More Balanced Polynomials: Cube Attacks on 810- and 825-Round Trivium with Practical Complexities

Hao Lei[1,2], Jiahui He[1,2], Kai Hu[3], and Meiqin Wang[1,2,4] (✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
leihao@mail.sdu.edu.cn,hejiahui2020@mail.sdu.edu.cn,mqwang@sdu.edu.cn
[2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China.
[3] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. kai.hu@ntu.edu.sg
[4] Quan Cheng Shandong Laboratory, Jinan, China

Abstract. The key step of the cube attack is to recover the special polynomial, the superpoly, of the target cipher. In particular, the balanced superpoly, in which there exists at least one secret variable as a single monomial and none of the other monomials contain this variable, can be exploited to reveal one-bit information about the key bits. However, as the number of rounds grows, it becomes increasingly difficult to find such balanced superpolies. Consequently, traditional methods of searching for balanced superpolies soon hit a bottleneck. Aiming at performing a cube attack on more rounds of Trivium with a practical complexity, in this paper, we present three techniques to obtain sufficient balanced polynomials.

1. Based on the structure of Trivium, we propose a variable substitution technique to simplify the superpoly.
2. Obtaining the additional balanced polynomial by combining two superpolies to cancel the two-degree terms.
3. We propose an experimental approach to construct high-quality large cubes which may contain more subcubes with balanced superpolies and a heuristic search strategy for their subcubes whose superpolies are balanced.

To illustrate the power of our techniques, we search for balanced polynomials for 810- and 825-round Trivium. As a result, we can mount cube attacks against 810- and 825-round Trivium with the time complexity of $2^{44.17}$ and $2^{53.17}$ round-reduced Trivium initializations, respectively, which can be verified in 48 minutes and 18 days on a PC with one A100 GPU. For the same level of time complexity, this improves the previous best results by 2 and 5 rounds, respectively.

Keywords: Trivium · cube attack · key-recovery attack · division property.

1 Introduction

The cube attack, proposed by Dinur and Shamir at EUROCRYPT 2009 [8], is one of the most powerful cryptanalysis techniques for symmetric ciphers. It has been successfully used to attack various stream ciphers such as Kreyvium, Acorn and Trivium [6,27,5,24,19,3].

Trivium was designed by Cannière and Preneel, as a bit-oriented synchronous stream cipher which was selected as one of the eSTREAM hardware-oriented finalists, and the international standard under ISO/IEC 29192-3:2012 [5]. Trivium has attracted extensive attention because of its simple structure and high level of security. Since the cube attack was proposed, it has become one of the most effective cryptanalytic techniques to analyze the reduced-round variants of Trivium. Currently, the best cube attacks on Trivium are those enhanced by divsion-properties [24], which have reached 848 rounds [12] but with an impractical complexity that is very close to the exhaustive search.

At the same time, the cube attacks on reduced-round Trivium with practical complexities also attract much attention, for the practical attacks have the potential to reveal more internal structural properties of Trivium and inspire new techniques for cube attacks. In [8], the authors proposed the random walk method to attack 767-round Trivium with about 2^{45} initializations. Next, Fouque et al. found many cubes with linear superpolies by improving the time complexity of computing cubes, then the 784-round of Trivium could be attacked with about 2^{39} initializations [9]. Later, in [31], Ye et al. proposed an effective method to construct cubes for linear superpolies and they gave a practical attack against 805-round Trivium with about $2^{41.4}$ initializations. At FSE 2021, Sun proposed a new heuristic method to reject cubes without independent secret variables and they could perform practical attacks against 806- and 808-round Trivium with time complexity of $2^{39.88}$ and $2^{44.58}$ initializations, respectively [22]. Recently, Cheng et al. gave attacks on 815- and 820-round Trivium with $2^{47.32}$ and $2^{53.17}$ initializations, respectively [7]. These results are summarized in Table 1.

1.1 Our Contributions

This paper focuses on practical key-recovery attacks against reduced-round Trivium. In order to attack a higher number of rounds, we propose the following methods.

Simplify Superpolies by Variable Substitutions. At the cost of adding one extra variable, the variable substitution can greatly simplify the superpoly, so that some unbalanced superpolies can be transformed into balanced ones. Thanks to this technique, more simple and balanced superpolies are utilized, from which more information about secret variables can be extracted.

More Balanced Polynomials by Canceling the Quadratic Terms. Balanced superpolies are important for practical attacks on round-reduced Trivium. However, for Trivium with a higher number of rounds, it is hard to find cubes

with balanced superpolies. We propose a method to obtain one additional balanced polynomial by canceling the quadratic terms in two superpolies. The new balanced polynomial is the sum of these two superpolies.

A Modified Algorithm to Construct a Better Mother Cube. Finding a large cube that contains many subcubes with balanced superpolies is important for the practical attack on round-reduced Trivium. In the following paper, this large cube is called a mother cube. By examining the relationship between the superpoly of a mother cube and the superpolies of its subcubes, we can more accurately judge whether a mother cube is suitable for a practical attack.

A Heuristic Strategy for Searching for Balanced Subcubes. It often takes a long time to recover the superpoly of a cube for high rounds of Trivium. Moreover, a high-dimensional mother cube has a large number of subcubes. To reduce the search space, we propose two strategies for dividing the search space by examining the relationship between the superpoly of a cube and the superpolies of its subcubes, which allows us to obtain sufficient balanced superpolies for a practical key-recovery attack with the reduced search space.

As an application, we apply our methods to 810-round Trivium and 825-round Trivium. The complexities of the cube attacks on 810 and 825 rounds of Trivium are respectively $2^{44.58}$ and $2^{53.09}$ round-reduced Trivium initializations. We list our attacks as well as the previous practical/theoretical cube attacks on Trivium for a better comparison in Table 1. We also implemented these two attacks on a PC with an A100 GPU. The experimental results showed that the whole keys can be recovered within 48 minutes and 18 days for 810 and 825 rounds of Trivium, respectively.

All source codes for our algorithms in this paper are provided at the git repository https://github.com/lhoop/ObtainMoreBS.

1.2 Outline

In Section 2, some related concepts and definitions are introduced. An overview of previous works on round-reduced practical attacks on Trivium is given in Appendix C. In Section 3, we propose three techniques to obtain a lot of balanced polynomials for round-reduced Trivium and a heuristic search strategy that reduces the cube search space of round-reduced Trivium. In Section 4, we mount cube attacks against 810- and 825-round Trivium by our techniques. Finally, we draw our conclusions in Section 5.

2 Preliminaries

2.1 Notation

We use bold italic lowercase letters to represent bit vectors, such as $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ where x_i is the i-th element of $\boldsymbol{x}$. For any n-bit vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we

Table 1. A summary of cube attacks on Trivium

Type	# of rounds	Cube size	# of key bits	Total time	Ref.
Practical	672	12	63	$2^{18.56}$	[8]
	709	22-23	79	$2^{29.14}$	[18]
	767	28-31	35	$2^{45.00}$	[8]
	784	30-33	42	$2^{39.00}$	[9]
	805	32-38	42	$2^{41.40}$	[31]
	806	33-37	45	$2^{39.88}$	[22]
	808	39-41	37	$2^{44.58}$	[22]
	810	**40-42**	**39**	$\mathbf{2^{44.17}}$	**Section** 4.1
	815	44-46	35	$2^{47.32}$	[7]
	820	48-51	30	$2^{53.17}$	[7]
	825	**49-52**	**31**	$\mathbf{2^{53.09}}$	**Section** 4.2
Theoretical	799	32-37	18	$2^{62.00}$	[9]
	802	34-37	8	$2^{72.00}$	[29]
	805	28	7	$2^{73.00}$	[17]
	806	34-37	16	$2^{64.00}$	[31]
	835	35	5	$2^{75.00}$	[17]
	832	72	1	$2^{79.01}$	[26]
	832	72	> 1	$< 2^{79.01}$	[30]
	840	78	1	$2^{79.58}$	[10]
	840	75	3	$2^{77.32}$	[15]
	841	78	1	$2^{79.58}$	[10]
	841	76	2	$2^{78.58}$	[15]
	842	78	1	$2^{79.58}$	[11]
	842	76	2	$2^{78.58}$	[15]
	843	54-57,76	5	$2^{76.58}$	[14]
	843	78	1	$2^{79.58}$	[22]
	844	54-55	2	$2^{78.00}$	[14]
	845	54-55	2	$2^{78.00}$	[14]
	846	51-54	1	$2^{79.00}$	[12]
	847	52-53	1	$2^{79.00}$	[12]
	848	51-54	1	$2^{79.00}$	[12]

define $\boldsymbol{u} \succeq \boldsymbol{v}$ if $u_i \geq v_i$ for all $0 \leq i < n$. Similarly, we define $\boldsymbol{u} \preceq \boldsymbol{v}$ if $u_i \leq v_i$ for all $0 \leq i < n$. Blackboard bold uppercase letters (e.g. $\mathbb{X}, \mathbb{K}, \mathbb{L}, \dots$) are used to represent sets of bit vectors. And we use the composition operator ($\circ$) to compose two functions. For example, $g \circ f(x) = g(f(x))$.

2.2 Boolean Functions and Algebraic Degree

Boolean Function. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function whose algebraic normal form (ANF) is

$$f(\boldsymbol{x}) = f(x_0, x_1, \dots, x_{n-1}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \prod_{i=0}^{n-1} x_i^{u_i},$$

where $a_{\boldsymbol{u}} \in \mathbb{F}_2$, and

$$\boldsymbol{x}^{\boldsymbol{u}} = \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{n-1} x_i^{u_i} \text{ with } x_i^{u_i} = \begin{cases} x_i, & \text{if } u_i = 1, \\ 1, & \text{if } u_i = 0, \end{cases}$$

is called a **monomial**. We use the notation $\boldsymbol{x}^{\boldsymbol{u}} \rightarrow f$ to indicate that the coefficient $a^{\boldsymbol{u}}$ of $\boldsymbol{x}^{\boldsymbol{u}}$ in f is 1, i.e., $\boldsymbol{x}^{\boldsymbol{u}}$ appears in f. Otherwise, $\boldsymbol{x}^{\boldsymbol{u}} \nrightarrow f$. In this work, we will use $\boldsymbol{x}^{\boldsymbol{u}}$ and $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ interchangeably to avoid the awkward notation $\boldsymbol{x}^{(i)\boldsymbol{u}^{(j)}}$ when both $\boldsymbol{x}$ and $\boldsymbol{u}$ have superscripts.

One important feature of a Boolean function is its algebraic degree which is denoted by $\deg(f)$ and defined as

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

where $wt(\boldsymbol{u})$ is the Hamming weight of $\boldsymbol{u}$, i.e., $wt(\boldsymbol{u}) = \sum_{i=0}^{n-1} u_i$.

Vectorial Boolean Function. Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ be a vectorial Boolean function with $\boldsymbol{y} = (y_0, y_1, \ldots, y_{m-1}) = \boldsymbol{f}(\boldsymbol{x}) = (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$. For $\boldsymbol{v} \in \mathbb{F}_2^m$, we use $\boldsymbol{y}^{\boldsymbol{v}}$ to denote the product of some coordinates of $\boldsymbol{y}$:

$$\boldsymbol{y}^{\boldsymbol{v}} = \prod_{i=0}^{m-1} y_i^{v_i} = \prod_{i=0}^{m-1} (f_i(\boldsymbol{x}))^{v_i},$$

which is a Boolean function in $\boldsymbol{x}$.

2.3 Pseudo-code of Trivium

Trivium is a bit oriented synchronous stream cipher, the main building block of Trivium is a 288-bit nonlinear feedback shift register which is divided into three small registers. For initialization, the 80 bit secret variables $\boldsymbol{k} = (k_0, k_1, \ldots, k_{79})$ is loaded into the first register, and the 80-bit IV (i.e., public variables) $\boldsymbol{v} = (v_0, v_1, \ldots, v_{79})$ is loaded into the second register. For the third register, all state bits are set to 0 except the last three bits. After 1152 state updates, Trivium starts to output keystream bits. The pseudo-code of Trivium is described in Algorithm 1.

2.4 Cube Attack

The cube attack was first proposed by Dinur and Shamir in EUROCRYPT 2009 [8]. It is a powerful cryptanalytic technique against stream ciphers. For a cipher with n public variables and m secret variables, each output bit of the cipher can be represented as a polynomial in these secret and public variables. Let z be an output bit, $\boldsymbol{x} = (x_0, x_1, ..., x_{n-1})$ as the public variables and $\boldsymbol{k} = (k_0, k_1, ..., k_{m-1})$ as the secret variables. z can be expressed as

$$z = f(\boldsymbol{k}, \boldsymbol{x}).$$

Algorithm 1: Pseudo-code of Trivium

$s_0, s_1, ..., s_{92} \leftarrow (k_0, ..., k_{79}, 0, ..., 0)$
$s_{93}, s_{94}, ..., s_{176} \leftarrow (v_0, ..., v_{79}, 0, ..., 0)$
$s_{177}, s_{178}, ..., s_{287} \leftarrow (0, ..., 0, 1, 1, 1)$
for *i from 1 to N (Number of initialization rounds)* **do**
  $ta \leftarrow s_{65} \oplus s_{92} \oplus s_{90}s_{91} \oplus s_{170}$
  $tb \leftarrow s_{161} \oplus s_{176} \oplus s_{174}s_{175} \oplus s_{263}$
  $tc \leftarrow s_{242} \oplus s_{287} \oplus s_{285}s_{286} \oplus s_{68}$
  if $i > 1152$ **then**
    $z_{i-1152} \leftarrow s_{65} \oplus s_{92} \oplus s_{161} \oplus s_{176} \oplus s_{242} \oplus s_{287}$
  end
  $s_0, s_1, ..., s_{92} \leftarrow (tb, s_0, s_1, ..., s_{91})$
  $s_{93}, s_{94}, ..., s_{176} \leftarrow (ta, s_{93}, s_{94}, ..., s_{175})$
  $s_{177}, s_{178}, ..., s_{287} \leftarrow (tc, s_{177}, s_{178}, ..., s_{286})$
end

Let $I = \{x_{c_1}, x_{c_2}, \ldots, x_{c_d}\} \subset \{x_0, x_1, \ldots, x_{n-1}\}$ be a subset of public variables, assume $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{x \in I} x$ is its corresponding term. Then, $f(\boldsymbol{x}, \boldsymbol{k})$ can be uniquely expressed as

$$f(\boldsymbol{x}, \boldsymbol{k}) = p(\boldsymbol{x}, \boldsymbol{k}) \cdot \boldsymbol{x}^{\boldsymbol{u}} + q(\boldsymbol{x}, \boldsymbol{k}),$$

where $q(\boldsymbol{x}, \boldsymbol{k})$ misses at least one variable in I. A **cube** determined by I is denoted as C_I and contains all 2^d possible combinations of the values of variables in I. C_I is a d-dimensional cube because the size of I is d. $\boldsymbol{x}^{\boldsymbol{u}}$ is called a **cube term**. The public variables in I are called cube variables and the remaining public variables are called non-cube variables. $p(\boldsymbol{x}, \boldsymbol{k})$ is called the **superpoly** of C_I which can be computed by

$$\bigoplus_{\boldsymbol{x} \in C_I} f(\boldsymbol{x}, \boldsymbol{k}) = p(\boldsymbol{x}, \boldsymbol{k}).$$

Therefore, we can get the value of the superpoly $p(\boldsymbol{x}, \boldsymbol{k})$ by $2^{|I|}$ calls to the initialization oracle.

For a superpoly P, if a variable k_i that appears only in a one-degree monomial, we say P is a **balanced superpoly** for the **balanced variable** k_i. Moreover, if all variables in P are balanced variables, we say P is a **linear superpoly**. For example, $P_1 = k_1 \oplus k_2 \oplus k_3k_4$ is a balanced superpoly for balanced variables k_1 and k_2, $P_2 = k_1 \oplus k_2$ is a linear superpoly. We say a cube is **balanced** if its superpoly is balanced for some balanced variables. For a balanced superpoly whose value is known, we can deduce its one balanced variable by enumerating the values of other variables.

2.5 The Bit-Based Division Property and Monomial Prediction

The division property is a generalization of integral property, which was proposed by Todo in [23,25]. The conventional bit-based division property can be used to evaluate the algebraic degree of a cube and the three-subset division property

without unknown subset can be used to recover superpoly. The definitions of the conventional bit-based division property and three-subset division property are provided in Appendix A.

The monomial prediction, proposed by Hu et al. in [15], is another language of division property from a pure algebraic perspective. By counting the so-called monomial trails, the monomial prediction can determine if a monomial of IV appears in the polynomial of the output of the cipher.

Let $\boldsymbol{f} : \mathbb{F}_2^{n_0} \to \mathbb{F}_2^{n_r}$ be a composite vectorial Boolean function of a sequence of r smaller function $\boldsymbol{f}^{(i)} : \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_{i+1}}, 0 \le i \le r-1$ as

$$\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \cdots \circ \boldsymbol{f}^{(0)}.$$

Let $\boldsymbol{x}^{(i)} \in \mathbb{F}_2^{n_i}$ and $\boldsymbol{x}^{(i+1)} \in \mathbb{F}_2^{n_{i+1}}$ be the input and output variables of $\boldsymbol{f}^{(i)}$, respectively. Suppose $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ is a monomial of $x^{(0)}$, it is easy to find all monomials of $x^{(1)}$ satisfying $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)})$. For every such $\pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)})$, we then find all the $\pi_{\boldsymbol{u}^{(2)}}(\boldsymbol{x}^{(2)})$ satisfying $\pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)}) \to \pi_{\boldsymbol{u}^{(2)}}(\boldsymbol{x}^{(2)})$. Finally, if we are interested in whether $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, we collect some transitions from $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ to $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ as

$$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)}) \to \cdots \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}).$$

Every such transition is called a monomial trail from $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, denoted by $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. All the trails from $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ to $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ are denoted by $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, which is the set of all trails. Then whether $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ is determined by the size of $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, represented as $|\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})|$. If there is no trail from $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ to $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, we say $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ and accordingly $|\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})| = 0$.

Theorem 1 (Integrated from [10,13,15,11]). *Let* $\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \cdots \circ \boldsymbol{f}^{(0)}$ *defined as above.* $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ *if and only if*

$$|\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})| \equiv 1 \pmod 2.$$

Propagation Rules of the Monomial Prediction. Any component of a symmetric cipher can be regarded as a vectorial Boolean function. To model the propagation of the monomial prediction for a vectorial Boolean function, a common method is to list all the possible (input, output) tuples according to the definition of the monomial prediction [15]. These tuples can be transformed into a set of linear inequalities and thus modeled with MILP [21,20,4]. Since any symmetric cipher can be decomposed into a sequence of the basic operations XOR, AND and COPY. We provide their concrete propagation rules for these basic functions and MILP models in Appendix B.

3 Obtain More Balanced Polynomials

To mount a practical key-recovery attack on high rounds of Trivium, we need a lot of small cubes whose superpolies are simple and balanced. However, small cubes for high rounds of Trivium tend to have complex and unbalanced superpolies.

In order to obtain more simple and balanced polynomials, we propose a method based on the variable substitution technique to simplify the superpolies and a method to obtain more balanced polynomials by canceling some quadratic terms between superpolies. We introduce these two methods in Sections 3.1 and 3.2, respectively. Additionally, according to an observation, we modify the algorithm in [31, Section 3.1] to obtain a mother cube containing more subcubes whose superpolies are balanced and provide a heuristic search strategy to make a trade-off between the search space and the search quality. They are introduced in Section 3.3.

3.1 Variable Substitutions

The Principle of Variable Substitutions. Notice that after the state of Trivium was updated 24 times, the last 11 bits of the secret variables will only exist in the state in a specific form.

$$(s_0, s_1, \ldots, s_{92}) \leftarrow (k_0, k_1, \ldots, k_{79}, 0, \ldots, 0);$$

$$(s_{93}, s_{94}, \ldots, s_{176}) \leftarrow (v_0, v_1, \ldots, v_{79}, 0, \ldots, 0);$$

$$(s_{177}, s_{178}, \ldots, s_{287}) \leftarrow (0, \ldots, 0, 1, 1, 1).$$

After 24 rounds of updates,

$$(s'_0, s'_1, \ldots, s'_{92}) \leftarrow (tb_{23}, tb_{22}, \ldots, tb_0, k_0, \ldots, k_{68});$$

$$(s'_{93}, s'_{94}, \ldots, s'_{176}) \leftarrow (ta_{23}, ta_{22}, \ldots, ta_0, v_0, \ldots, v_{59});$$

$$(s'_{177}, s'_{178}, \ldots, s'_{287}) \leftarrow (tc_{23}, tc_{22}, \ldots, tc_0, 0, \ldots, 0),$$

where ta_i, tb_i, tc_i are three bits updated by the round function of the i-th round of Trivium.
From the expressions of $s'_0, s'_1, ..., s'_{287}$, we find that $k_{69}, k_{70}, \ldots, k_{79}$ are only involved in $s'_{94}, s'_{95}, .., s'_{105}$.

$$s'_{94} = ta_{23} = k_{42} \oplus k_{69} \oplus k_{68}k_{67} \oplus v_{54},$$

$$s'_{95} = ta_{22} = k_{43} \oplus k_{70} \oplus k_{69}k_{68} \oplus v_{55},$$

$$\vdots$$

$$s'_{105} = ta_{12} = k_{53} \oplus 0 \oplus k_{79}k_{78} \oplus v_{65}.$$

We use new variables $p_{69}, \ldots, p_{80}$ to replace these polynomials of secret variables.

$$p_{69} = k_{42} \oplus k_{69} \oplus k_{68}k_{67},$$

$$p_{70} = k_{43} \oplus k_{70} \oplus k_{69}k_{68},$$

$$\vdots$$

$$p_{80} = k_{53} \oplus 0 \oplus k_{79}k_{78}.$$

For the sake of clarity, we also use $p_0, \ldots, p_{68}$ to replace the other secret variables ($p_0 = k_0, p_1 = k_1, \ldots, p_{68} = k_{68}$).

Then all state bits of round 24 are only related to the 81 new secret variables $\boldsymbol{p} = (p_0, p_1, \ldots, p_{80})$ and the 80 public variables $\boldsymbol{v} = (v_0, v_1, \ldots, v_{79})$. $(s'_0, \ldots, s'_{287})$ can be written as the output of a vectorial Boolean function of $\boldsymbol{p}$ and $\boldsymbol{v}$, namely,

$$\boldsymbol{y} = (s'_0, \ldots, s'_{287}) = \boldsymbol{g}(\boldsymbol{v}, \boldsymbol{p}) = (g_0(\boldsymbol{v}, \boldsymbol{p}), g_1(\boldsymbol{v}, \boldsymbol{p}), \ldots, g_{287}(\boldsymbol{v}, \boldsymbol{p})).$$

And the output bit after r rounds ($r > 24$) can be expressed as follows,

$$z^r = h(\boldsymbol{s'}) \circ \boldsymbol{g}(\boldsymbol{v}, \boldsymbol{p}),$$

where $h(\boldsymbol{s'})$ is a Boolean function from $(s'_0, \ldots, s'_{287})$ to z^r.

This implies that we can represent the superpoly as a polynomial of only $\boldsymbol{p}$. Since polynomials of degree 2 in $\boldsymbol{k}$ (e.g., $k_{42} + k_{69} + k_{68}k_{67}$) are now replaced by single variables (e.g., p_{69}), the $\boldsymbol{p}$-representation of the superpoly is likely to have a lower algebraic degree and thus, is simpler.

Therefore, after representing superpolies as polynomials of $\boldsymbol{p}$, more simple and balanced superpolies may be obtained. We show this in Example 1.

Example 1. Let $A = k_{42} \oplus k_{69} \oplus k_{68}k_{67}$ and $B = k_{45} \oplus k_1(k_{45} \oplus k_{72} \oplus k_{71}k_{70})$ be a balanced superpoly and an unbalanced superpoly of $\boldsymbol{k}$, respectively. After variable substitutions, we have $A = p_{69}$, $B = p_{45} \oplus p_1 p_{72}$, which means A becomes a linear superpoly of $\boldsymbol{p}$ and B becomes a balanced superpoly of $\boldsymbol{p}$.

To illustrate the power of this technique, we search for balanced superpolies among the same set of cubes for 825-round Trivium before and after variable substitutions, respectively. We evaluate the simplicity of each superpoly by the number of variables it contains and record the number of occurrences of balanced superpolies at different levels of simplicity. See Table 2 for details.

Table 2. Distribution of balanced superpolies (B.S.) involving different numbers of variables.

# variables involved	#B.S. of $\boldsymbol{k}$	#B.S. of $\boldsymbol{p}$
≤ 5	34	42
≤ 10	42	67
≤ 20	90	122
≤ 40	187	235
≤ 60	275	318
≤ 80	322	354

Substituting Variables Back. The variable substitution technique simplifies superpolies at the cost of adding one extra variable, i.e., p_{80}. Notice that the

values of $k_0, \ldots, k_{79}$ can be derived from the values of $p_0, \ldots, p_{79}$ as follows,

$$\begin{aligned} p_0, \ldots, p_{68} &\Rightarrow k_0, \ldots, k_{68}, \\ p_{69}, k_{42}, k_{67}, k_{68} &\Rightarrow k_{69}, \\ p_{70}, k_{43}, k_{68}, k_{69} &\Rightarrow k_{70}, \\ &\vdots \\ p_{79}, k_{52}, k_{78}, k_{77} &\Rightarrow k_{79}. \end{aligned}$$

Therefore, p_{80} is actually redundant and can be expressed as a polynomial of $p_0, \ldots, p_{79}$. In other words, $f : (k_0, \ldots, k_{79}) \rightarrow (p_0, \ldots, p_{79})$ is actually a bijective function.

In this paper, we always consider the superpolies represented as polynomials of $\boldsymbol{p}$. Once the values of all 81 new secret variables (i.e., $p_0, \ldots, p_{80}$) are obtained, we first derive the values of $k_0, \ldots, k_{79}$ from $p_0, \ldots, p_{79}$, then check whether the value of p_{80} is correct. The concrete process of substituting $\boldsymbol{p}$ back to obtain $\boldsymbol{k}$ is shown in Algorithm 2.

Algorithm 2: Recovering $\boldsymbol{k}$ from $\boldsymbol{p}$

Input: $(p_0, p_1, \ldots, p_{80})$
Output: $(k_0, k_1 \ldots, k_{79})$ or $false$

for *i from 0 to 68* **do**
  $k_i \leftarrow p_i$;
for *i from 69 to 79* **do**
  $k_i \leftarrow p_i \oplus k_{i-27} \oplus k_{i-1}k_{i-2}$;
$check \leftarrow k_{53} \oplus k_{78}k_{79}$;
if $p_{80} == check$ **then**
  return $(k_0, k_1 \ldots, k_{79})$
else
  return *false*

3.2 More Balanced Polynomials by Canceling Quadratic Terms

After constructing a mother cube, we can obtain a large number of subcubes, but only a very small fraction of subcubes have balanced superpolies. More balanced polynomials are required when we perform the practical attack on high rounds of Trivium. In order to obtain more balanced polynomials, we provide an algorithm to obtain additional balanced polynomials by canceling the common quadratic terms between superpolies. First, we define two types of superpolies.

Definition 1 (Type-I Superpoly). *If there exists a single monomial p_i and a quadratic monomial p_ip_j, and none of the other monomials contain p_i, then we call this superpoly a type-I superpoly of a variable p_i.*

Definition 2 (Type-II Superpoly). *If there exists a variable p_i that appears in a quadratic monomial $p_i p_j$, and none of the other monomials contain this variable, then we call this superpoly a type-II superpoly of a variable p_i.*

For example, a superpoly $A = p_{73} \oplus p_{73}p_{25}$ is a type-I superpoly of a variable p_{73} and a superpoly $B = p_{73}p_{25}$ is a type-II superpoly of variables p_{73} and p_{25}.

Combining Rule. For a specific variable p_i, if we can find both a type-I superpoly of p_i and a type-II superpoly of p_i and the quadratic monomials containing p_i in these two superpolies are the same, then we can get a balanced polynomial by adding these two superpolies. For example, we can obtain a balanced polynomial C with p_{73} being a balanced variable by adding the type-I superpoly A and the type-II superpoly B,

$$C = A \oplus B = p_{73} \oplus p_{73}p_{25} \oplus p_{73}p_{25} = p_{73}.$$

In the superpolies recovery process, we use SageMath [2] to determine whether a superpoly is a type-I superpoly or a type-II superpoly.

For 825-round Trivium, we find 540 cubes with type-I superpolies or type-II superpolies. By the combining rule, we obtained 782 additional balanced polynomials. This increases our number of balanced polynomials from 451 to 1323.

3.3 Relationship Between a Cube and its Subcubes

In our practical cube attacks on Trivium, we focus on balanced superpolies rather than superpolies of low degrees.

For example, consider two superpolies A and B.

$$A : p_1p_2 \oplus p_2p_3 = c_1,$$

$$B : p_1 \oplus p_3p_{10}p_4p_6 = c_2.$$

The degrees of A and B are 2 and 4, respectively. We prefer B to A because B is balanced so that we can use it to deduce p_1 by enumerating p_3, p_{10}, p_4, p_6.

Inspiration. In the search process, we obtain a set of cubes whose superpolies are estimated to have low degrees based on the division property, but the superpolies of these cubes are almost all unbalanced. While for another set of cubes, we find that a lot of their superpolies are balanced, although their superpolies are estimated to have high degrees. This inspires us to investigate how to locate the balanced superpolies more accurately. With experiments, we have the following observation.

Observation 1 *For a given x-dimensional cube I, if its superpoly is balanced, then the superpolies of its $(x-1)$-dimensional subcubes have a greater probability of being balanced.*

We present some data in Appendix D to support this observation. Based on this observation, we propose a method for constructing a better mother cube that may contain more subcubes with balanced superpolies and a heuristic search method to reduce the search space.

A Modified Algorithm to Construct a Mother Cube for Balanced Superpolies. We modify the algorithm for constructing the mother cube in [31, Section 3.1] to obtain a better mother cube that may have more subcubes with balanced superpolies.

First, at the stage of determining the starting cube, Ye et al. predicted a preference bit $s_\lambda^{(r)}$ for r-round Trivium. $s_\lambda^{(r)}$ can be written as

$$s_\lambda^{(r)} = s_{i_1^\lambda}^{(r-\lambda)} \cdot s_{i_2^\lambda}^{(r-\lambda)} \oplus s_{i_3^\lambda}^{(r-\lambda)} \oplus s_{i_4^\lambda}^{(r-\lambda)} \oplus s_{i_5^\lambda}^{(r-\lambda)},$$

it have five state bits from $(r-\lambda)$-round Trivium. Next, $s_{i_1^\lambda}^{(r-\lambda)}$ and $s_{i_2^\lambda}^{(r-\lambda)}$ are the dominant parts in determining whether $s_\lambda^{(r)}$ contributes to linear terms in the superpoly. Therefore, the starting cube should have a linear superpoly in $s_{i_1^\lambda}^{(r-\lambda)}$ or $s_{i_2^\lambda}^{(r-\lambda)}$. We are less strict about the starting cube because we focus on balanced superpolies, so our starting cube only need to have a simple balanced superpoly in $s_{i_1^\lambda}^{(r-\lambda)}$ or $s_{i_2^\lambda}^{(r-\lambda)}$.

Second, at the stage of extending the starting cube, we will obtain a lot of candidate mother cubes after the expansion. The criteria for selection from candidate mother cubes by Tian et al. is choosing the mother cube which has the most subcubes of degree less than 5 [7]. However, it may not be accurate enough to judge whether superpoly is more likely to be balanced by its degree. We propose a more accurate way to determine the best mother cube from the candidate mother cubes, i.e., the x-dimensional candidate mother cube that contains the most $(x-1)$-dimensional subcubes with simple balanced superpolies is selected.

If an x-dimensional mother cube contains most $(x-1)$-dimensional subcubes with simple balanced superpolies. Then by Observation 1, the superpolies of their $(x-2)$-dimensional subcubes have a greater probability of being balanced and so do the superpolies of for $(x-3)$- and $(x-4)$-dimensional subcubes.

As an example, for 825-round Trivium, we select

$$\begin{aligned} I_2 = \{&p_3, p_4, p_5, p_8, p_9, p_{10}, p_{16}, p_{17}, p_{20}, p_{22}, p_{25}, p_{26}, p_{28}, p_{29}, p_{32}, p_{34}, p_{37}, \\ &p_{39}, p_{40}, p_{43}, p_{44}, p_{45}, p_{46}, p_{47}, p_{48}, p_{49}, p_{51}, p_{52}, p_{53}, p_{54}, p_{55}, p_{56}, p_{57}, p_{58}, \\ &p_{59}, p_{60}, p_{61}, p_{62}, p_{64}, p_{66}, p_{67}, p_{69}, p_{70}, p_{71}, p_{72}, p_{74}, p_{76}, p_{78}, p_{79}, p_{80}\} \end{aligned}$$

as our mother cube. The size of I_2 is 53 and it has 7 52-dimensional subcubes with simple balanced superpolies. Among the subcubes of these 7 52-dimensional subcubes, there are 44 51-dimensional subcubes with balanced superpolies. Further more, 87 50-dimensional subcubes with balanced superpolies can be discovered from these 44 51-dimensional subcubes.

A Heuristic Search Strategy for Searching Balanced Cubes. At ASIACRYPT 2022 [12], He et al. proposed an efficient method based on the core monomial prediction to recover the superpolies for up to 848 rounds of Trivium. We utilize this method as a tool to search for balanced subcubes (i.e., subcubes

with balanced superpolies) among the subcubes of a mother cube by directly recovering the superpolies of subcubes.

According to our experiments, if the superpoly of a subcube is complex, it often takes a long time to recover it. Therefore, when recovering the superpoly for a specific subcube, we set a time limit. If the superpoly of a subcube is not recovered within the time limit, we consider the superpoly to be complex and discard this subcube. This simple trick speeds up the search for balanced subcubes of a mother cube. However, for a 53-dimensional mother cube, it has 53 52-dimensional subcubes, 1378 51-dimensional subcubes, and 23426 50-dimensional subcubes. Recovering the superpolies for these subcubes one by one is still impossible in a reasonable time, but reducing the search space for subcubes means some balanced superpolies may be missed.

To deal with this problem, we give two heuristic strategies based on Observation 1 to reject potentially useless subcubes in advance.

- **First strategy.** If an x-dimensional cube is determined to have an unbalanced and non-zero superpoly, all $(x-1)$-dimensional subcubes of this cube are considered to have unbalanced superpolies and are rejected.
- **Second strategy.** If an x-dimensional cube is determined to have a balanced or zero superpoly, all $(x-1)$-dimensional subcubes of this cube are considered to possibly have balanced superpolies and will be checked.

The difference between the first strategy and the second strategy is that when an $(x-1)$-dimensional cube is simultaneously the subcube of several x-dimensional cubes and one of these x-dimensional cubes has an unbalanced and non-zero superpoly, then the $(x-1)$-dimensional cube will be rejected according to the first strategy, but will be checked according to the second strategy. In other words, the first strategy is more aggressive in narrowing down the search space.

To further illustrate the effect of reducing the search space on the number of balanced superpolies, we give specific experimental data using the first strategy, the second strategy, and no strategy (i.e., the search over the whole $(x-1)$-dimensional subcubes, called full search) in Table 3.

For an x-dimensional mother cube, we usually search over its $(x-1)$- and $(x-2)$-dimensional subcubes by the full search, then the second strategy is used to search over its $(x-3)$-dimensional subcubes and $(x-4)$-dimensional subcubes. Finally, we use the first strategy to search for $(x-5)$-dimensional subcubes. These search strategies help us find a sufficient number of balanced superpolies with an acceptable computational effort.

4 Application

In this section, we apply our improved methods to 810- and 825-round Trivium. Due to the page limit, superpolies used in this section are provided at https://github.com/lhoop/ObtainMoreBS/tree/main/data.

Table 3. A comparison between search space and balanced cubes for 50-dimensional subcubes of a 53-dimensional mother cube A_0 for 825-round Trivium, where $A_0 = \{v_2, v_5, v_8, v_{10}, v_{12}, v_{15}, v_{17}, v_{19}, v_{23}, v_{29}, v_{31}, v_{41}, v_{44}, v_{46}, v_{51}, v_{55}, v_{63}, v_{66}, v_{72}, v_{78}, v_3, v_0, v_{69}, v_6, v_{26}, v_7, v_{50}, v_{68}, v_{25}, v_{48}, v_{33}, v_4, v_{21}, v_{76}, v_{36}, v_{16}, v_{14}, v_{37}, v_{38}, v_{39}, v_{59}, v_{61}, v_{18}, v_{53}, v_{34}, v_{74}, v_{40}, v_1, v_{57}, v_9, v_{13}, v_{22}, v_{35}\}$

Method	Search space	# of balanced cubes	Space rate†	Balanced cubes rate‡
first strategy	1365	63	5.83%	40.38%
second strategy	12201	156	52.1%	96.30%
no strategy	23426	162	100%	100%

†: Space rate = (search space)/(the whole search space).
‡: Balanced cubes rate = (balanced cubes)/(balanced cubes from the whole search space).

4.1 A Key-Recovery Attack on 810-Round Trivium with Practical Complexity

Determine a Mother Cube. Using the algorithm in [31], we predict $s_{66}^{(810)}$ as the preference bit for 810-round Trivium. We choose many cubes of size 19 and search for cubes whose superpolies in $s_{285}^{(744)}$ are balanced. Then, the cube

$$Sa_1 = \{v_2, v_6, v_8, v_{10}, v_{11}, v_{15}, v_{19}, v_{21}, v_{25}, v_{29}, v_{30}, v_{32}, v_{34}, v_{36}, v_{39}, v_{41}, v_{43}, v_{45}, v_{50}\}$$

is obtained. Its superpoly is p_{56}. Also, we search for cubes whose superpolies in $s_{286}^{(744)}$ are balanced. We get the cube

$$Sa_2 = \{v_0, v_2, v_4, v_8, v_{10}, v_{11}, v_{17}, v_{19}, v_{25}, v_{29}, v_{30}, v_{32}, v_{34}, v_{36}, v_{39}, v_{41}, v_{43}, v_{45}, v_{50}\}.$$

Its superpoly is p_{56}. We first choose Sa_1 as the starting cube to extend.

The public variables are added to Sa_1 iteratively to make the degree[1] of the superpoly decrease to a minimum value other than 0. Then Sa_3 is obtained.

$$Sa_3 = \{v_2, v_6, v_8, v_{10}, v_{11}, v_{15}, v_{19}, v_{21}, v_{25}, v_{29}, v_{30}, v_{32}, v_{34}, v_{36}, v_{39}, v_{41}, v_{43}, v_{45}, v_{50}, v_0, v_{75}, v_{12}, v_4, v_{14}, v_{20}, v_{22}, v_{16}, v_{27}, v_{23}, v_{72}, v_{52}, v_{55}, v_{60}, v_{37}, v_{79}, v_{62}, v_{64}, v_{47}, v_{54}, v_{70}\}.$$

The upper bound of the degree of its superpoly is 8 and its size is 40. We note that this set contains all elements in Sa_2 except v_{17}. So we replace v_{16} in Sa_3 with v_{17} to make it fully contain Sa_2.

$$Sa_3' = \{v_2, v_6, v_8, v_{10}, v_{11}, v_{15}, v_{19}, v_{21}, v_{25}, v_{29}, v_{30}, v_{32}, v_{34}, v_{36}, v_{39}, v_{41}, v_{43}, v_{45}, v_{50}, v_0, v_{75}, v_{12}, v_4, v_{14}, v_{20}, v_{22}, v_{17}, v_{27}, v_{23}, v_{72}, v_{52}, v_{55}, v_{60}, v_{37}, v_{79}, v_{62}, v_{64}, v_{47}, v_{54}, v_{70}\}.$$

[1] We evaluate the upper bound of the degree of the superpoly based on the division property modeled with MILP.

And after adding one element of set $A = \{v_{40}, v_{53}, v_{57}, v_{58}, v_{67}, v_{68}, v_{77}, v_{48}\}$, the degree of Sa'_3 is less than 5. Then, we select four elements from set A to add to Sa'_3 and get 70 44-dimensional candidate mother cubes. We examine the number of balanced superpolies in the 43-dimensional subcubes for each of the 70 candidate mother cubes, then the mother cube

$$\begin{aligned} Sa_4 = \{&v_0, v_2, v_4, v_6, v_8, v_{10}, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{19}, v_{20}, v_{21}, v_{22}, v_{23}, v_{25}, v_{27}, \\ &v_{29}, v_{30}, v_{32}, v_{34}, v_{36}, v_{37}, v_{39}, v_{41}, v_{43}, v_{45}, v_{47}, v_{50}, v_{52}, v_{54}, v_{55}, v_{60}, v_{62}, \\ &v_{64}, v_{70}, v_{72}, v_{75}, v_{79}, v_{68}, v_{57}, v_{53}, v_{48}\} \end{aligned}$$

is selected. It has two simple 43-dimensional subcubes whose superpolies are linear.

Search for Balanced Subcubes. We add a time limit to the search time. A full search is performed over all 42- and 43-dimensional subcubes of Sa_4. The 40- and 41-dimensional subcubes are searched using the second strategy. The method used to recover superpolies is from [12], and we modify it to recover superpolies for new secret variables. Then several superpolies are obtained. After using SageMath to extract balanced or quadratic superpolies, 405 balanced superpolies and 526 quadratic superpolies are obtained. Then we obtain additional 275 balanced polynomials from 526 quadratic superpolies using the combining rule in Section 3.2.

Determine the Order of Derivation. We pick 39 polynomials from 680 balanced polynomials. The corresponding cubes and the independent bits contained by these polynomials are listed in Table 4. 39 variables can be deduced from these polynomials. The specific polynomials are provided at https://github.com/lhoop/ObtainMoreBS/tree/main/data/810_superpoly.

A Practical Attack on a PC. The size of Sa_4 is 44, so it takes 2^{44} requests to obtain all the values of these 39 polynomials. Next, we need to enumerate the values of 42 variables: $\{p_0, p_2, p_3, p_4, p_6, p_7, p_9, p_{10}, p_{17}, p_{18}, p_{20}, p_{23}, p_{25}, p_{28}, p_{30}, p_{33}, p_{35}, p_{38}, p_{39}, p_{40}, p_{42}, p_{44}, p_{45}, p_{47}, p_{48}, p_{49}, p_{51}, p_{52}, p_{56}, p_{57}, p_{58}, p_{59}, p_{60}, p_{63}, p_{66}, p_{69}, p_{70}, p_{73}, p_{77}, p_{78}, p_{79}, p_{80}\}$.

For each enumeration, the values of the remaining 39 variables can be deduced iteratively in the order $(p_{76}, p_{61}, p_{64}, p_{74}, p_{62}, p_{41}, p_{46}, p_{11}, p_{37}, p_{34}, p_{21}, p_{22}, p_{54}, p_{24}, p_{50}, p_{12}, p_{36}, p_{19}, p_{65}, p_5, p_{29}, p_8, p_{16}, p_{53}, p_{26}, p_{14}, p_{43}, p_{68}, p_{55}, p_{67}, p_{71}, p_{27}, p_{75}, p_{31}, p_{15}, p_1, p_{32}, p_{13}, p_{72})$.

There are 2^{42} enumerations, for each enumeration, we use Algorithm 2 to substitute new secret variables back to original secret variables. Half of the enumerations will be excluded because the value of p_{80} does not match. This check only costs constant time. So actually, there are only 2^{41} enumerations of original secret variables. With 2^{41} round-reduced Trivium initializations, the correct key can be filtered out of these 2^{41} candidate keys. To sum up, the whole attack costs $2^{44} + 2^{41}$ round-reduced Trivium initializations. On a PC with an A100 GPU, we can perform the whole attack in 48 minutes.

4.2 A Key-Recovery Attack on 825-Round Trivium with Practical Complexity

Determine a Mother Cube. We predict $s_{66}^{(825)}$ as the preference bit for 825-round Trivium. Then we choose many cubes of sizes 20 and search for cubes whose superpolies in $s_{286}^{(759)}$ are balanced. Finally, the cube

$$\begin{aligned} Sb_1 = \{&v_2, v_5, v_6, v_8, v_{10}, v_{12}, v_{15}, v_{19}, v_{23}, v_{29}, v_{34}, v_{41}, v_{44}, v_{46}, v_{53}, v_{55}, v_{63}, \\ &v_{66}, v_{72}, v_{78}\} \end{aligned}$$

is selected. Its superpoly is $p_{66} \oplus p_{24} \oplus p_{22}p_{23} \oplus 1$. The public variables are added to Sb_1 iteratively to make the degree of the superpoly decrease to a minimum value other than 0. Then Sb_2 is obtained.

$$\begin{aligned} Sb_2 = \{&v_2, v_5, v_8, v_{10}, v_{12}, v_{15}, v_{17}, v_{19}, v_{23}, v_{29}, v_{31}, v_{41}, v_{44}, v_{46}, v_{51}, v_{55}, v_{63}, v_{66}, \\ &v_{72}, v_{78}, v_3, v_0, v_{69}, v_6, v_{26}, v_7, v_{50}, v_{68}, v_{25}, v_{48}, v_{33}, v_4, v_{21}, v_{76}, v_{36}, v_{16}, \\ &v_{14}, v_{37}, v_{38}, v_{39}, v_{59}, v_{61}, v_{18}, v_{53}, v_{34}, v_{74}, v_{40}, v_1, v_{57}, v_9\}. \end{aligned}$$

The upper bound of the degree of its superpoly is 6 and its size is 50. And after adding one variable of set $B = \{v_{40}, v_{53}, v_{57}, v_{58}, v_{67}, v_{68}, v_{77}, v_{48}\}$, the degree of Sb_2 is less than 5. Then we select three variables from set B to add to Sb_2 and obtain 280 53-dimensional candidate mother cubes. For each of the 280 candidate mother cubes, we examine the number of balanced superpolies that can be generated by its 52-dimensional subcubes, then the mother cube

$$\begin{aligned} Sb_3 = \{&v_2, v_5, v_8, v_{10}, v_{12}, v_{15}, v_{17}, v_{19}, v_{23}, v_{29}, v_{31}, v_{41}, v_{44}, v_{46}, v_{51}, v_{55}, v_{63}, v_{66}, \\ &v_{72}, v_{78}, v_3, v_0, v_{69}, v_6, v_{26}, v_7, v_{50}, v_{68}, v_{25}, v_{48}, v_{33}, v_4, v_{21}, v_{76}, v_{36}, v_{16}, \\ &v_{14}, v_{37}, v_{38}, v_{39}, v_{59}, v_{61}, v_{18}, v_{53}, v_{34}, v_{74}, v_{40}, v_1, v_{57}, v_9, v_{13}, v_{22}, v_{35}\} \end{aligned}$$

is selected. It has 7 52-dimensional subcubes whose superpolies are simple and balanced.

Search for Balanced Subcubes. A full search is performed over all 52- and 51-dimensional subcubes of Sb_3. The 50- and 49-dimensional subcubes are searched using the second strategy, respectively. Then several superpolies are obtained. We use SageMath to extract balanced or quadratic superpolies and obtain 354 balanced superpolies and 422 quadratic superpolies. Then we obtain an extra 872 balanced polynomials from 422 quadratic superpolies using the combining rule in Section 3.3.

Determine the Order of Derivation. We pick 31 polynomials from 1226 balanced polynomials. The corresponding cubes and the independent bits contained by these polynomials are listed in Table 5. 31 variables can be deduced from these polynomials. The specific polynomials are provided at https://github.com/lhoop/ObtainMoreBS/tree/main/data/825_superpoly.

A Practical Attack on a PC. The size of Sb_3 is 53, so it takes 2^{53} requests to obtain all the values of these 31 polynomials. Next, we need to enumerate the values of 50 variables: $\{p_3, p_4, p_5, p_8, p_9, p_{10}, p_{16}, p_{17}, p_{20}, p_{22}, p_{25}, p_{26}, p_{28}, p_{29}, p_{32}, p_{34}, p_{37}, p_{39}, p_{40}, p_{43}, p_{44}, p_{45}, p_{46}, p_{47}, p_{48}, p_{49}, p_{51}, p_{52}, p_{53}, p_{54}, p_{55}, p_{56}, p_{57}, p_{58}, p_{59}, p_{60}, p_{61}, p_{62}, p_{64}, p_{66}, p_{67}, p_{69}, p_{70}, p_{71}, p_{72}, p_{74}, p_{76}, p_{78}, p_{79}, p_{80}\}$.

For each enumeration, the values of the remaining 31 variables can be deduced iteratively in the order $(p_{73}, p_{75}, p_{77}, p_{63}, p_{14}, p_{31}, p_{11}, p_{35}, p_{27}, p_{33}, p_{12}, p_{41}, p_{30}, p_{65}, p_{38}, p_1, p_{13}, p_{15}, p_{50}, p_{42}, p_6, p_7, p_{18}, p_{68}, p_{24}, p_{23}, p_2, p_0, p_{19}, p_{36}, p_{21})$.

There are 2^{50} enumerations, for each enumeration, we use Algorithm 2 to substitute new secret variables back to original secret variables. Half of the enumerations will be excluded because the value of p_{80} does not match. This check only costs a constant time. So actually, we only need 2^{49} enumerations of original secret variables. With 2^{49} round-reduced Trivium initializations, the correct key can be filtered out of these 2^{49} candidate keys. To sum up, the whole attack costs $2^{53} + 2^{49}$ round-reduced Trivium initializations. On a PC with an A100 GPU, we can perform the whole attack in 18 days.

5 Conclusion

In this paper, we focus on practical full key-recovery attacks on Trivium. We propose a variable substitution technique to simplify the superpoly and a new method to obtain a new balanced polynomial by combining two superpolies to cancel out the quadratic terms. Moreover, by an observation that the subcubes of a cube whose superpoly is balanced are more likely to have balanced superpolies, we modify the original algorithm to construct a better mother cube that contains more subcubes with balanced superpolies, and propose a heuristic strategy for searching for cubes with balanced superpolies. As a result, we use our new methods to perform full key-recovery attacks on 810- and 825-round Trivium, which can be done with time complexity $2^{44.17}$ and $2^{53.09}$ round-reduced Trivium initializations, respectively. It is experimentally verified that the two attacks could be completed in 48 minutes and 18 days on a PC with one A100 GPU (128×256 threads), respectively. We also time-test previous attacks on 808- and 820-round Trivium with the same number of threads [22,7]. For the attack on 808-round Trivium in [22], it could be completed in 12 hours with our GPU. And for the attack on 820-round Trivium in [7], we estimate that the attack would be completed in 19 days with our GPU. These experiments confirm that we can improve the previous results for 2 and 5 rounds without increasing the time complexity.

Acknowledgment. The authors would like to thank Raghvendra Rohit as our shepherd and other anonymous reviewers that have helped us improve the quality of this paper. This research is supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 62032014), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025).

References

1. Gurobi Optimization. https://www.gurobi.com
2. Sagemath. https://www.sagemath.org
3. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Dunkelman, O. (ed.) Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5665, pp. 1–22. Springer (2009), https://doi.org/10.1007/978-3-642-03317-9_1
4. Boura, C., Coggia, D.: Efficient MILP modelings for sboxes and linear layers of SPN ciphers. IACR Trans. Symmetric Cryptol. **2020**(3), 327–361 (2020), https://doi.org/10.13154/tosc.v2020.i3.327-361
5. Cannière, C.D., Preneel, B.: Trivium. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs - The eSTREAM Finalists, LNCS, vol. 4986, pp. 244–266. Springer (2008), https://doi.org/10.1007/978-3-540-68351-3_18
6. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. J. Cryptol. **31**(3), 885–916 (2018), https://doi.org/10.1007/s00145-017-9273-9
7. Che, C., Tian, T.: An experimentally verified attack on 820-round trivium. In: Deng, Y., Yung, M. (eds.) Information Security and Cryptology - 18th International Conference, Inscrypt 2022, Beijing, China, December 11-13, 2022, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13837, pp. 357–369. Springer (2022), https://doi.org/10.1007/978-3-031-26553-2_19
8. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 278–299. Springer (2009), https://doi.org/10.1007/978-3-642-01001-9_16
9. Fouque, P., Vannet, T.: Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. IACR Cryptol. ePrint Arch. p. 312 (2015), http://eprint.iacr.org/2015/312
10. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against Trivium and Grain-128AEAD. In: Canteaut, A., Ishai, Y. (eds.) 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. Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer (2020), https://doi.org/10.1007/978-3-030-45721-1_17
11. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. J. Cryptol. **34**(3), 22 (2021), https://doi.org/10.1007/s00145-021-09383-2
12. He, J., Hu, K., Preneel, B., Wang, M.: Stretching cube attacks: Improved methods to recover massive superpolies. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 13794, pp. 537–566. Springer (2022), https://doi.org/10.1007/978-3-031-22972-5_19

13. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 537–566. Springer (2020), https://doi.org/10.1007/978-3-030-64837-4_18
14. Hu, K., Sun, S., Todo, Y., Wang, M., Wang, Q.: Massive superpoly recovery with nested monomial predictions. In: Tibouchi, M., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 13090, pp. 392–421. Springer (2021), https://doi.org/10.1007/978-3-030-92062-3_14
15. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 12491, pp. 446–476. Springer (2020), https://doi.org/10.1007/978-3-030-64837-4_15
16. Hu, K., Wang, M.: Automatic search for a variant of division property using three subsets. In: Matsui, M. (ed.) Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11405, pp. 412–432. Springer (2019), https://doi.org/10.1007/978-3-030-12612-4_21
17. Liu, M., Yang, J., Wang, W., Lin, D.: Correlation cube attacks: From weak-key distinguisher to key recovery. In: Nielsen, J.B., Rijmen, V. (eds.) 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. Lecture Notes in Computer Science, vol. 10821, pp. 715–744. Springer (2018), https://doi.org/10.1007/978-3-319-78375-8_23
18. Mroczkowski, P., Szmidt, J.: Corrigendum to: The cube attack on stream cipher Trivium and quadraticity tests. IACR Cryptol. ePrint Arch. p. 32 (2011), http://eprint.iacr.org/2011/032
19. Salam, M.I., Bartlett, H., Dawson, E., Pieprzyk, J., Simpson, L., Wong, K.K.: Investigating cube attacks on the authenticated encryption stream cipher Acorn. In: Batten, L., Li, G. (eds.) Applications and Techniques in Information Security - 6th International Conference, ATIS 2016, Cairns, QLD, Australia, October 26-28, 2016, Proceedings. Communications in Computer and Information Science, vol. 651, pp. 15–26 (2016), https://doi.org/10.1007/978-981-10-2741-3_2
20. Sasaki, Y., Todo, Y.: New algorithm for modeling s-box in MILP based differential and division trail search. In: Farshim, P., Simion, E. (eds.) SecITC 2017. LNCS, vol. 10543, pp. 150–165. Springer (2017), https://doi.org/10.1007/978-3-319-69284-5_11
21. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 158–178. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_9
22. Sun, Y.: Automatic search of cubes for attacking stream ciphers. IACR Trans. Symmetric Cryptol. **2021**(4), 100–123 (2021), https://doi.org/10.46586/tosc.v2021.i4.100-123

23. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) 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. Lecture Notes in Computer Science, vol. 9056, pp. 287–314. Springer (2015), https://doi.org/10.1007/978-3-662-46800-5_12
24. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 250–279. Springer (2017), https://doi.org/10.1007/978-3-319-63697-9_9
25. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 357–377. Springer (2016), https://doi.org/10.1007/978-3-662-52993-5_18
26. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: Galbraith, S.D., Moriai, S. (eds.) 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. Lecture Notes in Computer Science, vol. 11923, pp. 398–427. Springer (2019), https://doi.org/10.1007/978-3-030-34618-8_14
27. Wu, H.: Acorn v3. Submission to CAESAR competition (2016)
28. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) 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. Lecture Notes in Computer Science, vol. 10031, pp. 648–678 (2016), https://doi.org/10.1007/978-3-662-53887-6_24
29. Ye, C., Tian, T.: A new framework for finding nonlinear superpolies in cube attacks against Trivium-like ciphers. In: Susilo, W., Yang, G. (eds.) Information Security and Privacy - 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10946, pp. 172–187. Springer (2018), https://doi.org/10.1007/978-3-319-93638-3_11
30. Ye, C., Tian, T.: Algebraic method to recover superpolies in cube attacks. IET Inf. Secur. **14**(4), 430–441 (2020), https://doi.org/10.1049/iet-ifs.2019.0323
31. Ye, C., Tian, T.: A practical key-recovery attack on 805-round Trivium. In: Tibouchi, M., Wang, H. (eds.) 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. Lecture Notes in Computer Science, vol. 13090, pp. 187–213. Springer (2021), https://doi.org/10.1007/978-3-030-92062-3_7

Appendix

A The Definition of Bit-based Division Property

Definition 3 (Conventional Bit-Based Division Property [23]). *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 $D^{1^n}_{\mathbb{K}}$, it fulfills the following conditions:*

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

Definition 4 (Three-Subset Division Property [25]). *Let $\mathbb{X}$ be a multi-set whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ and $\mathbb{L}$ be two sets 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:*

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

B Propagation Rules of XOR, AND and COPY for Monomial Prediction

According to [10], the rules for XOR, AND and COPY are the following:

Rule 1 (XOR [10,11]) *Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (x_0 \oplus x_1, x_2, \ldots, x_{n-1})$ be the input and output vector of a* XOR *function. Considering a monomial of $\boldsymbol{x}$ as $\boldsymbol{x}^{\boldsymbol{u}}$, the monomials $\boldsymbol{y}^{\boldsymbol{v}}$ of $\boldsymbol{y}$ meet the condition that $\boldsymbol{x}^{\boldsymbol{u}} \rightarrow \boldsymbol{y}^{\boldsymbol{v}}$ only when $\boldsymbol{v}$ satisfies*

$$\boldsymbol{v} = (u_0 + u_1, u_2, \ldots, u_{n-1}), \quad (u_0, u_1) \in \{(0,0), (0,1), (1,0)\}.$$

Rule 2 (AND [10,11]) *Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (x_0 \vee x_1, x_2, \ldots, x_{n-1})$ be the input and output vector of an* AND *function. Considering a monomial of $\boldsymbol{x}$ as $\boldsymbol{x}^{\boldsymbol{u}}$, the monomials $\boldsymbol{y}^{\boldsymbol{v}}$ of $\boldsymbol{y}$ meet the condition that $\boldsymbol{x}^{\boldsymbol{u}} \rightarrow \boldsymbol{y}^{\boldsymbol{v}}$ only when $\boldsymbol{v}$ satisfies*

$$\boldsymbol{v} = (u_0, u_2, \ldots, u_{n-1}), \quad (u_0, u_1) \in \{(0,0), (1,1)\}.$$

Rule 3 (COPY [10,11]) *Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (x_0, x_0, x_1, x_2, \ldots, x_{n-1})$ be the input and output vector of a* COPY *function. Considering a monomial of $\boldsymbol{x}$ as $\boldsymbol{x}^{\boldsymbol{u}}$, the monomials $\boldsymbol{y}^{\boldsymbol{v}}$ of $\boldsymbol{y}$ meet the condition that $\boldsymbol{x}^{\boldsymbol{u}} \rightarrow \boldsymbol{y}^{\boldsymbol{v}}$ only when $\boldsymbol{v}$ satisfies*

$$\boldsymbol{v} = \begin{cases} (0, 0, u_2, \ldots, u_{n-1}), & \textit{if } u_0 = 0. \\ (0, 1, u_2, \ldots, u_{n-1}), (1, 0, u_2, \ldots, u_{n-1}), (1, 1, u_2, \ldots, u_{n-1}), & \textit{if } u_0 = 1. \end{cases}$$

MILP Models of the Propagation Trails. The propagation trails of the monomial prediction can be traced using the following MILP models:

Model 1 (XOR [10,11]) *Let* $(a_0, a_1, \ldots, a_{n-1}) \xrightarrow{\text{XOR}} b$ *be a propagation trail of* XOR*. The following inequalities suffice to describe all the valid trails for* XOR*:*

$$\begin{cases} \mathcal{M}.var \leftarrow a_0, a_1, \ldots, a_{n-1}, b \text{ as binary}; \\ \mathcal{M}.con \leftarrow b = a_0 + a_1 + \cdots + a_{n-1}. \end{cases}$$

Model 2 (AND [10,11]) *Let* $(a_0, a_1, \ldots, a_{n-1}) \xrightarrow{\text{AND}} b$ *be a propagation trail of* AND*. The following inequalities suffice to describe all the valid trails for* AND*:*

$$\begin{cases} \mathcal{M}.var \leftarrow a_0, a_1, \ldots, a_{n-1}, b \text{ as binary}; \\ \mathcal{M}.con \leftarrow b = a_i, \ \forall i \in \{0, 1, \ldots n-1\}. \end{cases}$$

Model 3 (COPY [10,11]) *Let* $a \xrightarrow{\text{COPY}} (b_0, b_1, \ldots, b_{n-1})$ *be a propagation trail of* COPY*. The following inequalities suffice to describe all the valid trails for* COPY*:*

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_0, b_1, \ldots, b_{n-1} \text{ as binary}; \\ \mathcal{M}.con \leftarrow b_0 + b_1 + \cdots + b_{n-1} \geq a; \\ \mathcal{M}.con \leftarrow a \geq b_i, \ \forall i \in \{0, 1, \ldots, n-1\}. \end{cases}$$

If the MILP solver supports the OR ($\vee$) *operation, then the model can also be represented by:*

$$\begin{cases} \mathcal{M}.var \leftarrow a, b_0, b_1, \ldots, b_{n-1} \text{ as binary}; \\ \mathcal{M}.con \leftarrow a = b_0 \vee b_1 \vee \cdots \vee b_{n-1}. \end{cases}$$

C Previous Practical Cube Attacks on Trivium

C.1 Construct a Mother Cube for Linear Superpolies [31]

At ASIACRYPT 2021, Ye et al. presented a heuristic algorithm for constructing a mother cube that may contain many subcubes with linear superpolies. Such a mother cube is very helpful for the practical attack on round-reduced Trivium because all values of superpolies of its subcubes can be recovered by $2^{|I|}$ Trivium initializations, where $|I|$ is the size of the mother cube. Specifically, the construction of a mother cube consists of two steps, namely determining a starting cube and then extending the starting cube.

Determine a Starting Cube Heuristically. Based on the structural analysis of Trivium, Ye et al. found that one of six internal state bits involved in the output function is more likely to contribute to linear terms in the superpoly of r-round Trivium. They called this bit the preference bit of r-round Trivium.

Definition 5 (The Preference Bit [31]). *Among the six internal state bits in the output function of r-round Trivium, the internal state bit which is most likely to contribute linear superpolies is called the preference bit of r-round Trivium.*

The preference bit can be determined by estimating the number of linear terms of $\boldsymbol{k}$ contained in the ANF of the six internal bits. For details, see [31].

After they predicted a preference bit $s_\lambda^{(r)}$ of r-round Trivium (the λ-th bit of the state), according to the update function of Trivium, $s_\lambda^{(r)}$ can be written as

$$s_\lambda^{(r)} = s_{i_1^\lambda}^{(r-\lambda)} \cdot s_{i_2^\lambda}^{(r-\lambda)} \oplus s_{i_3^\lambda}^{(r-\lambda)} \oplus s_{i_4^\lambda}^{(r-\lambda)} \oplus s_{i_5^\lambda}^{(r-\lambda)}.$$

It has five state bits from $(r-\lambda)$-round Trivium.

$s_{i_1^\lambda}^{(r-\lambda)}$ and $s_{i_2^\lambda}^{(r-\lambda)}$ are the dominant parts in determining whether $s_\lambda^{(r)}$ contributes to linear terms. Therefore, the starting cube is the cube which has a linear superpoly in $s_{i_1^\lambda}^{(r-\lambda)}$ or $s_{i_2^\lambda}^{(r-\lambda)}$.

Extend the Starting Cube by Greedy Strategies. The starting cube can be extended to a larger mother cube by adding some public variables that are not in the cube. The size of the mother cube can not be too large so that we can obtain the values of superpolies corresponding to its subcubes in practical time. On the other hand, the mother cube needs to contain enough low-degree subcubes that are more likely to have linear superpolies. In order to satisfy the above conditions, Ye et al. combined the division property based degree evaluation method with some greedy strategies to extend the starting cube. Their method has two stages. Before introducing these two stages, we first give the following definitions.

Definition 6 (Steep IV Variable [31]). *Let $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}\}$ be a set containing ℓ IV variables. Then, an IV variable $b \in B = \{v_0, v_1, \ldots, v_{n-1}\} \backslash I$ is called a steep IV variable of I if*

$$ds(I \cup \{b\}) = \min\{ds(I \cup \{v\}) \mid v \in B\},$$

where $ds(I)$ is the degree of the superpoly of I in secret variables.

Definition 7 (Gentle IV Variable [31]). *Let $I = \{v_{i_1}, v_{i_2}, \ldots, v_{i_\ell}\}$ be a set containing ℓ IV variables. Then, an IV variable $b \in B = \{v_0, v_1, \ldots, v_{n-1}\} \backslash I$ is called a gentle IV variable of I if*

$$ds(I \cup \{b\}) = \max\{ds(I \cup \{v\}) \mid ds(I \cup \{v\}) \leq ds(I), v \in B\},$$

where $ds(I)$ is the degree of the superpoly of I in secret variables.

Let I be a starting cube. In the first stage, the steep variables of I are added to I iteratively to make the degree of the superpoly decrease to a minimum value other than 0. Then in the second stage, the gentle IV variables are added to I iteratively to reduce the degree of the superpoly slowly. Finally, cubes with

degrees of their superpolies close to 1 are obtained and merged to construct the mother cube, namely

$$I \cup \{v_i \in \{v_0, v_1, \ldots, v_{n-1}\} \backslash I \mid \text{ the upper bound of } \mathrm{ds}(I \cup v_i) \text{ is close to } 1\},$$

where I is the cube before adding the last gentle IV variable.

For 805-round Trivium, by searching for the subcubes of these mother cubes, they found more than 1000 subcubes with linear superpolies and picked 37 cubes whose superpolies are linearly independent to perform the practical attack.

C.2 Attack by Balanced Superpolies [22]

The cubes with linear superpolies only account for a small fraction of all superpolies of Trivium, therefore, it is often difficult to obtain a sufficient number of linear superpolies for practical attacks. To overcome this bottleneck, Sun et al. further extended the linear superpoly to a more general concept, the balanced superpoly [22].

A superpoly is balanced if there exists at least one secret variable as a single monomial and none of the other monomials contain this variable in this superpoly, and this variable is called the balanced variable in the superpoly. If we know the value of a balanced superpoly, we can deduce the value of the balanced variable by enumerating the values of other variables.

For example, consider a superpoly $P_3 = k_1 \oplus k_2 k_3$. P_3 is a balanced superpoly with a balanced variable k_1. Assume that we know that the value of P_3 is 0, the value of k_1 can be obtained by enumerating the values of k_2 and k_3: $(k_2 = 0, k_3 = 0) \Rightarrow (k_1 = 0)$, $(k_2 = 0, k_3 = 1) \Rightarrow (k_1 = 0)$, $(k_2 = 1, k_3 = 0) \Rightarrow (k_1 = 0)$, $(k_2 = 1, k_3 = 1) \Rightarrow (k_1 = 1)$.

Note that if the value of a balanced variable is already deduced, it may be used to deduce another balanced variable. According to this rule, the values of many balanced variables in balanced superpolies can be deduced in order.

For 808-round Trivium, Sun et al. used the method in Appendix C.1 to construct a mother cube. Then they searched for its subcubes whose superpolies are balanced. Finally, they obtained many balanced superpolies and picked 37 balanced superpolies to perform a practical attack.

C.3 Construct a Mother Cube for Balanced Superpolies [7]

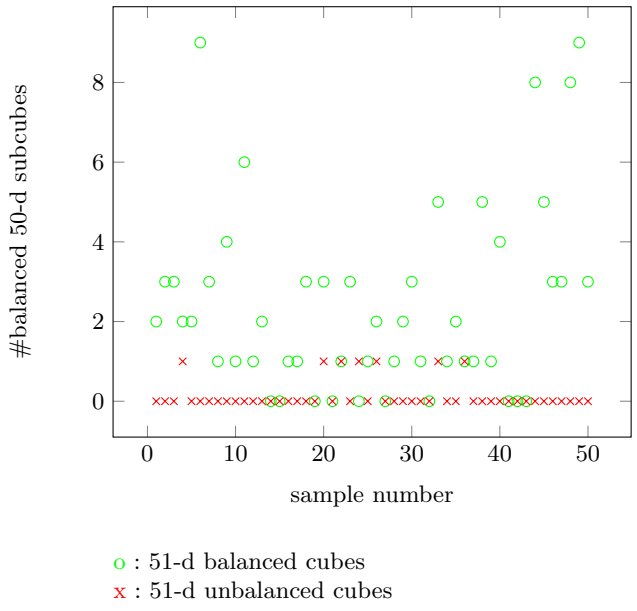
Last year, Che et al. modified a part of the algorithm in Appendix C.1 to construct a mother cube that may contain many subcubes with balanced superpolies. The process of determining the starting cube is the same. Because balanced polynomials do not need degrees to be close to 1, in the process of expanding a starting cube, they only used the first stage to reduce the degree of the cube quickly by adding steep IV variables. They wanted a mother cube with many subcubes whose degrees are less than 5. Assume I_a is the cube before adding the last steep IV variable, they constructed a set A which contained all v_i that satisfied (1) $v_i \in \{v_0, v_1, \ldots, v_{n-1}\} \backslash I_a$; (2) the upper bound of $\mathrm{ds}(I_a \cup v_i)$ is less than 5.

If all the variables in A are added to the I_a, the final mother cube will become too large. So they only chose some variables from the set A and added them to the I_a. Then they obtained many candidate mother cubes. For example, there would be 10 candidate mother cubes if three variables were chosen from a set A of size 5. One of the candidate cubes was selected as the mother cube if it had the most subcubes of degree less than 5.

D Some Experimental Data for Observation 1

For 825 rounds of Trivium, we selected fifty 51-dimensional balanced cubes and fifty 51-dimensional unbalanced cubes to test. And measured the number of its 50-dimensional balanced subcubes for each selected cube. The experimental results are shown in Figure 1.

Fig. 1. The number of balanced 50-dimensional subcubes for fifty 51-dimensional balanced cubes and fifty 51-dimensional unbalanced cubes.



E Balanced Cubes for Attacking 810- and 825-round Trivium

Table 4. Balanced cubes for attacking 810-round Trivium. Sorted by the deducing order. I_1 are the cube indices of Sa_4.

Cube indices	Independent bits	Cube indices	Independent bits
$I_1\backslash\{2,14,29,45\}$	p_{76}	$I_1\backslash\{0,60,64\}$	p_{24}, p_{29}, p_{66}
$I_1\backslash\{19,22,29,52\}$	p_{61}	$I_1\backslash\{12,15,30,39\}$	p_8, p_{35}
$I_1\backslash\{11,47,53,67\}$	p_{64}	$I_1\backslash\{34,37,62\}$	p_{16}
$I_1\backslash\{11,19\}$	p_{61}, p_{74}	$I_1\backslash\{4,11,54\}$	p_8, p_{52}, p_{53}
$I_1\backslash\{11,34,52,54\}$	p_{62}	$I_1\backslash\{0,36,55,64\}$	$p_{23}, p_{26}, p_{38}, p_{50}, p_{53}, p_{65}$
$I_1\backslash\{2,50,57\}$	p_{41}	$I_1\backslash\{11,19,54,60\}$	p_{14}, p_{23}
$I_1\backslash\{2,11,15,30\}$	p_{46}, p_{76}	$I_1\backslash\{0,17,48,54\}$	$p_{16}, p_{25}, p_{43}, p_{52}$
$I_1\backslash\{12,15,30,57\}$	p_{11}	$I_1\backslash\{0,47,79\}$	p_{68}
$I_1\backslash\{41,47,62\}$	p_{37}, p_{64}	$I_1\backslash\{0,25,62\}$	p_{52}, p_{55}
$I_1\backslash\{15,48,57,62\}$	p_{34}, p_{61}	$I_1\backslash\{34,62,67\}$	p_{40}, p_{55}, p_{67}
$I_1\backslash\{29,53,54,62\}$	p_{21}, p_{48}	$I_1\backslash\{29,34,75\}$ $+I_1\backslash\{34,52,55\}$	p_{71}
$I_1\backslash\{29,48,67,70\}$	p_{22}, p_{64}	$I_1\backslash\{23,25,47\}$	p_{27}, p_{29}, p_{63}
$I_1\backslash\{11,27,43,47\}$	p_{21}, p_{48}, p_{54}	$I_1\backslash\{4,11\}$	$p_3, p_{12}, p_{30} p_{75}$
$I_1\backslash\{23,47,55\}$	p_{24}, p_{66}	$I_1\backslash\{0,34,48,62\}$	p_4, p_{31}
$I_1\backslash\{0,37,54,79\}$	p_{23}, p_{50}	$I_1\backslash\{0,29,67,70\}$	$p_6, p_{12}, p_{15}, p_{38}, p_{39}, p_{42}$ $p_{51}, p_{60}, p_{74}, p_{75}, p_{78}$
$I_1\backslash\{0,20,62\}$	p_{12}, p_{39}	$I_1\backslash\{19,52,54,60\}$	p_1
$I_1\backslash\{34,48,62\}$	p_{36}, p_{63}	$I_1\backslash\{11,19,41,60\}$	p_{32}, p_{68}
$I_1\backslash\{2,14,57,60\}$	p_{19}, p_{46}	$I_1\backslash\{0,19\}$	$p_8, p_{13}, p_{17}, p_{24}, p_{32}, p_{35}$ p_{42}, p_{51}, p_{75}
$I_1\backslash\{11,30,62\}$	p_{65}	$I_1\backslash\{0,19,43\}$	p_{72}, p_{77}
$I_1\backslash\{0,29,55\}$	$p_5, p_{41}, p_{47}, p_{56}$		

$$I_1 = \{0,2,4,6,8,10,11,12,14,15,17,19,20,21,22,23,25,27,29,30,32,34,36,37,39,41,43,45,47,50,52,54,55,60,62,64,70,72,75,79,68,57,53,48\}.$$

Table 5. Balanced cubes for attacking 825-round Trivium. Sorted by the deducing order. I_2 are the cube indices of Sb_3.

Cube indices	Independent bits	Cube indices	Independent bits
$I_2\backslash\{0,18\}$	p_{73}	$I_2\backslash\{9,15\}$	$p_{13}, p_{22}, p_{40}, p_{49}, p_{61}, p_{73}$
$I_2\backslash\{0,35,41\}$	p_{75}	$I_2\backslash\{17,26\}$	$p_{15}, p_{60}, p_{63}, p_{73}$
$I_2\backslash\{7,13,35\}$	p_{77}	$I_2\backslash\{10,15\}$	p_{50}, p_{71}
$I_2\backslash\{25,35,50\}$	p_{63}	$I_2\backslash\{25,34,38,63\}$	p_{29}, p_{42}, p_{69}
$I_2\backslash\{15,69\}$	p_{14}, p_{77}	$I_2\backslash\{0,19,22\}$	p_6, p_{15}, p_{57}
$I_2\backslash\{0,1,38\}$	p_{31}	$I_2\backslash\{13,19,39\}$	p_7, p_{34}
$I_2\backslash\{2\}$	p_{11}	$I_2\backslash\{8,13,19\}$	p_{18}, p_{45}
$I_2\backslash\{15,23,34,36\}$	p_{35}, p_{62}, p_{77}	$I_2\backslash\{17,19,22\}$	p_{68}
$I_2\backslash\{17,22,25,69\}$	p_{27}	$I_2\backslash\{13,39,66\}$	p_{24}
$I_2\backslash\{21,22\}$ $+I_2\backslash\{4,17,51\}$	p_{33}	$I_2\backslash\{1,13,51,53\}$ $+I_2\backslash\{4,17,33,51\}$	p_{23}
$I_2\backslash\{15,23\}$	p_{12}, p_{63}	$I_2\backslash\{1,13,17,38\}$	$p_2, p_4, p_7, p_{23}, p_{50}, p_{74}$
$I_2\backslash\{17,25\}$	p_{41}, p_{75}	$I_2\backslash\{25,63,69\}$	$p_0, p_{13}, p_{50}, p_{51}, p_{69}, p_{71}$ p_{72}, p_{78}
$I_2\backslash\{9,15,17\}$	p_3, p_{30}	$I_2\backslash\{0,3,34\}$	p_1, p_{19}, p_{43}
$I_2\backslash\{7,22,63\}$	p_{65}	$I_2\backslash\{13,15,39\}$	$p_6, p_{23}, p_{27}, p_{30}, p_{36}, p_{64}$
$I_2\backslash\{33,63,69\}$	p_{38}	$I_2\backslash\{17,26,33\}$	p_{21}
$I_2\backslash\{17,26,41\}$	p_1		

$I_2 = \{2, 5, 8, 10, 12, 15, 17, 19, 23, 29, 31, 41, 44, 46, 51, 55, 63, 66, 72, 78, 3, 0, 69, 6, 26, 7, 50, 68, 25, 48, 33, 4, 21, 76, 36, 16, 14, 37, 38, 39, 59, 61, 18, 53, 34, 74, 40, 1, 57, 9, 13, 22, 35\}$.

F MILP Model for New Secret Variables after Variable Substitute

Algorithm 3: MILP model for new secret variables $(p_0, \ldots, p_{80})$

Procedure TriviumCore($\mathcal{M}, \mathrm{x}_0, \ldots, \mathrm{x}_{287}, \mathrm{i}_1, \mathrm{i}_2, \mathrm{i}_3, \mathrm{i}_4, \mathrm{i}_5$):
  $\mathcal{M}.var \leftarrow \mathrm{y}_{i_1}, \mathrm{y}_{i_2}, \mathrm{y}_{i_3}, \mathrm{y}_{i_4}, \mathrm{y}_{i_5}, \mathrm{z}_1, \mathrm{z}_2, \mathrm{z}_3, \mathrm{z}_4$ as binary
  $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{a} = \mathrm{z}_3$
  $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{a} = \mathrm{z}_4$
  $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{y}_{\mathrm{i}_5} = \mathrm{x}_{\mathrm{i}_5} + \mathrm{a} + \mathrm{z}_1 + \mathrm{z}_2$
  $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{x}_{\mathrm{i}_\mathrm{j}} = \mathrm{y}_{\mathrm{i}_\mathrm{j}} \vee \mathrm{z}_\mathrm{j}$ for all $\mathrm{j} \in \{1, 2, 3, 4\}$
  for $i \in \{0, 1, \ldots, 287\}$ *without* i_1, i_2, i_3, i_4, i_5 **do**
    $\mathrm{y}_\mathrm{i} = \mathrm{x}_\mathrm{i}$
  return $(\mathcal{M}, \mathrm{y}_0, \mathrm{y}_1, \ldots, \mathrm{y}_{287})$
Procedure TriviumCorep($\mathcal{M}, \mathrm{x}_0, \ldots, \mathrm{x}_{287}, \mathrm{p}$):
  $\mathcal{M}.var \leftarrow \mathrm{y}_{170}, \mathrm{z}$ as binary
  $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{x}_{170} = \mathrm{y}_{170} \vee \mathrm{z}$
  $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{y}_{92} = \mathrm{z} + \mathrm{p}$
  for $i \in \{0, 1, \ldots, 287\}$ *without* $92, 170$ **do**
    $\mathrm{y}_\mathrm{i} = \mathrm{x}_\mathrm{i}$
  return $(\mathcal{M}, \mathrm{y}_0, \mathrm{y}_1, \ldots, \mathrm{y}_{287})$
Procedure ModelNewVariables(*round* R, *cube indices* I):
  Prepare empty MILP Model M
  $\mathcal{M}$. var $\leftarrow \mathrm{s}_i^0$ for $i \in \{0, 1, \ldots, 287\}$
  for $i = 0$ *to 68* **do**
    $\mathrm{p}_\mathrm{i} \leftarrow \mathrm{s}_\mathrm{i}^0$
  for $i = 69$ *to 80* **do**
    $\mathcal{M}.\mathrm{var} \leftarrow \mathrm{p}_\mathrm{i}$
  for $i = 69$ *to 92 and* $i = 93 + 80$ *to 284* **do**
    $\mathcal{M}.\mathrm{con} \leftarrow \mathrm{s}_\mathrm{i}^0 = 0$
  for $i = 93$ *to 172* **do**
    $\mathcal{M} \cdot$ con $\leftarrow \mathrm{s}_i^0 = 1 \forall i - 93 \in I$
    $\mathcal{M} \cdot$ con $\leftarrow \mathrm{s}_i^0 = 0 \forall i - 93 \notin I$
  for $r = 0$ *to* $r = R - 1$ **do**
    if $r > 11$ *and* $r < 24$ **then**
      $(\mathcal{M}, \mathrm{x}_0, \ldots, \mathrm{x}_{287}) =$ TriviumCorep$(\mathcal{M}, \mathrm{s}_1^\mathrm{r}, \ldots, \mathrm{s}_{288}^\mathrm{r}, p_{92-r})$
    else
      $(\mathcal{M}, \mathrm{x}_0, \ldots, \mathrm{x}_{287}) =$ TriviumCore$(\mathcal{M}, \mathrm{s}_1^\mathrm{r}, \ldots, \mathrm{s}_{288}^\mathrm{r}, 65, 170, 90, 91, 92)$
    $(\mathcal{M}, \mathrm{y}_0, \ldots, \mathrm{y}_{287}) =$TriviumCore$(\mathcal{M}, \mathrm{x}_1, \ldots, \mathrm{x}_{288}, 161, 263, 174, 175, 176)$
    $(\mathcal{M}, \mathrm{z}_0, \ldots, \mathrm{z}_{287}) =$TriviumCore$(\mathcal{M}, \mathrm{y}_1, \ldots, \mathrm{y}_{288}, 242, 68, 285, 286, 287)$
    $(\mathrm{s}_0^{r+1}, \ldots, \mathrm{s}_{287}^{i+1}) =$TriviumCore$(\mathrm{z}_{287}, \mathrm{z}_0, \ldots, \mathrm{z}_{286})$
  for $i \in \{0, 1, \ldots, 287\}$ *without* $65, 92, 161, 176, 242, 287$ **do**
    $\mathcal{M} \cdot$ con $\leftarrow \mathrm{s}_i^{\mathrm{R}-1} = 0$
  $\mathcal{M} \cdot$ con $\leftarrow \mathrm{s}_{65}^{\mathrm{R}-1} + \mathrm{s}_{92}^{\mathrm{R}-1} + \mathrm{s}_{161}^{\mathrm{R}-1} + \mathrm{s}_{176}^{\mathrm{R}-1} + \mathrm{s}_{242}^{\mathrm{R}-1} + \mathrm{s}_{287}^{\mathrm{R}-1} = 1$
  return $(\mathcal{M}, \mathrm{p}_0, \mathrm{p}_1, \ldots, \mathrm{p}_{80})$