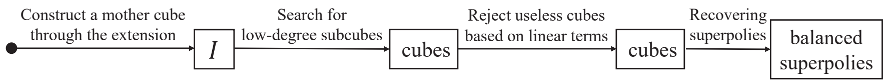
Construct a mother cube through the extension
I
Search for low-degree subcubes
cubes
Reject useless cubes based on linear terms
cubes
Recovering superpolies
balanced superpolies

Fig. 1. The sketch of our idea

As an illustration, we apply our methods, whose sketch is shown in Fig. 1, to Trivium. The new algorithm successfully finds many valuable cubes for 815- and 820-round Trivium, and we also recover all their superpolies. Practical attacks on 815- and 820-round Trivium are given. As a comparison, we summarize full key-recovery attacks against the round-reduced Trivium in Table 1.

1. For 815-round Trivium, we construct a mother cube I_1, whose size is 47. Then, we search subcubes with one less variable of 16 46-dimensional and more than 100 45-dimensional subcubes. Using low-degree subcubes with linear terms, we obtain more than 200 balanced superpolies containing 68 different independent secret variables. Through linearization, we chose 35 superpolies that excluded enough illegal keys to leave only 2^{45} possible keys. The total online complexity for attacking 815-round Trivium is $2^{47} + 2^{45}$, which can be done practically.
2. For 820-round Trivium, we construct two mother cubes I_2 and I_3, both of which have size 52. Initially, we consider I_2 and search for its subcubes. However, after obtaining some independent secret variables, we found that the recovered linear terms are frequently repeated. We guess most of the independent secret variables related to I_2 have been recovered. Therefore, we construct another mother cube I_3. Finally, we obtain more than 100 balanced superpolies containing 54 different independent secret variables using I_2 and I_3. After choosing 30 balanced superpolies, we can enumerate the values of 50 secret variables and obtain the values of the remaining 30 secret variables within constant time. The total online complexity for attacking 820-round Trivium is $2^{53}+2^{50}$. This attack on 820-round Trivium improves the previous best practical cube attacks by 12 more rounds.

The source codes of the proposed algorithm, including those searching for valuable cubes and recovering superpolies, were released at https://github.com/LLuckyRabbit/search-for-valuables-cubes.

This paper is organized as follows. Section 2 introduces necessary notations and some preliminaries. The new search algorithm is reported in Section 3, and its applications to Trivium are in Section 4. Conclusions are drawn in Section 5.

2 Preliminaries

In this section, we introduce some related concepts and definitions.

Table 1. A summary of key-recovery attacks on Trivium

Attack type	# of rounds	Off-line phase		On-line phase	Total time	ref.
		cube size	# of key bits			
Practical	672	12	63	2^{17}	$2^{18.56}$	[1]
	767	28-31	35	2^{45}	$2^{45.00}$	[1]
	784	30-33	42	2^{38}	$2^{39.00}$	[2]
	805	32-38	42	2^{38}	$2^{41.40}$	[11]
	806	33-37	45	2^{35}	$2^{39.88}$	[14]
	808	39-41	37	2^{43}	$2^{44.58}$	[14]
	815	44-46	35	2^{45}	$2^{47.32}$	Sect. 4.1
	820	48-51	30	2^{50}	$2^{53.17}$	Sect. 4.2
Theoretical	799	32-37	18	2^{62}	$2^{62.00}$	[2]
	802	34-37	8	2^{72}	$2^{72.00}$	[15]
	805	28	7	2^{73}	$2^{73.00}$	[16]
	832	72	1	2^{79}	$2^{79.01}$	[6,4,17]
	832	72	> 1	2^{79}	$< 2^{79.01}$	[18]
	835	35	5	2^{75}	$2^{75.00}$	[16]
	840	75	3	2^{77}	$2^{77.32}$	[12]
	840	78	1	2^{79}	$2^{79.58}$	[9]
	841	78	1	2^{79}	$2^{79.58}$	[9]
	841	76	2	2^{78}	$2^{78.58}$	[12]
	842	76	2	2^{79}	$2^{78.58}$	[12]
	842	78	1	2^{79}	$2^{79.58}$	[19]
	843	54-57,76	5	2^{75}	$2^{76.58}$	[13]
	843	78	1	2^{79}	$2^{79.58}$	[14]
	844	54-55	2	2^{78}	$2^{78.00}$	[13]
	845	54-55	2	2^{78}	$2^{78.00}$	[13]

We take 2^{60} as the boundary between practical attack and theoretical attack. Because computations within 2^{60} can be completed in a limited time with limited resources.

2.1 Boolean functions and algebraic degree

A Boolean function on n variables is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, where $\mathbb{F}_2$ is the finite field with two elements and $\mathbb{F}_2^n$ is an n-dimensional vector space over $\mathbb{F}_2$. A Boolean function f can be uniquely represented as a multivariable polynomial over $\mathbb{F}_2$,

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{\boldsymbol{c}=(c_1,c_2,\ldots,c_n)\in\{0,1\}^n} a_{\boldsymbol{c}} \prod_{i=1}^{n} x_i^{c_i},$$

which is called the algebraic normal form (ANF) of f, where $a_{\boldsymbol{c}} \in \mathbb{F}_2$. In the following paper, $\prod_{i=1}^{n} x_i^{c_i}$ is called a term of f. One important feature of a Boolean function is its algebraic degree which is denoted by $\deg(f)$ and defined as

$$\deg(f) = \max\{wt(\boldsymbol{c}) | a_{\boldsymbol{c}} \neq 0\},$$

where $wt(\boldsymbol{c})$ is the Hamming Weight of $\boldsymbol{c}$, i.e., $wt(\boldsymbol{c}) = \sum_{i=1}^{n} c_i$.

2.2 Trivium

Trivium is a bit-oriented synchronous stream cipher that was one of the eSTREAM hardware-oriented finalists. The main building block of Trivium is a Galois nonlinear feedback shift register, and its internal states are 288 bits in total. For every clock cycle, three bits of the internal state are updated by quadratic feedback functions, and all the remaining bits of the internal state are updated by shifting. 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. Algorithm 1 describes the pseudo-code of Trivium. For more details, please refer to [10].

Algorithm 1 Pseudo-code of Trivium

```
1: $(s_1, s_2, \ldots, s_{93}) \leftarrow (k_1, k_2, \ldots, k_{80}, 0, \ldots, 0)$;
2: $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (v_1, v_2, \ldots, v_{80}, 0, \ldots, 0)$;
3: $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$;
4: for $i$ from 1 to $N$ do
5:     $t_1 \leftarrow s_{66} \oplus s_{93} \oplus s_{91}s_{92} \oplus s_{171}$;
6:     $t_2 \leftarrow s_{162} \oplus s_{177} \oplus s_{175}s_{176} \oplus s_{264}$;
7:     $t_3 \leftarrow s_{243} \oplus s_{288} \oplus s_{286}s_{287} \oplus s_{69}$;
8:     if $i > 1152$ then
9:         $z_{i-1152} \leftarrow s_{66} \oplus s_{93} \oplus s_{162} \oplus s_{177} \oplus s_{243} \oplus s_{288}$;
10:    end if
11:    $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$;
12:    $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$;
13:    $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$;
14: end for
```

2.3 Cube attacks

The cube attack was first proposed by Dinur and Shamir in [1]. In a cube attack, the output bit of the cryptographic algorithm can be regarded as a Boolean function in the secret variables $\boldsymbol{k} = (k_1, k_2, \ldots, k_n)$ and the public variables $\boldsymbol{v} = (v_1, v_2 \ldots, v_m)$, expressed as $f(\boldsymbol{k}, \boldsymbol{v})$. For a randomly chosen set $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_d}\}$, $f(\boldsymbol{k}, \boldsymbol{v})$ can be represented uniquely as

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

where $t_I = v_{i_1} \cdots v_{i_d}$, and q_I misses at least one variable in I. The public variables in I are called **cube variables**, while the remaining IV variables are called non-cube variables. The IV variables in I are set as active, that is, all possible combinations of $0/1$ are taken, and the rest of the IV variables are inactive and set to constants. The set of these values is denoted as a **cube**, and the polynomial p_I is called the **superpoly** of C_I in f. For convenience, we also call p_I the superpoly of I in f. It can be seen that the summation of the 2^d functions derived from f by assigning all the possible values to d variables in I equals to p_I, that is,

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

Therefore, in the online phase, it takes 2^d requests (that is, 2^d calls to the initialization function) to get the value of the superpoly $p_I(\boldsymbol{k}, \boldsymbol{v})$.

Obviously, $p_I(\boldsymbol{k}, \boldsymbol{v})$ is much simpler than $f(\boldsymbol{k}, \boldsymbol{v})$. Once an attacker recovers a certain number of superpolies, he could build a system of equations on secret key variables $\boldsymbol{k}$ by inquiring the values of all the superpolies. Then some information about the secret variables can be achieved. In particular, if a superpoly is balanced, namely $|\{\boldsymbol{k} \in \mathbb{F}_2^n \mid f(\boldsymbol{k}) = 0\}| = |\{\boldsymbol{k} \in \mathbb{F}_2^n \mid f(\boldsymbol{k}) = 1\}| = 2^{n-1}$, then 2^{n-1} illegal keys will be filtered out. However, in a key recovery attack, it is difficult to obtain ℓ-bits of information about the key even if there are ℓ balanced superpolies. Moreover, when ℓ is large, it is almost impossible. We use the following toy example to illustrate this problem.

Example 1. Let $f_1 = x_1x_2 \oplus x_3 \oplus x_4$ and $f_2 = x_2x_3 \oplus x_1 \oplus x_4$ be balanced polynomials on the variables $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$. It is easy to verify

$$|\{\boldsymbol{x} \in \mathbb{F}_2^4 \mid f_1(\boldsymbol{x}) = 0, f_2(\boldsymbol{x}) = 0\}| = |\{\boldsymbol{x} \in \mathbb{F}_2^4 \mid f_1(\boldsymbol{x}) = 1, f_2(\boldsymbol{x}) = 1\}| = 6,$$

$$|\{\boldsymbol{x} \in \mathbb{F}_2^4 \mid f_1(\boldsymbol{x}) = 1, f_2(\boldsymbol{x}) = 0\}| = |\{\boldsymbol{x} \in \mathbb{F}_2^4 \mid f_1(\boldsymbol{x}) = 0, f_2(\boldsymbol{x}) = 1\}| = 2.$$

When $f_1 = f_2 = 0$ or $f_1 = f_2 = 1$, there are 6 possible variables left, not $2^{4-2} = 4$. It can be seen that the two balanced superpolies do not provide 2-bits of information.

2.4 The division property

In the following paper, we use the conventional bit-based division property to evaluate the algebraic degree and the three-subset division property without unknown subset to recover superpoly.

The bit-based division property. In [20], the authors proposed the conventional bit-based division property whose definition is as follows.

Definition 1 (Conventional Bit-Based Division Property [20]). *Let $\mathbb{X}$ be a multi-set 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 multi-set $\mathbb{X}$ has the division property $D_{\mathbb{K}}^{1^n}$, it fulfills the following conditions:*

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

where $\boldsymbol{u} \succeq \boldsymbol{\alpha}$ if and only if $u_i \geq k_i$ for all i and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i^{u_i}$.

The division property depends on some vectors, and the propagation of the division property for every round function is actually the transformation of vectors. Based on this observation, Xiang et al. in [21] first introduced the concept of division trails, a set of ordered vectors describing the propagation of division property. Based on the concept of division trails, a linear inequality system is proposed to describe the propagation of division property. For all feasible solutions of the linear inequality system to correspond precisely to all division trails, the authors described the propagation rules for AND, COPY, and XOR with MILP models, see [21] for the details. Therefore, they could build an Mixed Integer Linear Programming (MILP) model to cover all the possible division trials generated during the propagation.

The division property based degree evaluation. Since Todo et al. introduced division property into cube attack [4], the process of cube attack was converted into an MILP problem, and mathematical software was used for calculation, which significantly improved the power of the cube attacks. The algebraic degree of superpolies is also an important feature, which can be used for searching the cube and other attacks. In the following paper, we use the degree evaluation algorithm proposed in [5], which was based on the following proposition.

Proposition 1 ([5]). *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 $I=\{i_1,i_2,\ldots,i_{|I|}\}\subset\{1,2,\ldots,m\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1},v_{i_2},\ldots,v_{i_{|I|}}\}$ are taking all possible combinations of values. Let $\boldsymbol{k}_I$ be an m-dimensional bit vector such that $\boldsymbol{v}^{\boldsymbol{I}}=t_I=v_{i_1}v_{i_2}\cdots v_{i_{|I|}}$. Let $\boldsymbol{k}_\Lambda$ be an n-dimensional bit vector. If there is no division trail such that $(\boldsymbol{k}_\Lambda||\boldsymbol{k}_I)\xrightarrow{f}1$, then the monomial $x^{\boldsymbol{k}_\Lambda}$ is not involved in the superpoly of the cube C_I.*

According to Proposition 1, for a positive integer d, if all the division property $x^{\boldsymbol{k}_\Lambda}$ with $wt(\boldsymbol{k}_\Lambda)>d$ cannot propagate to the output bit, then it is shown that d is an upper bound on the algebraic degree of the superpoly. Therefore, the attacker can evaluate the algebraic degree of a stream cipher by solving the MILP model that maximizes the objective function $\sum_{j=1}^{n}x_j$. For more details, please refer to Section 4 of [5].

The three-subset division property. The set of u is divided into two subsets in the conventional division property, where one is the subset such that $\bigoplus_{\boldsymbol{x}\in\mathbb{X}}\boldsymbol{x}^{\boldsymbol{u}}$ is unknown, and the other is the subset such that the sum is 0. Three-subset division property extends the number of subsets from 2 to 3, and its definition is given below.

Definition 2 (Three-Subset Division Property [20]). *Let $\mathbb{X}$ be a multi-set 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 multi-set $\mathbb{X}$ has the division property $D^{1^n}_{\mathbb{K},\mathbb{L}}$, it fulfills the following conditions:*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}}\boldsymbol{x}^{\boldsymbol{u}}=\begin{cases}\textit{unknown} & \textit{if there exists } \boldsymbol{\alpha} \textit{ in } \mathbb{K} \textit{ s.t. } \boldsymbol{u}\succeq\boldsymbol{\alpha},\\ 1 & \textit{else if there exists } \boldsymbol{\beta} \textit{ in } \mathbb{L} \textit{ s.t. } \boldsymbol{u}=\boldsymbol{\beta},\\ 0 & \textit{otherwise,}\end{cases}$$

where $\boldsymbol{u} \succeq \boldsymbol{\alpha}$ *if and only if* $u_i \geq k_i$ *for all* i *and* $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i^{u_i}$.

Wang et al. [6] adopted the concept of three-subset division property and developed an algorithm to recover superpolies, which reduced the time complexity of the algorithm. Hao et al. [9] found that recovering superpolies based on three subsets is not always effective because of the large size of $\mathbb{L}$. Therefore, the concept of three-subset division property without unknown subset and a new MILP modeling method which could be used to recover the exact superpoly for a given cube are proposed, see [9] for details. In Section 4, we will recover superpolies using three-subset division property without unknown subset.

2.5 A heuristic algorithm of constructing cubes targeting linear superpolies

In [11], Ye et al. combined the division property based degree evaluation method with some greedy strategies to construct cubes targeting linear superpolies. The author heuristically gave a small set of cube variables and then extended it iteratively. In order to construct cubes targeting linear superpolies, the extension phase is subdivided into two stages. Before reviewing the two stages, we shall first give the following definitions.

Definition 3 (Steep IV Variable [11]). *Let* $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}\}$ *be a set containing* ℓ *cube variables. Then, an IV variable* $b \in B = \{v_0, v_1, \ldots, v_{m-1}\} \setminus I$ *is called a steep IV variable of* I *if* $ds(I \cup \{b\}) = \min\{ds(I \cup \{v\}) | v \in B\}$, *where* $ds(I)$ *is the degree of the superpoly of* I *in key variables.*

Definition 4 (Gentle IV Variable [11]). *Let* $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}\}$ *be a set containing* ℓ *cube variables. Then, an IV variable* $b \in B$ *is called a gentle IV variable of* I *if* $ds(I \cup \{b\}) = \max\{ds(I \cup \{v\}) | ds(I \cup \{v\}) \leq ds(I), v \in B\}$, *where* $B = \{v_0, v_1, \ldots, v_{m-1}\} \setminus I$ *and* $ds(I)$ *is the degree of the superpoly of* I.

In the first stage, selecting a steep IV variable in each extension can quickly reduce the degree of the superpoly corresponding to the cube. However, it may fail to construct cubes with linear superpolies by only adding steep IV variables. The goal of the second stage is to ensure that the degree of the superpoly could be close to 1 rather than suddenly dropping to 0. Therefore, Ye et al. proposed selecting a gentle IV variable in each extension, which slowly decreases the superpoly degree. It is more hopeful to construct cubes with linear superpolies.

3 A Search Algorithm for Valuable Cubes

To mount key-recovery attacks, enough valuable cubes need to be collected. A modified algorithm for constructing mother cubes and an efficient method for searching low-degree subcubes are introduced in Sections 3.1 and 3.2, respectively. In Section 3.3, we present a method for rejecting useless cubes. Combining the new methods given in these three sections, we present a novel general framework for making key-recovery attacks.

3.1 A modified algorithm of constructing mother cubes

In [11], Ye et al. completed a practical key-recovery attack on 805-round Trivium using linear superpolies. As the number of rounds increases, the size of the candidate cube increases, and the number of linear superpolies decreases. Consequently, recovering linear superpolies is not enough to mount a practical attack. We aim to extend linear superpolies to low-degree superpolies in a practical attack. To reduce the number of requests in the online phase, we construct a mother cube and then search its subcubes. When constructing the mother cube, we want a cube that has many low-degree subcubes and is as small as possible. Therefore, we do not need to add some gentle IV variables that make the degree of cube approach 1, and we only need the first stage of the algorithm in Section 2.5.

We modify the beginning and the end of the first stage. In [11], the authors gave a method for determining starting cube sets. In fact, many starting cubes meet the criteria. We consider selecting the cube with the smallest degree as a starting set. In the end, because the iteration only adds a steep IV variable, the degree of the cube drops to 0 quickly. Then, in the last iteration, the resulting cubes with a non-zero degree are low-degree cubes. However, we want a mother cube that contains many low-degree subcubes, not some cubes whose last variable is different. Therefore, for the results of the last iteration, we select several IV variables to add to the cube to get a large cube. Then, how many IV variables to choose and which IV variables to choose is a matter for us to consider.

We focus on IV variables that reduce the cube to a lower degree, such as IV variables with a cube degree less than five after being added. For example, assuming there are five such IV variables, we will get ten new large cubes if we combine the three and add them to the original cube. The algebraic degrees of all subcubes with one less variable are evaluated for the ten large cubes. The large cube is selected as the mother cube if it has the most subcubes of degree less than 5. In particular, if the degrees of all subcubes are not small, for example, greater than 4, it is considered that more IV variables are needed to reduce the degree of the whole.

3.2 A method for searching low-degree subcubes

For the mother cube obtained in Section 3.1, the next step is to search for valuable subcubes. In [14], Sun directly dealt with candidate subcubes to recover the terms containing the secret variable k_i in the subsystem to select the cube that is more likely to have independent variable k_i. However, when attacking 820-round Trivium, a mother cube of size 52 has many subcubes. If we deal with them together and then judge, the time is unacceptable. One of the most straightforward solutions is to divide the whole system into smaller subsystems, making the search algorithm more inaccurate. Also, all subsystems have to be tested, which can be time-consuming.

Among all subcubes, there will be some complicated cubes, which occupy most of the solving time. Naturally, we tend to choose subcubes that are more likely to be valuable and solve faster. The algebraic degree is the most common way to measure the complexity of a cube. It is generally agreed that low-degree superpolies are easier to recover. In particular, low-degree superpolies are also more likely to have independent variables. Therefore, we first evaluate the degree of a subcube and then judge whether the subcube is valuable. Since the subcubes are contained in a large cube, we can strategically evaluate algebraic degrees. Our strategy is based on the following observation.

Observation 1. *For a given cube I, the degree of most subcubes is higher than the degree of I.*

We do a simple experimental test and select ten large cubes. In all the subcubes with one variable less, about 90% of the subcubes have higher degrees than the original cube. Therefore, we use a deep-first-search strategy to evaluate degrees.

Taking 815-round Trivium as an example, for a 47-dimensional cube, we first enumerate the degrees of 46-dimensional subcubes. Then enumerate the subcubes with one less variable for the cubes whose degrees are less than 5 in the 46-dimensional subcubes. Because of observation 1, it is difficult to get some low-degree subcubes in a cube whose degree is greater than 5. Therefore, we always enumerate subcubes with one less variable for low-degree cubes until there is no cube whose degree is less than 5. Using this strategy, we can enumerate a small number of subcubes and obtain the vast majority of low-degree subcubes, effectively speeding up the search for low-degree subcubes.

3.3 A method for searching valuable subcubes

A large number of low-degree subcubes can be obtained using the method in Section 3.2. The problem to be solved here is how to search for valuable cubes among many low-degree subcubes. The natural idea is to test whether a specific secret variable k_i is independent. This method only needs to compute the monomials that involve the variable k_i and is more efficient. However, Trivium, for example, has 80 secret variables. If each secret variable is tested, it can be more complicated than recovering the superpolies directly, so verifying all secret variables for a cube is impractical.

Instead of pointlessly verifying all secret variables, we consider which variables need to be verified for a given cube. We note that the independent secret variable is a linear term, and the existence of linear terms is necessary for a superpoly to have independent secret variables. Therefore, if a superpoly does not contain linear terms, we can reject it without error. If a superpoly contains linear terms, we only need to verify that these variables are independent. Based on this consideration, we give the criteria for the primary filtration of candidate cubes.

The primary filtration. *If the superpoly corresponding to a cube has no linear term, then we can reject the cube.*

Table 2. Statistics the remaining cubes after filtering for 820-round Trivium

Dim	#Cubes	#Cubes with linear terms	#Remaining cubes
51	31	7	3
50	78	30	17
49	364	157	77
48	195	96	54
total	668	290	151

Linear terms account for only a small part of the superpoly, so it is efficient to recover all the linear terms of a cube. We make simple statistics on the low-degree cubes of 820-round Trivium, and the cube with linear terms accounts for about 40%. The specific data are listed in Table 2.

After we recover the superpolies of many subcubes, we find that these superpolies have complex algebraic relations, that is, one superpoly may be generated by the combination of other superpolies. Therefore, some linear terms occur frequently in superpolies. We only need one superpoly containing the independent secret variable for a linear term that occurs frequently. Therefore, we can also do secondary filtering for cubes that contain the same linear term.

The secondary filtration. *If the linear terms of the superpoly corresponding to a cube have been recovered in linear or quadratic balanced superpolies, then we can reject the cube.*

For independent variables appearing in balanced superpolies of degree three or more, we expect to obtain simpler superpolies. Therefore, we reject only the independent variables previously obtained in linear or quadratic balanced superpolies. In particular, to perform the secondary filtration more efficiently, we first recover the cube with the lowest degree for all cubes with the same linear term. In the fourth column of Table 2, we also list the remaining cubes filtered twice during the practical recovery process. The details of our idea are described in Algorithm 2.

When recovering the remaining cubes, we can use the observation given in [14] that the higher-degree terms obtained by the subsystem will appear in the superpoly with high probability. Therefore, we divide the whole system into several subsystems to solve the superpoly. If the high-degree term related to the linear term is obtained in a subsystem, then we can reject the cube. In the practical recovery, the superpolies that we fully recover are all balanced.

4 Applications

In this section, we apply the new framework to Trivium. Practical attacks on 815- and 820-round Trivium are given. Due to the page limit, superpolies used in this section can be found at https://github.com/LLuckyRabbit/search-for-valuables-cubes/tree/main/superpolies/results.

Algorithm 2 The algorithm of searching valuable cubes based on linear terms

```
Require: a set of low-degree cubes $B = \{I_1, \ldots, I_c\}$ and the target round $r$
 1: $K \leftarrow \emptyset$;
 2: for $I \in B$ do
 3:     Recover the linear terms of the superpoly corresponding to cube $I$;
 4:     $L \leftarrow$ linear terms variables;
        /* Primary filtration and secondary filtration */
 5:     if $L \neq \emptyset$ and $L \not\subset K$ then
 6:         Recover the superpoly $p$ corresponding to cube $I$;
 7:         if superpoly $p$ is balanced then
 8:             Record cube $I$ and superpoly $p$;
                /* The independent secret variables corresponding to the simple superpoly is updated,
                which can be used for secondary filtering. */
 9:             if $\deg(p) \leq 2$ then
10:                 $K \leftarrow K \cup \{$independent secret variables$\}$;
11:             end if
12:         end if
13:     end if
14: end for
```

4.1 A practical key-recovery attack on 815-round Trivium

In this subsection, we target 815-round Trivium. According to the modified algorithm for constructing the mother cube in Section 3.1, we obtain a set of indexes I_1, where

$$I_1 = \{0,1,2,3,4,6,8,10,12,14,15,16,18,19,20,23,25,27,29,31,33,34,35,36,$$
$$37,40,41,42,43,44,46,51,53,55,56,58,59,60,61,62,66,67,69,71,73,77,79\}.$$

The size of I_1 is 47, so it only needs 2^{47} requests to obtain all the values of superpolies whose related cubes are from the subsets of I_1.

In I_1, there are 13 46-dimensional subcubes of degrees less than 5. In particular, there are three subcubes whose algebraic degree is evaluated as five and whose corresponding superpolies are linear or constant. Therefore, we choose the 16 46-dimensional subcubes to continue enumerating their 45-dimensional subcubes. If more balanced superpolies are needed, some 45-dimensional subcubes are selected to evaluate the algebraic degrees of their 44-dimensional subcubes until we get enough balanced superpolies. We obtain over 200 balanced superpolies for 815-round Trivium and picked 35 to attack. These selected valuable cubes and their corresponding independent secret variables are listed in Table 3, and the specific superpolies can be found on the GitHub website. There are ten linear superpolies among them. As can be seen from Table 3, among the 35 balanced superpolies selected, 13 are from the 46-dimensional subcube $I_1\backslash\{44\}$, and 11 are from the 46-dimensional subcube $I_1\backslash\{67\}$. This shows that the importance of subcubes is different.

To attack 815-round Trivium practically, we need a linearization method to deal with nonlinear balanced superpolies. Consistent with the method in [14], we first enumerate some variables to linearize the partial superpolies to obtain the values of the new independent variables. Once the values of some independent variables are deduced, they can be used to deduce the values of other independent

Table 3. Valuable cubes for attacking 815-round Trivium

Cube indices	Independent bits	Cube indices	Independent bits
$I_1\backslash\{58,67\}$	k_{23}	$I_1\backslash\{44,71\}$	$k_{33},k_{42},k_{51},k_{62}$
$I_1\backslash\{43,67\}$	k_{35}	$I_1\backslash\{41,44\}$	$k_{33},k_{38},k_{42},k_{49},k_{60},k_{65}$
$I_1\backslash\{19,58,67\}$	k_{48}	$I_1\backslash\{2,42,44\}$	k_4,k_{54}
$I_1\backslash\{2,44,60\}$	k_{35},k_{49}	$I_1\backslash\{33,67\}$	k_{12},k_{39}
$I_1\backslash\{35,43\}$	k_{35},k_{56}	$I_1\backslash\{19,67\}$	$k_{12},k_{31},k_{39},k_{40},k_{47},k_{67}$
$I_1\backslash\{2\}$	k_{58}	$I_1\backslash\{14,41\}$	k_5,k_{14},k_{41},k_{42}
$I_1\backslash\{41,44,67\}$	k_{60}	$I_1\backslash\{56,58,67\}$	k_{25}
$I_1\backslash\{58,59\}$	k_{62}	$I_1\backslash\{58,67,71\}$	k_{44},k_{53}
$I_1\backslash\{3,18\}$	k_{43},k_{70}	$I_1\backslash\{19,44\}$	$k_{10},k_{19},k_{28},k_{37},k_{46},k_{55}$
$I_1\backslash\{3,60\}$	k_{43},k_{47},k_{70}	$I_1\backslash\{33,44,66\}$	$k_6,k_{10},k_{11},k_{19},k_{20},$ $k_{28},k_{38},k_{47},k_{55},k_{72}$
$I_1\backslash\{0,15,41\}$	k_{24},k_{37},k_{56}	$I_1\backslash\{19,33,44\}$	$k_8,k_{17},k_{26},k_{57},k_{59}$
$I_1\backslash\{42,44,46\}$	k_{50},k_{51}	$I_1\backslash\{44,53,67\}$	$k_{11},k_{20},k_{29},k_{47}$
$I_1\backslash\{3,19,44\}$	k_{44},k_{56},k_{71}	$I_1\backslash\{3,29\}$	k_{18}
$I_1\backslash\{34,41,44\}$	k_6,k_{49}	$I_1\backslash\{3,14,41\}$	k_{22}
$I_1\backslash\{36,58\}$	k_7,k_{48}	$I_1\backslash\{31\}$	$k_2,k_{24},k_{29},k_{47},$ $k_{49},k_{53},k_{56},k_{74}$
$I_1\backslash\{36,43\}$	k_{33}	$I_1\backslash\{14,33,44\}$	k_3,k_6,k_{30}
$I_1\backslash\{43,62,67\}$	k_{55}	$I_1\backslash\{31,67\}$	$k_3,k_{10},k_{12},k_{19},k_{21},$ $k_{27},k_{28},k_{66},k_{75}$
$I_1\backslash\{0,2\}$	k_{34},k_{47}		

variables iteratively. A toy example is given to illustrate this iterative deduction in [14]. Firstly, we need to enumerate the values of 45 variables:

$$\{k_0,k_1,k_2,k_3,k_8,k_9,k_{10},k_{11},k_{13},k_{14},k_{15},k_{16},k_{17},k_{19},k_{21},k_{27},k_{28},$$
$$k_{32},k_{36},k_{37},k_{39},k_{40},k_{41},k_{45},k_{50},k_{52},k_{54},k_{57},k_{59},k_{61},k_{63},k_{64},k_{65},$$
$$k_{66},k_{67},k_{68},k_{69},k_{70},k_{71},k_{72},k_{73},k_{76},k_{77},k_{78},k_{79}\},$$

and the complexity is 2^{45}. For each enumeration, the values of the remaining 35 variables can be deduced iteratively in the order:

$$(k_{23},k_{35},k_{48},k_{49},k_{56},k_{58},k_{60},k_{62},k_{43},k_{47},k_{24},k_{51},k_{44},k_6,k_7,k_{33},k_{55},$$
$$k_{34},k_{42},k_{38},k_4,k_{12},k_{31},k_5,k_{25},k_{53},k_{46},k_{20},k_{26},k_{29},k_{18},k_{22},k_{74},k_{30},k_{75}),$$

and this deduction only costs constant time. The total attack complexity is $2^{47}+2^{45}$. On a PC with two RTX3090 GPUs, we mount a practical key-recovery attack within six hours. Specifically, we use two GPUs to obtain the corresponding values of all cubes in 3.2 hours and then use one GPU to guess and enumerate all possible keys in about 2 hours. Finally, we successfully obtain the 80-bit key. Comparing 808-round Trivium attack in [14], we increase the number of attacked rounds by seven by adding only three IV variables. This also shows that our search algorithm is more efficient.

4.2 A practical key-recovery attack on 820-round Trivium

In this subsection, we target 820-round Trivium. Firstly, we construct a set of indexes I_2 of size 52, where

$$I_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 25, 26, 28, 29,$$

$$30, 31, 32, 34, 36, 37, 41, 43, 44, 46, 49, 52, 53, 55, 56, 59, 61, 62, 64, 66, 68, 72, 74, 76, 79\}.$$

Using the search algorithm in Section 3, we obtain over 60 balanced superpolies by searching some subcubes. In searching for valuable subcubes, we find that most of the recovered linear terms are repeated, that is, there are very few subcubes left after secondary filtering. Based on this discovery, we think that for the mother cube, most of the independent secret variables contained in its subcubes are obtained. It is not worthwhile to continue searching subcubes for the few possible independent secret variables. However, by selecting superpolies and using the enumeration method described in Section 4.1, we can only recover about the values of 20 variables. Therefore, in order to obtain some new independent secret variables, we construct another set of indexes I_3 of size 52, where

$$I_3 = \{0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30,$$

$$31, 32, 34, 36, 38, 39, 40, 41, 42, 46, 49, 51, 52, 53, 54, 55, 58, 61, 63, 66, 69, 72, 74, 76, 78\}.$$

Since some simple balanced superpolies have been obtained in I_2, we filter out a lot of low-degree subcubes with linear terms in I_3. Finally, we obtain more than 100 balanced superpolies containing 54 different independent secret variables using I_2 and I_3. By selecting 30 balanced superpolies, we can linearize the equation system and perform a key-recovery attack on 820-round Trivium. These selected valuable cubes and their corresponding independent secret variables are listed in Table 4, and the specific superpolies can be found on the GitHub website.

First, for I_2 and I_3, it takes 2^{53} requests to obtain all the values of these 30 superpolies. Next, we need to enumerate the values of 50 variables:

$$\{k_0, k_1, k_4, k_5, k_6, k_7, k_9, k_{11}, k_{12}, k_{15}, k_{16}, k_{17}, k_{18}, k_{19}, k_{21}, k_{22}, k_{23}, k_{24},$$

$$k_{28}, k_{30}, k_{31}, k_{32}, k_{33}, k_{34}, k_{35}, k_{37}, k_{38}, k_{40}, k_{41}, k_{42}, k_{44}, k_{45}, k_{46}, k_{47}, k_{48},$$

$$k_{49}, k_{50}, k_{52}, k_{57}, k_{59}, k_{62}, k_{64}, k_{67}, k_{68}, k_{69}, k_{71}, k_{73}, k_{76}, k_{77}, k_{78}\},$$

and the complexity is 2^{50}. For each enumeration, the values of the remaining 30 variables can be deduced iteratively in the order:

$$(k_{55}, k_{61}, k_{63}, k_{51}, k_{43}, k_{27}, k_{56}, k_{58}, k_{79}, k_{25}, k_{53}, k_{54}, k_{70}, k_{39},$$

$$k_{29}, k_2, k_{36}, k_{10}, k_{72}, k_{26}, k_{13}, k_{14}, k_{60}, k_{65}, k_{74}, k_3, k_{75}, k_8, k_{20}, k_{66}),$$

and this deduction only costs constant time. The total attack complexity is $2^{53} + 2^{50}$. Because this calculation is the same as that in Section 4.1, we estimate that the attack on 820-round Trivium could be completed in two weeks on the same computer.

Table 4. Valuable cubes for attacking 820-round Trivium

Cube indices	Independent bits	Cube indices	Independent bits
$I_2\backslash\{66\}$	k_{55}	$I_3\backslash\{2,14\}$	k_2, k_{29}
$I_2\backslash\{4,7,62\}$	k_{61}	$I_3\backslash\{38,54\}$	k_{36}
$I_2\backslash\{3,13\}$	k_{63}	$I_2\backslash\{4,20,44\}$	k_{10}, k_{37}
$I_2\backslash\{3,26\}$	k_{51}, k_{78}	$I_3\backslash\{51\}$	k_{45}, k_{72}
$I_2\backslash\{3,62,66\}$	k_{43}, k_{51}, k_{78}	$I_2\backslash\{17,53,62,68\}$	$k_{12}, k_{26}, k_{27}, k_{39}, k_{54}$
$I_3\backslash\{6,42,52\}$	k_{27}	$I_3\backslash\{18,31\}$	k_{13}, k_{40}
$I_3\backslash\{7,9\}$	k_{56}	$I_3\backslash\{18,24,31\}$	k_{14}, k_{41}, k_{68}
$I_3\backslash\{7,58\}$	k_{58}	$I_2\backslash\{1,31\}$	k_{33}, k_{60}
$I_3\backslash\{3,7,17\}$	k_{52}, k_{79}	$I_2\backslash\{5,62,66\}$	k_{38}, k_{65}
$I_3\backslash\{1,3,32\}$	k_{25}, k_{52}	$I_3\backslash\{23,38\}$	k_{27}, k_{47}, k_{74}
$I_2\backslash\{6,29\}$	k_{53}	$I_3\backslash\{23,52\}$	k_3, k_{63}
$I_3\backslash\{18,54\}$	k_{27}, k_{54}	$I_3\backslash\{13,63\}$	k_{48}, k_{75}
$I_3\backslash\{27,53\}$	k_{43}, k_{58}, k_{70}	$I_3\backslash\{21,29,55,68\}$	k_8
$I_2\backslash\{53,68\}$	k_{12}, k_{39}, k_{63}	$I_2\backslash\{14,29,37,68\}$	k_{20}
$I_3\backslash\{6,18,29\}$	k_{29}	$I_2\backslash\{6,21,29\}$	k_{66}

5 Conclusion

In this paper, we focus on full key-recovery attacks on Trivium. A cube leading to a special kind of balanced superpoly is called a valuable cube. We present a novel framework to efficiently search for valuable cubes in cube attacks so that many balanced superpolies can be collected. As applications, two attacks on 815- and 820-round Trivium are given with time complexity $2^{47.32}$ and $2^{53.17}$, respectively. It is experimentally verified that the two attacks could be completed in six hours and two weeks on a PC with two RTX3090 GPUs, respectively. Although the key recovery process is practical, it seems unpractical to collect so many keystream bits required in our attacks during online communication. Hence, we call our attacks on 815- and 820-round Trivium experimentally verified attacks. Since the idea of this new framework to search for valuable cubes is generic in cube attacks, we believe that it is also helpful in cube attacks on other NFSR-based cryptosystems.

When analyzing Trivium with some large number of rounds, e.g., 845, recovering only a linear term of a superpoly is already time-consuming because of the large round number and large cube size. In this case, it is infeasible to sieve several subcubes. Hence, targeting Trivium with more than 845 rounds is worthy of working on in the future.

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. Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *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*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, 2015.
4. 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.
5. 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.
6. SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. Milp-aided method of searching division property using three subsets and applications. In *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, pages 398–427, 2019.
7. 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.
8. Chen-Dong Ye and Tian Tian. Revisit division property based cube attacks: Key-recovery or distinguishing attacks? *IACR Trans. Symmetric Cryptol.*, 2019(3):81–102, 2019.
9. Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 466–495. Springer, 2020.
10. Christophe De Cannière and Bart Preneel. Trivium. In *New Stream Cipher Designs - The eSTREAM Finalists*, pages 244–266. 2008.
11. Chen-Dong Ye and Tian Tian. A practical key-recovery attack on 805-round trivium. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 187–213. Springer, 2021.
12. Kai Hu, Siwei Sun, Meiqin Wang, and Qingju Wang. An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea,*

December 7-11, 2020, Proceedings, Part I, volume 12491 of *Lecture Notes in Computer Science*, pages 446–476. Springer, 2020.

13. Kai Hu, Siwei Sun, Yosuke Todo, Meiqin Wang, and Qingju Wang. Massive superpoly recovery with nested monomial predictions. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 392–421. Springer, 2021.
14. Yao Sun. Automatic search of cubes for attacking stream ciphers. *IACR Trans. Symmetric Cryptol.*, 2021(4):100–123, 2021.
15. 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.
16. 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.
17. 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.
18. Chen-Dong Ye and Tian Tian. Algebraic method to recover superpolies in cube attacks. *IET Information Security*, 14(4):430–441, 2020.
19. Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset. *IACR Cryptol. ePrint Arch.*, 2020:441, 2020.
20. 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.
21. 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.