A Practical Method to Recover Exact Superpoly in Cube Attack

SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang and TaiRong Shi

PLA SSF Information Engineering University, Zhengzhou, China, wsp2110@126.com

Abstract. Cube attack is an important cryptanalytic technique against symmetric cryptosystems, especially for stream ciphers. The key step in cube attack is recovering superpoly. However, when cube size is large, the large time complexity of recovering the exact algebraic normal form (ANF) of superpoly confines cube attack. At CRYPTO 2017, Todo *et al.* applied conventional bit-based division property (CBDP) into cube attack which could exploit large cube sizes. However, CBDP based cube attacks cannot ensure that the superpoly of a cube is non-constant. Hence the key recovery attack may be just a distinguisher. Moreover, CBDP based cube attacks can only recover partial ANF coefficients of superpoly. The time complexity of recovering the reminding ANF coefficients is very large, because it has to query the encryption oracle and sum over the cube set. To overcome these limits, in this paper, we propose a practical method to recover the ANF coefficients of superpoly. This new method is developed based on bit-based division property using three subsets (BDPT) proposed by Todo at FSE 2016. We apply this new method to reduced-round Trivium. To be specific, the time complexity of recovering the superpoly of 832-round Trivium at CRYPTO 2017 is reduced from 2^{77} to practical, and the time complexity of recovering the superpoly of 839-round Trivium at CRYPTO 2018 is reduced from 2^{79} to practical. Then, we propose a theoretical attack which can recover the superpoly of Trivium up to 842 round. As far as we know, this is the first time that the superpoly can be recovered for Trivium up to 842 rounds.

Keywords: Trivium · MILP · Cube attack · Division property · Stream cipher

1 Introduction

Cube attack proposed by Dinur and Shamir [2] at EUROCRYPT 2009 is one of the general cryptanalytic techniques against symmetric cryptosystems. And it can be seen as a generalization of higher-order differential attack [11] and chosen IV statistical attack [4, 5]. The core idea of cube attack is to simplify the polynomial by summing the output of cryptosystem over a subset of public variables, called *cube*. And the target of cube attack is to recover secret variables from the simplified polynomial denoted as *superpoly*. In the original paper of cube attack [2], the authors regarded stream cipher as a blackbox polynomial and introduced a linearity test to recover the structure of the superpoly. Moreover, a quadraticity test was introduced in [14]. Recently, many variants of cube attacks were put forward such as dynamic cube attacks [3], conditional cube attacks [10], correlation cube attacks [12], division property based cube attacks [18, 19], and deterministic cube attacks [22].

When applying cube attack to stream cipher, we have to analyze the ANF of superpoly. At the beginning, due to the complicated structure of stream cipher, the cube attacks regarded it as blackbox. Therefore, the ANF of superpoly could only be detected by practical experiments. Such as in [6], when the cube size was 32, the secret keys of 799-round Trivium could be recovered. Note that the initial cube attacks are experimental

cryptanalysis, and we cannot evaluate the security when the size of cube exceeds 40, which limits their wide applications.

In [12], Liu *et al.* proposed a new key recovery attack on Trivium referred as *correlation cube attack*, which could mount to 835-round Trivium using small dimensional cubes. For a cube, they first tried to find a set of low-degree polynomials, called a *basis*. Then, by exploiting the conditional correlation properties between the low-degree basis and the superpoly, they could obtain a set of probabilistic equations with the secret key variables. In [22], Ye *et al.* proposed a new variant of cube attack, named *deterministic cube attacks*. Their attacks were developed based on degree evaluation method proposed by Liu *et al.* at CRYPTO 2017 [13]. They proposed a special type of cube that the numeric degree of every term was always less than or equal to the cube size, called *useful cube*. With a 37-dimensional useful cube, they recovered the corresponding exact superpoly for up to 838-round Trivium. However, as the authors wrote in their paper, it seemed hard to increase the number of attacking round when the cube size increased. Namely, their methods didn't work well for large cube size.

Division property is a generalization of integral property proposed by Todo [17] at EUROCRYPT 2015. It can exploit the algebraic structure of block ciphers to construct integral distinguishers even if the block ciphers have non-bijective, bit-oriented, or low-degree structures. Then, at CRYPTO 2015, Todo applied this new technique to MISTY1 and achieved the first theoretical cryptanalysis of the full-round MISTY1 [15]. In order to exploit the concrete structure of round function, Todo and Morii [16] proposed bit-based division property at FSE 2016. There are two kinds of bit-based division property: conventional bit-based division property (CBDP) and bit-based division property using three subsets (BDPT). CBDP focuses on that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$ is 0 or unknown, while BDPT focuses on that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$ is 0, 1, or unknown. Therefore, BDPT can find more accurate integral characteristics than CBDP. For example, CBDP proved the existence of the 14-round integral distinguisher for SIMON32 while BDPT found the 15-round integral distinguisher [16].

Although CBDP and BDPT could find accurate integral distinguishers, the huge complexity once restricted their wide applications. As shown in [16], for an n-bit block cipher, the propagation of bit-based division property required 2^n complexity. At ASIACRYPT 2016, Xiang *et al.* [21] applied mixed integer linear programming (MILP) method to search integral distinguishers based on CBDP, which allowed them to analyze block ciphers with large sizes. Compared with the propagation of CBDP, the propagation of BDPT is more complicated and cannot be modeled by MILP method directly. But recently, by using pruning techniques, Wang *et al.* [20] proposed the MILP-aided method to solve the propagation of BDPT.

At CRYPTO 2017 [18], Todo *et al.* treated the polynomial as non-blackbox and applied CBDP to the cube attack on stream ciphers. Due to the MILP-aided CBDP, they evaluate the ANF of the superpoly with large cube size. By using a 72-dimensional cube, they proposed a theoretical cube attack on 832-round Trivium. Then, at CRYPTO 2018 [19], Wang *et al.* introduced several techniques (flag technique, degree evaluation, and term enumeration) to improve the CBDP based cube attacks. With these new techniques, they proposed the key recovery attack on 839-round Trivium. For CBDP based cube attacks, the superpolies of large cubes can be recovered by theoretical method. But the theory of CBDP can not ensure that the superpoly of a cube is non-constant. Hence the key recovery attack may be just a distinguisher.

Besides the CBDP based cube attack, it is also noticeable that, at CRYPTO 2018, Fu *et al.* [7] proposed a key recovery attack on 855-round Trivium which somewhat resembled dynamic cube attacks [3]. Their basic idea is finding a proper internal state whose ANF representation is P_1. Suppose the output bit polynomial z could be formally represented

as $z = P_1P_2 \oplus P_3$, where P_2 is much more complex than P_3. If multiply $P_1 \oplus 1$ in both sides, $(P_1 \oplus 1)\, z = (P_1 \oplus 1)\, P_3$ will be simplified. The right guess of key bits involved in P_1 will lead the cube sum to be zero, otherwise the cube sum will be random. They declared that three secret key bits could be recovered from the 855-round Trivium with the online complexity of 2^{74}. For the attack in [7], the paper [9] pointed out that there was possibility that the correct key guesses and the wrong ones shared the same zero-sum property. It means the key recovery attack may degenerate to distinguish attack.

1.1 Our Contributions

Except for deterministic cube attacks, all the prior well-known cube attacks may have failure probabilities. However, the deterministic cube attacks don't work well for large cube size. To overcome these drawbacks, we propose a BDPT based cube attack which can recover the exact ANF of superpoly with large cube size.

Inspired by the CBDP based cube attack in [18, 19], our method is based on the propagation of BDPT. The BDPT focuses on not only the integral distinguishers whose sums are 0, but also the integral distinguishers whose sums are 1. So BDPT can determine that the ANF coefficients of some terms in polynomial are 1. In the following, our contributions are summarized into three aspects.

Using BDPT to Recover the ANF Coefficients of Polynomial. Since BDPT is a tool to find integral distinguishers whose sums are 0 or 1. We show the conditions under which the ANF coefficient of the maximum term in a polynomial can be recovered by the propagation of BDPT. In order to determine the remaining ANF coefficients through BDPT, we construct a new polynomial called *similar polynomial*. If we want to recover the ANF coefficient of a non-maximum term, we only need to recover the ANF coefficient of the maximum term in a similar polynomial.

MILP-aided Algorithm to Recover the ANF Coefficient of Superpoly. We show the conditions under which the ANF coefficients of superpoly can be obtained by the propagation of BDPT. However, when the number of vectors in the initial BDPT vector set is larger, it will cause trouble to the propagation of BDPT. So we present two techniques to reduce the initial BDPT's $\mathbb{L}$ set. Finally, we propose an MILP-aided algorithm to recover the ANF coefficient of superpoly.

The BDPT Based Cube Attack. In order to analyze the security of ciphers better, we divide ciphers into two categories: public-update ciphers and secret-update ciphers. For public-update ciphers, we proved that the exact ANF of superpoly can be fully recovered by BDPT. Fortunately, many stream ciphers belong to public-update ciphers. So our method has a good application future.

In order to verify the correctness and effectiveness of our method, we apply our new cube attack to Trivium which is a public-update cipher. Our results show that the CBDP based cube attack on 839-round Trivium in [19] is not a key recovery attack. For the CBDP based cube attack on 832-round Trivium in [18], only proper non-cube IV assignments can obtain a non-constant superpoly. Because our method can recover the ANF coefficients of superpoly in practical time, we show a theoretical attack on 842-round Trivium. The summarization of cube attacks on Trivium is shown in Table 1. The time complexity in the table means the time complexity of recovering superpoly. And c is the average computational complexity of tracing the propagation of BDPT using MILP-aided method.

1.2 Organization.

The remainder of this paper is organized as follows. Sect.2 provides the background of cube attacks, division property, and the CBDP based cube attack etc. In Sect.3, we use BDPT to analyze the ANF coefficients of polynomial and superpoly. In Sect.4, a new variant of cube attack based on BDPT is proposed. In Sect.5, we apply our new attack to Trivium. Sect.6 concludes the paper and summarizes our results.

Table 1: Summarization of cube attacks on Trivium

Rounds	Cube size	Exact superpoly	Complexity	Reference
799	32	no	practical	[6]
832	72	yes	2^{77}	[18]
			$2^{76.7}$	[19]
			practical	**Sect.5.3**
835	36/37	no	2^{75}	[12]
838	37	yes	practical	[22]
839	78	yes	2^{79}	[19]
			practical	**Sect.5.3**
842	79	yes	$2^{32} \cdot c$	**Sect.5.4**

2 Preliminaries

2.1 Notations

Here, we present the notations used throughout this paper. Let $\mathbb{F}_2$ denote the finite field $\{0,1\}$ and $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{F}_2^n$ be an n-bit vector, where a_i denotes the i-th bit of $\boldsymbol{a}$. The hamming weight of $\boldsymbol{a}$ is calculated as $hm(\boldsymbol{a}) = \sum_{i=1}^{n} a_i$. We use $\oplus$ and $+$ as the bit-based addition of $\mathbb{F}_2^n$ and addition of $\mathbb{Z}$, where $\mathbb{Z}$ denotes the integer ring. Let $\emptyset$ be an empty set. For a subset $I \subset \{1, 2, \ldots, n\}$, $\boldsymbol{u}_I$ denotes an n-dimensional bit vector satisfying $u_i = 1$ if $i \in I$ and $u_i = 0$ otherwise. For any $\boldsymbol{k}, \boldsymbol{k'} \in \mathbb{F}_2^n$, define $\boldsymbol{k} \succeq \boldsymbol{k'}$ if $k_i \geq k'_i$ holds for all $i = 1, 2, \ldots, n$, and $\boldsymbol{k} \preceq \boldsymbol{k'}$ if $k_i \leq k'_i$ holds for all $i = 1, 2, \ldots, n$.

2.2 Cube Attack

Cube attack, which can be regarded as an extension of higher-order differential cryptanalysis [11], was proposed by Dinur and Shamir at EUROCRYPT 2009 [2]. For a cipher with n secret variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and m public variables $\boldsymbol{v} = (v_1, v_2, \ldots, v_m)$, the output bits can be represented as $f(\boldsymbol{x}, \boldsymbol{v})$. In the case of stream ciphers, $\boldsymbol{x}$ is the secret key, $\boldsymbol{v}$ is the initialization vector, and $f(\boldsymbol{x}, \boldsymbol{v})$ is the first bit of the key stream sequences. Attackers determine an indices subset $I_v = \{i_1, i_2, \ldots, i_{|I_v|}\} \subset \{1, 2, \ldots, m\}$, then $f(\boldsymbol{x}, \boldsymbol{v})$ can be uniquely represented as

$$f(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{v}^{\boldsymbol{u}_{I_v}} \cdot p(\boldsymbol{x}, \boldsymbol{v}) \oplus q(\boldsymbol{x}, \boldsymbol{v}),$$

where $\boldsymbol{v}^{\boldsymbol{u}_{I_v}} = v_{i_1} \cdots v_{i_{|I_v|}}$. Then, $p(\boldsymbol{x}, \boldsymbol{v})$ is called the *superpoly* of I_v in $f(\boldsymbol{x}, \boldsymbol{v})$, and every term in $q(\boldsymbol{x}, \boldsymbol{v})$ misses at least one variable from $\left(v_{i_1}, v_{i_2}, \ldots, v_{i_{|I_v|}}\right)$.

Because $p(\boldsymbol{x}, \boldsymbol{v})$ doesn't contain any of the cube variables $\left(v_{i_1}, v_{i_2}, \ldots, v_{i_{|I_v|}}\right)$, the value of it can only be affected by secret key bits and the assignment to the non-cube public variables. Attackers can prepare a cube set denoted as C_{I_v, J_v, K_v}, where public variables indexed by I_v are taking all possible combinations of values, public variables

indexed by $J_v \subset \{1,2,\ldots,m\} - I_v$ are set to constant 1, and public variables indexed by $K_v = \{1,2,\cdots,m\} - I_v - J_v$ are set to constant 0. Then, it can be expressed as follow

$$C_{I_v,J_v,K_v} = \{\boldsymbol{v} \in \mathbb{F}_2^m | v_i \in \mathbb{F}_2 \text{ for } i \in I_v, v_j = 1 \text{ for } j \in J_v, \text{ and } v_k = 0 \text{ for } k \in K_v\}. \quad (1)$$

What's more, the sum of $f(\boldsymbol{x},\boldsymbol{v})$ over the cube set C_{I_v,J_v,K_v} is

$$\bigoplus_{\boldsymbol{v}\in C_{I_v,J_v,K_v}} f(\boldsymbol{x},\boldsymbol{v}) = \bigoplus_{\boldsymbol{v}\in C_{I_v,J_v,K_v}} \boldsymbol{v}^{\boldsymbol{u}_{I_v}} \cdot p(\boldsymbol{x},\boldsymbol{v}) \oplus \bigoplus_{\boldsymbol{v}\in C_{I_v,J_v,K_v}} q(\boldsymbol{x},\boldsymbol{v}) = p_{I_v,J_v,K_v}(\boldsymbol{x}). \quad (2)$$

If $p_{I_v,J_v,K_v}(\boldsymbol{x})$ is not a constant polynomial, attackers can query the encryption oracle with the chosen cube C_{I_v,J_v,K_v} to get the equation with secret variables. Otherwise, Eq. (2) can only provide a distinguisher for the cipher.

2.3 Bit-based Division Property

Division property, a generalization of integral property, was proposed by Todo at EUROCRYPT 2015 [17]. Then, two kinds of bit-based division property (CBDP and BDPT) were introduced by Todo and Morii at FSE 2016 [16]. In this subsection, we will briefly introduce these two kinds of bit-based division property and their propagation rules.

Definition 1. (CBDP [16]). *Let $\mathbb{X}$ be a multiset whose elements take values from $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, where $\mathbb{K}$ denotes a set of n-dimensional vectors whose i-th element takes a value between 0 and 1, it fulfills the following conditions:*

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

where $\boldsymbol{u} \succeq \boldsymbol{k}$ if $u_i \geq k_i$ holds for all $i = 1,2,\ldots,n$, and $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=1}^n x_i^{u_i}$.

CBDP focuses on that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$ is 0 or unknown. Because CBDP is insufficient to find the 15-round integral distinguisher for SIMON32, the paper [16] introduced a new variant of bit-based division property called bit-based division property using three subsets (BDPT). BDPT focuses on that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$ is 0, 1, or unknown.

Definition 2. (BDPT [16]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ and $\mathbb{L}$ be two sets whose elements take n-dimensional bit vectors. When the multiset $\mathbb{X}$ has the BDPT $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, it fulfills the following conditions:*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} = \begin{cases} \text{unknown}, & \text{if there is } \boldsymbol{k} \in \mathbb{K} \text{ satisfying } \boldsymbol{u} \succeq \boldsymbol{k} \\ 1, & \text{else if there is } \boldsymbol{\ell} \in \mathbb{L} \text{ satisfying } \boldsymbol{u} = \boldsymbol{\ell} \\ 0, & \text{otherwise} \end{cases}.$$

If there are $\boldsymbol{k} \in \mathbb{K}$ and $\boldsymbol{k}' \in \mathbb{K}$ satisfying $\boldsymbol{k} \succeq \boldsymbol{k}'$, $\boldsymbol{k}$ can be removed from $\mathbb{K}$ because the vector $\boldsymbol{k}$ is redundant. What's more, if there are $\boldsymbol{\ell} \in \mathbb{L}$ and $\boldsymbol{k} \in \mathbb{K}$, the vector $\boldsymbol{\ell}$ is also redundant if $\boldsymbol{\ell} \succeq \boldsymbol{k}$. The propagation rules of $\mathbb{K}$ in CBDP are the same with BDPT. So here we only show the propagation rules of BDPT. For more details, please refer to [16].

BDPT Rule 1 (Copy [16]). *Let $(x_1,x_2,\ldots,x_n) \in \mathbb{F}_2^n$ be the input of a Copy function, and $(x_1,x_1,x_2,\ldots,x_n)$ be the output. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^{n+1}}$.*

$$\mathbb{K}' \leftarrow \begin{cases} \{(0,0,k_2,\ldots,k_n)\}, & \text{if } k_1 = 0 \\ \{(1,0,k_2,\ldots,k_n),(0,1,k_2,\ldots,k_n)\}, & \text{if } k_1 = 1 \end{cases},$$

$$\mathbb{L}' \leftarrow \begin{cases} \{(0,0,\ell_2,\ldots,\ell_n)\}, & \text{if } \ell_1 = 0 \\ \{(1,0,\ell_2,\ldots,\ell_n),(0,1,\ell_2,\ldots,\ell_n),(1,1,\ell_2,\ldots,\ell_n)\}, & \text{if } \ell_1 = 1 \end{cases},$$

computed from all $\boldsymbol{k} \in \mathbb{K}$ *and all* $\boldsymbol{\ell} \in \mathbb{L}$*, respectively.*

BDPT Rule 2 (And [16]). *Let* $(x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$ *be the input of And function, and* $(x_1 \wedge x_2, \ldots, x_n)$ *be the output. Assuming the input multiset* $\mathbb{X}$ *has* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$*, then the output multiset* $\mathbb{Y}$ *has* $\mathcal{D}^{1^{n-1}}_{\mathbb{K}',\mathbb{L}'}$

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_1 + k_2}{2} \right\rceil, k_3, \ldots, k_n\right), \mathbb{L}' \leftarrow \left(\left\lceil \frac{\ell_1 + \ell_2}{2} \right\rceil, \ell_3, \ldots, \ell_n\right),$$

where $\mathbb{K}'$ *is computed from all* $\boldsymbol{k} \in \mathbb{K}$ *and* $\mathbb{L}'$ *is computed from all* $\boldsymbol{\ell} \in \mathbb{L}$ *satisfying* $(\ell_1, \ell_2) = (0,0)$ *or* $(1,1)$*.*

BDPT Rule 3 (Xor [16]). *Let* $(x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$ *be the input of Xor function, and* $(x_1 \oplus x_2, x_3, \ldots, x_n)$ *be the output. Assuming the input multiset* $\mathbb{X}$ *has* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$*, then the output multiset* $\mathbb{Y}$ *has* $\mathcal{D}^{1^{n-1}}_{\mathbb{K}',\mathbb{L}'}$

$$\mathbb{K}' \leftarrow (k_1 + k_2, k_3, \ldots, k_n), \mathbb{L}' \xleftarrow{x} (\ell_1 + \ell_2, \ell_3, \ldots, \ell_n),$$

where $\mathbb{K}'$ *is computed from all* $\boldsymbol{k} \in \mathbb{K}$ *satisfying* $(k_1, k_2) = (0,0), (1,0),$ *or* $(0,1)$ *and* $\mathbb{L}'$ *is computed from all* $\boldsymbol{\ell} \in \mathbb{L}$ *satisfying* $(\ell_1, \ell_2) = (0,0)$*,* $(1,0)$*, or* $(0,1)$*. And* $\mathbb{L} \xleftarrow{x} \boldsymbol{\ell}$ *means*

$$\mathbb{L} := \begin{cases} \mathbb{L} \cup \{\boldsymbol{\ell}\} & \text{if the original } \mathbb{L} \text{ does not include } \boldsymbol{\ell}, \\ \mathbb{L} \setminus \{\boldsymbol{\ell}\} & \text{if the original } \mathbb{L} \text{ includes } \boldsymbol{\ell}. \end{cases}$$

BDPT Rule 4 (Xor with Secret Round Key [16]). *Let* $\mathbb{X}$ *and* $\mathbb{Y}$ *be the input and output multiset satisfying* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$ *and* $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$*, respectively. Then,* $\boldsymbol{y} \in \mathbb{Y}$ *is computed as* $\boldsymbol{y} = \boldsymbol{x} \oplus \boldsymbol{r_k}$*, where* $\boldsymbol{r_k}$ *is the secret round key. Assuming a round key is Xored with the* i*-th bit, then* $\mathbb{K}'$ *and* $\mathbb{L}'$ *is computed as*

$$\mathbb{K}' \leftarrow (\ell_1, \ell_2, \ldots, \ell_i \vee 1, \ldots, \ell_n), \mathbb{L}' = \mathbb{L},$$

for all $\boldsymbol{\ell} \in \mathbb{L}$ *satisfying* $\ell_i = 0$*.*

BDPT Rule 5 (S-box [20]). *Let* $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ *and* $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ *denote the input and output of an* n*-bit S-box, respectively. And* $y_i, i \in \{1, 2, \ldots, n\}$ *can be expressed as a Boolean function of* $(x_1, x_2, \ldots, x_n)$*. For the input BDPT* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}\}}$*, the output BDPT* $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$ *is as follows*

$$\mathbb{K}' = \{\boldsymbol{u}' \in \mathbb{F}_2^n | \text{ if } \boldsymbol{y}^{\boldsymbol{u}'} \text{ contains any term } \boldsymbol{x}^{\boldsymbol{v}} \text{ satisfying } \boldsymbol{v} \succeq \boldsymbol{u}, \boldsymbol{u} \in \mathbb{K}\},$$
$$\mathbb{L}' = \{\boldsymbol{u}' \in \mathbb{F}_2^n | \text{ if } \boldsymbol{y}^{\boldsymbol{u}'} \text{ contains the monomial } \boldsymbol{x}^{\boldsymbol{\ell}}\}.$$

When we consider the propagation of bit-based division property for public functions, we don't need to care about the dependencies between $\mathbb{K}$ and $\mathbb{L}$. However, independent propagations may generate many redundant vectors. Although for any $\boldsymbol{u}$, the redundant vectors in $\mathbb{K}'$ and $\mathbb{L}'$ do not affect whether the parity becomes 0, 1, or unknown, we should remove redundant vectors if possible because of the only reason of complexity.

2.4 The MILP Representation of CBDP

Although CBDP has been proven to be a powerful tool to find integral distinguishers, the time and memory complexity once restricted its applications to block ciphers whose block

sizes were large. At ASIACRYPT 2016, Xiang *et al.* [21] modeled CBDP propagations of three basic operations (Copy, Xor, And) by linear inequalities. Then, they converted a search algorithm under Todo's framework into an MILP problem and solved the MILP problem by the openly available MILP optimizer Gurobi [8]. Recently, Wang *et al.* [19] improved the MILP models of Copy, Xor, and And by introducing the flag technique and named their improved version as **copyf**, **xorf**, **andf**. Here, we only introduce the MILP models for **copyf**, **xorf**, **andf**.

Flag Technique [19]. Every variable in the MILP model $v \in \mathcal{M}.var$ corresponds to an additional flag $v.F \in \{0_c, 1_c, \delta\}$, where 1_c means the bit is constant 1, 0_c means the bit is constant 0, and δ means the remaining cases. Corresponding to the bitwise Copy, Xor, and And operations, the flag values 0_c, 1_c, δ have $=$, $\oplus$, and $\times$ operations. The $=$ operation is naturally $1_c = 1_c$, $0_c = 0_c$, and $\delta = \delta$. The $\oplus$ operation follows the rules:

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

The $\times$ operation follows the rules:

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

Proposition 1. (MILP Model for copyf [19]). *Let $a \to (b_1, b_2, \ldots, b_n)$ be a division trail of Copy. The following inequalities are sufficient to describe the propagation of the division property for* **copyf**

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

This process is denoted as $(\mathcal{M}, b_1, \ldots, b_n) \leftarrow \boldsymbol{copyf}(\mathcal{M}, a, n)$.

Proposition 2. (MILP Model for xorf [19]). *Let $(a_1, a_2, \ldots, a_n) \to b$ be a division trail of Xor. The following inequalities are sufficient to describe the propagation of the division property for* **xorf**

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

This process is denoted as $(\mathcal{M}, b) \leftarrow \boldsymbol{xorf}(\mathcal{M}, a_1, \ldots, a_n)$.

Proposition 3. (MILP Model for andf [19]). *Let $(a_1, a_2, \ldots, a_n) \to b$ be a division trail of And. The following inequalities are sufficient to describe the propagation of the division property for* **andf**

$$\begin{cases} \mathcal{M}.var \leftarrow a_1, a_2, \ldots, a_n, b \text{ as binary.} \\ \mathcal{M}.con \leftarrow b \geqslant a_i \text{ for all } i \in \{1, 2, \ldots, n\} \\ b.F = a_1.F \times a_2.F \times \cdots \times a_n.F \\ \mathcal{M}.con \leftarrow b = 0 \text{ if } b.F = 0_c \end{cases}$$

This process is denoted as $(\mathcal{M}, b) \leftarrow \boldsymbol{andf}(\mathcal{M}, a_1, \ldots, a_n)$.

Note that MILP models for **copyf**, **xorf**, and **andf** are sufficient to represent any circuit. We are able to construct a linear inequality system $\mathcal{L}$ describing r-round CBDP. All the feasible solutions of $\mathcal{L}$ correspond to all the r-round division trails, which are defined as follows.

Definition 3. (Division Trail [21]). *Let us consider the propagation of the CBDP* $\{\boldsymbol{k}\} \overset{def}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r$. *Moreover, for any vector* $\boldsymbol{k}_{i+1} \in \mathbb{K}_{i+1}$, *there must exist a vector* $\boldsymbol{k}_i \in \mathbb{K}_i$ *such that* $\boldsymbol{k}_i$ *can propagate to* $\boldsymbol{k}_{i+1}$ *by the propagation rules of CBDP. Furthermore, for* $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$, *if* $\boldsymbol{k}_i$ *can propagate to* $\boldsymbol{k}_{i+1}$ *for all* $i \in \{0, 1, \ldots r-1\}$, *we call* $(\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r)$ *an* r*-round division trail.*

If $\mathbb{K}_{r+1}$ for the first time contains all the n unit vectors, the CBDP propagation should stop and an r-round distinguisher can be derived. For more details, please refer to [15, 16, 17, 21].

2.5 The Pruning Techniques of BDPT

Compared with the propagation of CBDP, the propagation of BDPT is more complicated and cannot be modeled by MILP method directly. Recently, an MILP-aided method of searching integral distinguishers based on BDPT was proposed in [20]. The main insights are the pruning techniques of BDPT as below. For more information, please refer to [20].

Proposition 4. (Prune $\mathbb{K}$ [20]) *For* r*-round cipher* $f = f_r \cdot f_{r-1} \cdot \cdots \cdot f_1$, *let* $\mathcal{D}^{1^n}_{\mathbb{K}_i, \mathbb{L}_i}$ *be the input BDPT of* $f_{(r,i+1)} = f_r \cdot f_{r-1} \cdots f_{i+1}$. *If* $\mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{k}\}}$ *cannot generate the output unit vector* $\boldsymbol{e}_m$ *of* $f_{(r,i+1)}$ *based on CBDP, then* $\mathcal{D}^{1^n}_{\mathbb{K}_i, \mathbb{L}_i}$ *is equivalent to* $\mathcal{D}^{1^n}_{\mathbb{K}_i \to \boldsymbol{k}, \mathbb{L}_i}$ *on whether* $\boldsymbol{e}_m \in \mathbb{K}_r$ *and* $\boldsymbol{e}_m \in \mathbb{L}_r$ *or not, where* $\mathbb{K}_i \to \boldsymbol{k}$ *denotes removing* $\boldsymbol{k}$ *from* $\mathbb{K}_i$.

Proposition 5. (Prune $\mathbb{L}$ [20]) *For* r*-round cipher* $f = f_r \cdot f_{r-1} \cdot \cdots \cdot f_1$, *let* $\mathcal{D}^{1^n}_{\mathbb{K}_i, \mathbb{L}_i}$ *be the input BDPT of* $f_{(r,i+1)} = f_r \cdot f_{r-1} \cdots f_{i+1}$. *If* $\mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{\ell}\}}$ *cannot generate the output unit vector* $\boldsymbol{e}_m$ *of* $f_{(r,i+1)}$ *based on CBDP, then* $\mathcal{D}^{1^n}_{\mathbb{K}_i, \mathbb{L}_i}$ *is equivalent to* $\mathcal{D}^{1^n}_{\mathbb{K}_i, \mathbb{L}_i \to \boldsymbol{\ell}}$ *on whether* $\boldsymbol{e}_m \in \mathbb{K}_r$ *and* $\boldsymbol{e}_m \in \mathbb{L}_r$ *or not, where* $\mathbb{L}_i \to \boldsymbol{\ell}$ *denotes removing* $\boldsymbol{\ell}$ *from* $\mathbb{L}_i$.

2.6 The Cube Attack Based on CBDP

At CRYPTO 2017, Todo *et al.* successfully applied the CBDP to cube attack [18]. They regarded the superpoly as non-blackbox polynomial and proposed a method to determine the ANF of superpoly. Then, at CRYPTO 2018, Wang *et al.* [19] showed an improved version which could further reduce the complexity of recovering the superpoly.

Lemma 1. [18] *Let* $f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a^f_{\boldsymbol{u}} \cdot \boldsymbol{x}^{\boldsymbol{u}}$ *be a polynomial from* $\mathbb{F}_2^n$ *to* $\mathbb{F}_2$ *and* $a^f_{\boldsymbol{u}} \in \mathbb{F}_2$ *be the ANF coefficients. Let* $\boldsymbol{k}$ *be an* n*-dimensional bit vector. Assuming there is no division trail such that* $\mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{k}\}} \xrightarrow{f} 1$, *then* $a^f_{\boldsymbol{u}}$ *is always 0 for* $\boldsymbol{u} \succeq \boldsymbol{k}$.

Proposition 6. [18] *Let* $f(\boldsymbol{x}, \boldsymbol{v})$ *be a polynomial, where* $\boldsymbol{x} \in \mathbb{F}_2^n$ *and* $\boldsymbol{v} \in \mathbb{F}_2^m$ *denote the secret and public variables, respectively. For a cube set* C_{I_v, J_v, K_v} *defined as Eq. (1), let* $\boldsymbol{e}_i$ *be the unit vector whose only* i*-th element is 1. Assuming there is no division trail such that* $\mathcal{D}^{1^{n+m}}_{\mathbb{K}=\{(\boldsymbol{e}_i, \boldsymbol{u}_{I_v})\}} \xrightarrow{f} 1$, *then* x_i *is not involved in the superpoly of the cube* C_{I_v, J_v, K_v}.

When $f(\boldsymbol{x}, \boldsymbol{v})$ represents the first output bit after the initial iterations, we can identify the involved keys by checking whether there is division trial $\mathcal{D}^{1^{n+m}}_{\mathbb{K}=\{(\boldsymbol{e}_i, \boldsymbol{u}_{I_v})\}} \xrightarrow{f} 1$ for $i = 1, \cdots, n$ using the MILP modeling method. Then, we will get the involved keys set I and the ANF of $p_{I_v, J_v, K_v}(\boldsymbol{x})$ can be represented as

$$p_{I_v, J_v, K_v}(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \preceq \boldsymbol{u}_I} a^{p_{I_v, J_v, K_v}}_{\boldsymbol{u}} \cdot \boldsymbol{x}^{\boldsymbol{u}}$$

If the degree of the superpoly is upper bounded by d, then for all $\boldsymbol{u}$ satisfying $hw(\boldsymbol{u}) > d$, we have $a^{p_{I_v, J_v, K_v}}_{\boldsymbol{u}} = 0$. Such a degree evaluation is based on the following proposition, which can be regarded as a generalization of Proposition 6.

Proposition 7. [19] *For a set* $I_x = \{i_1, i_2, \ldots, i_{|I_x|}\} \subset \{1, 2, \ldots, n\}$*, if there is no division trail* $\mathcal{D}^{1^{n+m}}_{\mathbb{K}=\{(\boldsymbol{u}_{I_x}, \boldsymbol{u}_{I_v})\}} \xrightarrow{f} 1$*, then* $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ *is not involved in the superpoly of cube* C_{I_v,J_v,K_v}*.*

After getting the involved key set I and the degree d of superpoly, the superpoly can be represented with $\sum_{i=0}^{d}\binom{|I|}{i}$ coefficients. Therefore, by selecting $\sum_{i=0}^{d}\binom{|I|}{i}$ different $\boldsymbol{x}$'s, a linear system with $\sum_{i=0}^{d}\binom{|I|}{i}$ variables can be constructed. Then, the whole ANF of $p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be recovered by solving such a linear system. So the complexity of recovering the superpoly of cube C_{I_v,J_v,K_v} is $2^{|I_v|} \times \sum_{i=0}^{d}\binom{|I|}{i}$.

3 What Can BDPT Do

In this section, we propose a new technique to analyze the ANF coefficients of non-blackbox polynomial and superpoly in cube attack.

3.1 Analyze the ANF Coefficients of Non-blackbox Polynomial

Let $f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u}\in\mathbb{F}_2^n} a_{\boldsymbol{u}}^f \cdot \boldsymbol{x}^{\boldsymbol{u}}$ be a polynomial, where $\boldsymbol{x} \in \mathbb{F}_2^n$ denotes the variables. For subsets of indices $I \subset \{1, 2, \ldots, n\}$, $J \subset \{1, 2, \cdots, n\} - I$, and $K = \{1, 2, \cdots, n\} - I - J$, if fix the variable $x_j = 1$ for $j \in J$, and $x_k = 0$ for $k \in K$, we can obtain a new polynomial denoted as $f_{I,J,K}(\boldsymbol{x})$. And the ANF of $f_{I,J,K}(\boldsymbol{x})$ can be represented as follow

$$f_{I,J,K}(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u}\preceq\boldsymbol{u}_I} a_{\boldsymbol{u}}^{f_{I,J,K}} \cdot \boldsymbol{x}^{\boldsymbol{u}}.$$

Lemma 2. *Let* $C_{I,J,K}$ *be a set of* $2^{|I|}$ *values, where the variables* $\{x_i | i \in I\}$ *are taking all possible combinations of values,* $x_j = 1$ *for* $j \in J$*, and* $x_k = 0$ *for* $k \in K$*. Then, the initial BDPT of* $C_{I,J,K}$ *is* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$*, where* $\mathbb{K} = \emptyset$ *and* $\mathbb{L} = \{\boldsymbol{u} | \boldsymbol{u}_I \preceq \boldsymbol{u} \preceq \boldsymbol{u}_I \oplus \boldsymbol{u}_J\}$*.*

Proof. Because all the values of elements in cube set $C_{I,J,K}$ are fixed. Then, for any n-bit vector $\boldsymbol{u}$, the value of $\bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \boldsymbol{x}^{\boldsymbol{u}}$ is deterministic. According to Definition 2, we know that $\mathbb{K} = \emptyset$.

For any vector $\boldsymbol{u} \in \{\boldsymbol{u} | \boldsymbol{u}_I \preceq \boldsymbol{u} \preceq \boldsymbol{u}_I \oplus \boldsymbol{u}_J\}$, we have

$$\bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \boldsymbol{x}^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \prod_{i\in I} {x_i}^{u_i} \prod_{j\in J} {x_j}^{u_j} \prod_{k\in K} {x_k}^{u_k} = \bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \prod_{i\in I} x_i = 1. \tag{3}$$

Then, for any n-bit vector $\boldsymbol{u} \notin \{\boldsymbol{u} | \boldsymbol{u}_I \preceq \boldsymbol{u} \preceq \boldsymbol{u}_I \oplus \boldsymbol{u}_J\}$, if there is $k \in K$ satisfying $u_k = 1$, we have

$$\bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \boldsymbol{x}^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \prod_{i\in I} {x_i}^{u_i} \prod_{j\in J} {x_j}^{u_j} \prod_{k\in K} {x_k}^{u_k} = \bigoplus_{\boldsymbol{x}\in C_{I,J,K}} 0 = 0.$$

If there is $i' \in I$ satisfying $u_{i'} = 0$, we have

$$\bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \boldsymbol{x}^{\boldsymbol{u}} = \bigoplus_{\substack{\boldsymbol{x}\in C_{I,J,K}\\ x_{i'}=0}} \prod_{\substack{i\in I\\ i\neq i'}} {x_i}^{u_i} \prod_{j\in J} {x_j}^{u_j} \prod_{k\in K} {x_k}^{u_k} \oplus \bigoplus_{\substack{\boldsymbol{x}\in C_{I,J,K}\\ x_{i'}=1}} \prod_{\substack{i\in I\\ i\neq i'}} {x_i}^{u_i} \prod_{j\in J} {x_j}^{u_j} \prod_{k\in K} {x_k}^{u_k} = 0.$$

So, for any n-bit vector $\boldsymbol{u} \notin \{\boldsymbol{u} | \boldsymbol{u}_I \preceq \boldsymbol{u} \preceq \boldsymbol{u}_I \oplus \boldsymbol{u}_J\}$, we have $\bigoplus_{\boldsymbol{x}\in C_{I,J,K}} \boldsymbol{x}^{\boldsymbol{u}} = 0$. Overall, the initial BDPT of $C_{I,J,K}$ is $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$, where $\mathbb{K} = \emptyset$ and $\mathbb{L} = \{\boldsymbol{u} | \boldsymbol{u}_I \preceq \boldsymbol{u} \preceq \boldsymbol{u}_I \oplus \boldsymbol{u}_J\}$. □

Lemma 3. *Let* $f_{I,J,K}(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u} \preceq \boldsymbol{u}_I} a_{\boldsymbol{u}}^{f_{I,J,K}} \cdot \boldsymbol{x}^{\boldsymbol{u}}$ *be a polynomial, and* $C_{I,J,K}$ *be the cube set. Considering the initial BDPT* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$*, where* $\mathbb{K} = \emptyset$ *and* $\mathbb{L} = \{\boldsymbol{u} | \boldsymbol{u}_I \preceq \boldsymbol{u} \preceq \boldsymbol{u}_I \oplus \boldsymbol{u}_J\}$*, we have*

(1) *Assuming there are no division trails such that* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\{1\},\emptyset}$ *and the number of division trails such that* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\emptyset,\{1\}}$ *is odd, then the maximum term* $\boldsymbol{x}^{\boldsymbol{u}_I}$*'s ANF coefficient in* $f_{I,J,K}(\boldsymbol{x})$ *is 1.*

(2) *Assuming there are no division trails such that* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\{1\},\emptyset}$ *and the number of division trails such that* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\emptyset,\{1\}}$ *is even, then the maximum term* $\boldsymbol{x}^{\boldsymbol{u}_I}$*'s ANF coefficient in* $f_{I,J,K}(\boldsymbol{x})$ *is 0.*

Proof. The ANF coefficient of $\boldsymbol{x}^{\boldsymbol{u}_I}$ can be obtained by calculating

$$\bigoplus_{\boldsymbol{x} \in C_{I,J,K}} f_{I,J,K}(\boldsymbol{x}) = \bigoplus_{\boldsymbol{x} \in C_{I,J,K}} \bigoplus_{\boldsymbol{u} \preceq \boldsymbol{u}_I} a_{\boldsymbol{u}}^{f_{I,J,K}} \cdot \boldsymbol{x}^{\boldsymbol{u}} = a_{\boldsymbol{u}_I}^{f_{I,J,K}} \tag{4}$$

(1) When there are no division trails such that $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\{1\},\emptyset}$ and the number of division trails such that $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\emptyset,\{1\}}$ is odd, it means that the sum of $f_{I,J,K}(\boldsymbol{x})$ over all values of the cube set $C_{I,J,K}$ is 1, i.e. $\bigoplus_{\boldsymbol{x} \in C_{I,J,K}} f_{I,J,K}(\boldsymbol{x}) = 1$. According to Eq. (4), we obtain that $\boldsymbol{x}^{\boldsymbol{u}_I}$'s ANF coefficient in $f_{I,J,K}(\boldsymbol{x})$ is 1.

(2) When there are no division trails such that $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\{1\},\emptyset}$ and the number of division trails such that $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I,J,K}} \mathcal{D}^{1^n}_{\emptyset,\{1\}}$ is even, it means that the sum of $f_{I,J,K}(\boldsymbol{x})$ over all values of the cube set $C_{I,J,K}$ is 0, i.e. $\bigoplus_{\boldsymbol{x} \in C_{I,J,K}} f_{I,J,K}(\boldsymbol{x}) = 0$. According to Eq. (4), we obtain that $\boldsymbol{x}^{\boldsymbol{u}_I}$'s ANF coefficient in $f_{I,J,K}(\boldsymbol{x})$ is 0. □

Lemma 3 shows the relationship between BDPT and the ANF coefficient of the maximum term. But for the remaining ANF coefficients of polynomial, it is nontrivial to obtain their values. In order to determine the values of these ANF coefficients through BDPT, we construct a new polynomial called *similar polynomial*.

Definition 4. (Similar Polynomial). *For subsets of indices* $I' \subset I$*,* $K' = \{1, 2, \ldots, n\} - I' - J$*, the polynomial* $f_{I',J,K'}(\boldsymbol{x})$ *is called the similar polynomial of* $f_{I,J,K}(\boldsymbol{x})$*.*

Lemma 4. *If* $f_{I',J,K'}(\boldsymbol{x})$ *is the similar polynomial of* $f_{I,J,K}(\boldsymbol{x})$*, then the value of ANF coefficient* $a_{\boldsymbol{u}_{I'}}^{f_{I',J,K'}}$ *in* $f_{I',J,K'}(\boldsymbol{x})$ *is equal to the value of ANF coefficients* $a_{\boldsymbol{u}_{I'}}^{f_{I,J,K}}$ *in* $f_{I,J,K}(\boldsymbol{x})$*.*

Proof. For $f_{I,J,K}(\boldsymbol{x})$, if all the variables of $\{x_i | i \in I - I'\}$ are assigned 0, it becomes the function $f_{I',J,K'}(\boldsymbol{x})$. Compared with the ANF of $f_{I,J,K}(\boldsymbol{x})$, the ANF of $f_{I',J,K'}(\boldsymbol{x})$ only misses terms that contain any variables of $\{x_i | i \in I - I'\}$. Moreover, $\boldsymbol{x}^{\boldsymbol{u}_{I'}}$ doesn't contain any variables of $\{x_i | i \in I - I'\}$, so $a_{\boldsymbol{u}_{I'}}^{f_{I,J,K}} = a_{\boldsymbol{u}_{I'}}^{f_{I',J,K'}}$. □

Because $\boldsymbol{x}^{\boldsymbol{u}_{I'}}$ is the maximum term of $f_{I',J,K'}(\boldsymbol{x})$, we can use Lemma 3 to get the value of ANF coefficient $a_{\boldsymbol{u}_{I'}}^{f_{I',J,K'}}$. Then, according to Lemma 4, we can get the ANF coefficient of term $\boldsymbol{x}^{\boldsymbol{u}_{I'}}$ in $f_{I,J,K}(\boldsymbol{x})$.

3.2 Analyze the ANF Coefficients of Superpoly

The most important part of cube attack is recovering the superpoly. Once the superpoly is recovered, attackers can compute the sum of encryptions over the cube and get one equation about secret variables.

Proposition 8. *Let $f(\boldsymbol{x},\boldsymbol{v})$ be a polynomial, where $\boldsymbol{x}\in\mathbb{F}_2^n$ and $\boldsymbol{v}\in\mathbb{F}_2^m$ denote the secret and public variables, respectively. In cube attack, $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ denotes a function that the public variables indexed by $I_v\subset\{1,2,\cdots,m\}$ are chosen as cube variables, public variables indexed by $J_v\subset\{1,2,\cdots,m\}-I$ are set to 1, and the remaining public variables $K_v=\{1,2,\cdots,m\}-I_v-J_v$ are set to 0. Then, for $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$, when fixing the secret variables $\{x_k|k\in\{1,2,\cdots,n\}-I_x\}$ to 0, we get a new polynomial denoted as $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$, where $I_x\subset\{1,2,\cdots,n\}$ is an index subset of secret variables. Let C_{I_v,J_v,K_v} be a cube set of $2^{|I_v|}$ values, where the variables $\{v_i|i\in I_v\}$ are taking all possible combinations of values, $v_j=1$ for $j\in J_v$, and $v_k=0$ for $k\in K_v$. Then, the superpoly of C_{I_v,J_v,K_v} is $p_{I_v,J_v,K_v}(\boldsymbol{x})=\bigoplus_{\boldsymbol{v}\in C_{I_v,J_v,K_v}} f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$. For BDPT $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}$, where $\mathbb{K}=\emptyset$, and $\mathbb{L}=\{(\boldsymbol{u}_{I_x},\boldsymbol{u}_v)\,|\boldsymbol{u}_{I_v}\preceq\boldsymbol{u}_v\preceq\boldsymbol{u}_{I_v}\oplus\boldsymbol{u}_{J_v}\}$, we have*

(1) *Assuming there are no division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}\xrightarrow{f_{I_x,I_v,J_v,K_v}}\mathcal{D}^{1^{n+m}}_{\{1\},\emptyset}$, and the number of division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}\xrightarrow{f_{I_x,I_v,J_v,K_v}}\mathcal{D}^{1^{n+m}}_{\emptyset,\{1\}}$ is odd, then the ANF coefficient of term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in the superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ is 1.*

(2) *Assuming there are no division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}\xrightarrow{f_{I_x,I_v,J_v,K_v}}\mathcal{D}^{1^{n+m}}_{\{1\},\emptyset}$, and the number of division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}\xrightarrow{f_{I_x,I_v,J_v,K_v}}\mathcal{D}^{1^{n+m}}_{\emptyset,\{1\}}$ is even, then the ANF coefficient of term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in the superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ is 0.*

Proof. For the function $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})=\bigoplus_{(\boldsymbol{u}_x,\boldsymbol{u}_v)\in\left(\mathbb{F}_2^n,\mathbb{F}_2^m\right)} a^{f_{I_v,J_v,K_v}}_{(\boldsymbol{u}_x,\boldsymbol{u}_v)}(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_x,\boldsymbol{u}_v)}$, as showed in Sect.2.2, it can be unique represented as

$$f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})=\boldsymbol{v}^{\boldsymbol{u}_{I_v}}\cdot p_{I_v,J_v,K_v}(\boldsymbol{x})\oplus q_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})\,. \tag{5}$$

And the ANF of $p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be presented as $p_{I_v,J_v,K_v}(\boldsymbol{x})=\bigoplus_{\boldsymbol{u}\in\mathbb{F}_2^n} a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}}\cdot\boldsymbol{x}^{\boldsymbol{u}}$. Then, the ANF coefficient of term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in $p_{I_v,J_v,K_v}(\boldsymbol{x})$ is $a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}}$.

Moreover, the ANF of $\boldsymbol{v}^{\boldsymbol{u}_{I_v}}\cdot p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be presented as

$$\boldsymbol{v}^{\boldsymbol{u}_{I_v}}\cdot p_{I_v,J_v,K_v}(\boldsymbol{x})=\bigoplus_{\boldsymbol{u}\in\mathbb{F}_2^n} a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}}\cdot(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u},\boldsymbol{u}_{I_v})}\,.$$

Then, the ANF coefficient of term $(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}$ in $\boldsymbol{v}^{\boldsymbol{u}_{I_v}}\cdot p_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ is also $a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}}$.

Because every term in $q_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ misses at least one variable from $\{v_i|i\in I_v\}$, the term $(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}$ doesn't exist in $q_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$. According to Eq. (5), We obtain that the ANF coefficient of term $(\boldsymbol{x},\boldsymbol{v})^{\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v}}$ in $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ is $a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}}$. Namely,

$$a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}}=a^{f_{I_v,J_v,K_v}}_{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})} \tag{6}$$

(1) If there are no division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}\xrightarrow{f_{I_x,I_v,J_v,K_v}}\mathcal{D}^{1^{n+m}}_{\{1\},\emptyset}$, and the number of division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}\xrightarrow{f_{I_x,I_v,J_v,K_v}}\mathcal{D}^{1^{n+m}}_{\emptyset,\{1\}}$ is odd, according to Lemma 3, the term $(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}$'s ANF coefficient in $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ is 1. Then, according to Lemma 4, the term $(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}$'s ANF coefficient in $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ is also 1, that is $a^{f_{I_v,J_v,K_v}}_{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}=1$. Therefore, we can get $a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}}=1$ from Eq. (6).

(2) We can complete the proof in a similar way to the above **(1)**. □

Proposition 8 can imply the existences of some terms in superpoly. Compared with the linearity test of cube attack which is statistical in nature, our method is algebraic, deterministic and it has no low-degree restriction.

3.3 The Algorithm to Compute the ANF Coefficients of Superpoly

According to Sect.2.6, for a polynomial $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ and cube set C_{I_v,J_v,K_v}, we can use MILP method to evaluate the secret variables involved in the superpoly and the upper bound degree of superpoly. We denote the involved secret variables indices set as I and the upper bound degree as d. Then, in order to recover the superpoly, we only need to determine the coefficients $a_{\boldsymbol{u}}^{p_{I_v,J_v,K_v}}$ satisfying $\boldsymbol{u} \preceq \boldsymbol{u}_I$ and $hw(\boldsymbol{u}) \leq d$.

From Proposition 8, when there are no division trails such that $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^{n+m}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}_{\{1\},\emptyset}^{1^{n+m}}$, the ANF coefficient of $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be got by the propagation of $\mathbb{L} = \{(\boldsymbol{u}_{I_x},\boldsymbol{u}_v)\,|\boldsymbol{u}_{I_v} \preceq \boldsymbol{u}_v \preceq \boldsymbol{u}_{I_v} \oplus \boldsymbol{u}_{J_v}\}$. And the set $\mathbb{L}$ has $2^{|J_v|}$ vectors. If $|J_v|$ is large, it will cause trouble to the propagation of BDPT. According to Proposition 5, for $\boldsymbol{\ell} \in \mathbb{L}$, if there are no division trails such that $\mathcal{D}_{\mathbb{K}=\{\boldsymbol{\ell}\}}^{1^{n+m}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}_{\mathbb{K}=\{1\}}^{1^{n+m}}$, we can discard $\boldsymbol{\ell}$ from $\mathbb{L}$ without affecting the result of BDPT propagation. For $\mathbb{L} = \{(\boldsymbol{u}_{I_x},\boldsymbol{u}_v)\,|\boldsymbol{u}_{I_v} \preceq \boldsymbol{u}_v \preceq \boldsymbol{u}_{I_v} \oplus \boldsymbol{u}_{J_v}\}$, we will identify the set of involved public variable indices $J_v^{I_x} \subset J_v$. The meaning of $J_v^{I_x}$ is that the values of all the public variables $v_j, j \in J_v - J_v^{I_x}$ don't affect the result of BDPT propagation. Then, $\mathbb{L}$ can be replaced by $\mathbb{L}' = \left\{(\boldsymbol{u}_{I_x},\boldsymbol{u}_v)\,|\boldsymbol{u}_{I_v} \preceq \boldsymbol{u}_v \preceq \boldsymbol{u}_{I_v} \oplus \boldsymbol{u}_{J_v^{I_x}}\right\}$. The specific progress is shown in Algorithm 1.

Algorithm 1: Identify the involved public variables

```
1  procedure InvolvedPublicVariable (I_x, I_v, J_v, K_v)
2      Declare an empty MILP model M
3      Declare x as n MILP variables of M corresponding to secret variables.
4      Declare v as m MILP variables of M corresponding to public variables.
5      M.con ← x_i = 1 and assign x_i.F = δ for all i ∈ I_x
6      M.con ← x_i = 0 and assign x_i.F = 0 for all i ∈ {1, 2, ··· , n} − I_x
7      M.con ← v_i = 1 and assign v_i.F = δ for all i ∈ I_v
8      M.con ← v_i = 0 and assign v_i.F = 0 for all i ∈ K_v
9      M.con ← Σ_{i∈J_v} v_i = 1 and assign v_i.F = δ for all i ∈ J_v
10     Update M according to the function f_{I_x,I_v,J_v,K_v}(x, v)
11     initial J_v^{I_x} = ∅
12     do
13         solve MILP model M
14         if M is feasible then
15             pick index j ∈ J_v s.t. v_j = 1, and J_v^{I_x} = J_v^{I_x} ⋃{j}
16             M.con ← v_j = 0
17         end if
18     while M is feasible
19     return J_v^{I_x}
20 end procedure
```

In order to further reduce the number of vectors in $\mathbb{L}'$, we will get the upper bound hamming weight of $\mathbb{L}'$, denoted as $uhw_{(I_x,I_v,J_v^{I_x})}$. The meaning of $uhw_{(I_x,I_v,J_v^{I_x})}$ is that, for all $\boldsymbol{\ell} \in \mathbb{L}'$ satisfying $hw(\boldsymbol{\ell}) > uhw_{(I_x,I_v,J_v^{I_x})}$, the division trail $\mathcal{D}_{\mathbb{K}=\{\boldsymbol{\ell}\}}^{1^{n+m}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}_{\mathbb{K}=\{1\}}^{1^{n+m}}$ does not exist, then we can discard them from $\mathbb{L}'$. Therefore, $\mathbb{L}'$ can be replaced

by $\mathbb{L}'' = \left\{(\boldsymbol{u}_{I_x}, \boldsymbol{u}_v) \,|\boldsymbol{u}_{I_v} \preceq \boldsymbol{u}_v \preceq \boldsymbol{u}_{I_v} \oplus \boldsymbol{u}_{J_v^{I_x}} \text{ and } hw(\boldsymbol{u}_{I_x}, \boldsymbol{u}_v) \leq uhw_{\left(I_x, I_v, J_v^{I_x}\right)}\right\}$. Using MILP, this $uhw_{\left(I_x, I_v, J_v^{I_x}\right)}$ can be naturally modeled as the maximum of the objective function $\sum\limits_{i \in I_x} x_i \sum\limits_{j \in I_v \bigcup J_v^{I_x}} v_j$. The specific progress is shown in Algorithm 2.

Algorithm 2: Evaluate the upper bound hamming weight

```
1  procedure HammingEvaluate(I_x, I_v, J_v^{I_x})
2      Declare an empty MILP model M
3      Declare x as n MILP variables of M corresponding to secret variables.
4      Declare v as m MILP variables of M corresponding to public variables.
5      M.con ← x_i = 1 and assign x_i.F = δ for all i ∈ I_x
6      M.con ← x_i = 0 and assign x_i.F = 0 for all i ∈ {1, 2, ··· , n} − I_x
7      M.con ← v_i = 1 and assign v_i.F = δ for all i ∈ I_v
8      M.con ← v_i = 0 and assign v_i.F = 0 for all i ∈ {1, 2, ··· , m} − I_v − J_v^{I_x}
9      Assign v_i.F = δ for all i ∈ J_v^{I_x}
10     Set the objective function M.obj ← Σ_{i∈I_x} x_i Σ_{j∈I_v ⋃ J_v^{I_x}} v_j
11     Update M according to the function f_{I_x,I_v,J_v,K_v}(x, v)
12     Solve MILP model M
13     return The solution of M.
14 end procedure
```

After getting the set of involved secret key indices I and the upper bound degree d of superpoly, for every I_x satisfying $I_x \subset I$ and $hw(\boldsymbol{u}_{I_x}) \leq d$, we can get the corresponding involved public variables $J_v^{I_x}$, and the upper bound hamming weight $uhw_{\left(I_x, I_v, J_v^{I_x}\right)}$. Then, we propose Algorithm 3 to recover the ANF coefficient of $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$.

Algorithm 3: Recover the ANF coefficient of $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$

```
1  procedure RecoverCoefficient(I_x, J_v^{I_x}, uhw_{(I_x,I_v,J_v^{I_x})})
2    Initial 𝕂 = ∅
3    Initial 𝕃 = { (u_{I_x}, u_v) | u_{I_v} ⪯ u_v ⪯ u_{I_v} ⊕ u_{J_v}^{I_x} and hw(u_{I_x}, u_v) ≤ uhw_{(I_x,I_v,J_v^{I_x})} }
4    if there is division trail D_{𝕂,𝕃} --f_{I_x,I_v,J_v,K_v}--> D_{{1},∅}
5        return unknown
6    else if the number of division trails such that D_{𝕂,𝕃} --f_{I_x,I_v,J_v,K_v}--> D_{∅,{1}} is odd
7        return 1
8    else
9        return 0
10 end procedure
```

4 The BDPT Based Cube Attacks

In order to analyze the ciphers better, we divide them into two categories: public-update ciphers and secret-update ciphers.

Definition 5. Let $f = f_n \cdot f_{n-1} \cdot \cdots f_1(\boldsymbol{x}, \boldsymbol{v})$ be an n-round cipher, where f_i is the i-th round update function, $\boldsymbol{x}$ denotes the secret variables, and $\boldsymbol{v}$ denotes the public variables. If the secret variables aren't involved in any of the round update functions

f_i, $i \in \{1, 2, \cdots, n\}$, we call it public-update cipher. Otherwise we call it secret-update cipher.

4.1 The BDPT Based Cube Attack on Public-update Cipher

Proposition 9. *Let $f_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ be a public-update cipher. Then, for cube set C_{I_v,J_v,K_v}, the exact superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be fully recovered by the propagation of BDPT.*

Proof. The superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ is a function of secret variables $\boldsymbol{x}$. If for arbitrary term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$, we can determine its ANF coefficient. Then, the exact superpoly can be obtained.

Because $f_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ is a public-update cipher, $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ is also a public-update cipher. Then, for arbitrary term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$, we will research the division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}^{1^{n+m}}_{\{1\},\emptyset}$, where $\mathbb{K} = \emptyset$ and $\mathbb{L} = \{(\boldsymbol{u}_{I_x}, \boldsymbol{u}_v) \,|\boldsymbol{u}_{I_v} \preceq \boldsymbol{u}_v \preceq \boldsymbol{u}_{I_v} \oplus \boldsymbol{u}_{J_v}\}$. Let the output BDPT of $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ be $\mathcal{D}^{1^{n+m}}_{\mathbb{K}',\mathbb{L}'}$. The initial $\mathbb{K} = \emptyset$ means that there is no division trail from $\mathbb{K} = \emptyset$ to $\mathbb{K}'$. From Sect.2.3, we know that for public function, the BDPT propagation of $\mathbb{K}$ and $\mathbb{L}$ is independent. Only when the secret round key is involved, some vectors of $\mathbb{L}$ will affect $\mathbb{K}$. That means, there is no division trail from $\mathbb{L}$ to $\mathbb{K}'$ when all the update functions are public. So there is no division trail such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}^{1^{n+m}}_{\{1\},\emptyset}$. According to (1) and (2) of Proposition 8, the term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$'s ANF coefficient in the superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ can be determined by the number of division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}^{1^{n+m}}_{\emptyset,\{1\}}$. □

First of all, we can use MILP method to obtain the involved key indices $I = \{i_1, i_2, \cdots, i_{|I|}\}$ and the degree of the superpoly d. By using Algorithm 3, our attack strategy to recover partial secret variables consists of two phases: *offline phase* and *online phase*

1. **Offline Phase (Superpoly Recovery).** For a cube C_{I_v,J_v,K_v}, attackers query Algorithm 3 to determine all $\sum_{i=0}^{d} \binom{|I|}{i}$ ANF coefficients $a_{\boldsymbol{u}}^{p_{I_v,J_v,K_v}}$ satisfying $hw(\boldsymbol{u}) \leq d$ and $\boldsymbol{u} \preceq \boldsymbol{u}_I$.

2. **Online Phase (Partial Key Recovery).** Attackers query the encryption oracle and acquire the exact value of $p_{I_v,J_v,K_v}(\boldsymbol{x})$ by summing over the cube C_{I_v,J_v,K_v} as Eq. (2). Then, one polynomial about involved secret variables can be got, and some values in involved secret variables are discarded.

Time Complexity. In order to recover the exact superpoly, we need to query the Algorithm 3 $\sum_{i=0}^{d} \binom{|I|}{i}$ times. The time complexity of Phase 1 is $c \cdot \sum_{i=0}^{d} \binom{|I|}{i}$, where c is the average computational complexity of Algorithm 3. Phase 2 requires $2^{|I_v|}$ encryptions. Therefore, the number of encryptions that an available attack requires is

$$max\left\{c \cdot \sum_{i=0}^{d} \binom{|I|}{i}, 2^{|I_v|}\right\} < 2^n \tag{7}$$

Compared with CBDP based cube attack in Sect.2.6, we can know that when $c < 2^{|I_v|}$, our method can obtain better results.

4.2 The BDPT Based Cube Attack on Secret-update Cipher

Due to the influence of secret keys in the intermediate rounds, new vectors may be generated from $\mathbb{L}_i$ and added to $\mathbb{K}_i$. Therefore, the condition that there are no division trails such that $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}} \xrightarrow{f_{I_x,I_v,J_v,K_v}} \mathcal{D}^{1^{n+m}}_{\{1\},\emptyset}$ may not hold. Namely, only a part of the ANF coefficients in superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ can be obtained by BDPT. In the offline phase, if

there are N ANF coefficients that cannot be determined by BDPT, we have to get their ANF coefficients by the method used in the CBDP based cube attack. Therefore, the number of encryptions that an available attack requires is

$$max\left\{c\cdot\sum_{i=0}^{d}\binom{|I|}{i}+N\cdot 2^{|I_v|},2^{|I_v|}\right\}<2^n.$$

5 Applications to Trivium

For public-update ciphers, the exact ANF of superpoly can be fully recovered by exploring the propagation of BDPT. Fortunately, many present stream ciphers are public-update ciphers. In order to verify the correctness and effectiveness of our method, we apply it to Trivium [1] which is a public-update cipher.

5.1 Descriptions of Trivium

Trivium [1] is a bit-oriented stream cipher. The internal state of Trivium, denoted by $\boldsymbol{s}=(s_1,s_2,\ldots,s_{288})$, is initialized by loading the 80-bit Key and 80-bit IV into three registers, and the other state bits are set to 0 except for the last three bits of the third register. Then, the algorithm would not output any keystream bit until the internal state is updated 1152 rounds. A complete description of Trivium is given by the following simple pseudo-code.

$(s_1,s_2,\ldots,s_{93})\leftarrow(K_1,\ldots,K_{80},0,\ldots,0)$
$(s_{94},s_{95},\ldots,s_{177})\leftarrow(IV_1,\ldots,IV_{80},0,\ldots,0)$
$(s_{178},s_{179},\ldots,s_{288})\leftarrow(0,\ldots,0,1,1,1)$
for $i=1$ **to** N **do**
  if $i>1152$ **then**
    $z_{i-1152}\leftarrow s_{66}\oplus s_{93}\oplus s_{162}\oplus s_{177}\oplus s_{243}\oplus s_{288}$
  end if
  $t_1\leftarrow s_{66}\oplus s_{91}\cdot s_{92}\oplus s_{93}\oplus s_{171}$
  $t_2\leftarrow s_{162}\oplus s_{175}\cdot s_{176}\oplus s_{177}\oplus s_{264}$
  $t_3\leftarrow s_{243}\oplus s_{286}\cdot s_{287}\oplus s_{288}\oplus s_{69}$
  $(s_1,s_2,\ldots,s_{93})\leftarrow(t_3,s_1,\ldots,s_{92})$
  $(s_{94},s_{95},\ldots,s_{177})\leftarrow(t_1,s_{94},\ldots,s_{176})$
  $(s_{178},s_{179},\ldots,s_{288})\leftarrow(t_2,s_{178},\ldots,s_{287})$
end for

5.2 The MILP-aided Algorithm for Trivium

To outline our technique more conveniently, we describe Trivium using the following expression. Let $\boldsymbol{x}=(x_1,x_2,\cdots,x_{80})$ denote the secret variables, and $\boldsymbol{v}=(v_1,v_2,\cdots,v_{208})$ denote the public variables. For public variables, $v_{14},v_{15},\cdots,v_{93}$ are the IV variables whose values can be chosen by attackers, $\{v_{206},v_{207},v_{208}\}$ are set to 1, and others are set to 0. Then, the output bit of Trivium can be seen as the function of $(\boldsymbol{x},\boldsymbol{v})$. The only nonlinear component of Trivium is a 2-degree core function denoted as $\boldsymbol{s'}=f_{core}(\boldsymbol{s},i_1,\ldots,i_5)$, where $i_1,\ldots,i_5$ are indices, and $\boldsymbol{s},\boldsymbol{s'}$ are 288-bit state satisfying

$$s'_i=\begin{cases}s_{i_1}s_{i_2}+s_{i_3}+s_{i_4}+s_{i_5}, & i=i_5\\ s_i, & \text{otherwise.}\end{cases}$$

The MILP model of core function can be represented as $CoreModel(\mathcal{M}, \boldsymbol{s}, i_1, i_2, i_3, i_4, i_5)$ in Algorithm 4. The input of *CoreModel* consists of $\mathcal{M}$ as the current MILP model, a vector $\boldsymbol{s}$ describing the current CBDP of the 288-bit state, and 5 indices i_1, i_2, i_3, i_4, i_5 corresponding to the input bits. Then, *CoreModel* outputs the updated model $\mathcal{M}$, and a vector $\boldsymbol{s}'$ describing the CBDP after f_{core}. With the definition of *CoreModel*, the MILP model of Trivium from the r-th round to R-th round can be represented as $TriviumModel(r, R, \boldsymbol{k}, \boldsymbol{k}.F)$ in Algorithm 4. The input of *TriviumModel* consist of the initial round number r, and the number R for the last round, the initial CBDP $\mathbb{K} = \{\boldsymbol{k}\}$ and its flag $\boldsymbol{k}.F$. Then, *TriviumModel* output the MILP model $\mathcal{M}$ and the flag value vector $\boldsymbol{s}^r.F$.

Algorithm 4: MILP model of CBDP for Trivium

procedure *CoreModel*($\mathcal{M}$, $\boldsymbol{s}$, i_1, i_2, i_3, i_4, i_5)
  $\big(\mathcal{M}, s'_{i_1}, z_1\big) \leftarrow$ **copyf**$(\mathcal{M}, s_{i_1})$
  $\big(\mathcal{M}, s'_{i_2}, z_2\big) \leftarrow$ **copyf**$(\mathcal{M}, s_{i_2})$
  $\big(\mathcal{M}, s'_{i_3}, z_3\big) \leftarrow$ **copyf**$(\mathcal{M}, s_{i_3})$
  $\big(\mathcal{M}, s'_{i_4}, z_4\big) \leftarrow$ **copyf**$(\mathcal{M}, s_{i_4})$
  $(\mathcal{M}, a) \leftarrow$ **andf**$(\mathcal{M}, z_1, z_2)$
  $\big(\mathcal{M}, s'_{i_5}\big) \leftarrow$ **xorf**$(\mathcal{M}, a, z_2, z_3, z_4, s_{i_5})$
  for all $i \in \{1, 2, \cdots, 288\} - \{i_1, i_2, i_3, i_4, i_5\}$ **do**
    $s'_i = s_i$
  end for
  return $(\mathcal{M}, \boldsymbol{s}')$
end procedure

procedure *TriviumModel*(r, R, $\boldsymbol{k}$, $\boldsymbol{k}.F$)
  prepare empty MILP model $\mathcal{M}$
  $\mathcal{M}.var \leftarrow s_i^{r-1}$ for $i \in \{1, 2, \ldots, 288\}$
  $s_i^{r-1} = k_i$ and $s_i^{r-1}.F = k_i.F$ for $i \in \{1, 2, \ldots, 288\}$
  for $i = r$ to R **do**
    $(\mathcal{M}, \boldsymbol{s}') = CoreModel\big(\mathcal{M}, \boldsymbol{s}^{i-1}, 66, 171, 91, 92, 93\big)$
    $(\mathcal{M}, \boldsymbol{s}'') = CoreModel(\mathcal{M}, \boldsymbol{s}', 162, 264, 175, 176, 177)$
    $(\mathcal{M}, \boldsymbol{s}''') = CoreModel(\mathcal{M}, \boldsymbol{s}'', 243, 69, 286, 287, 288)$
    $\boldsymbol{s}^i = \boldsymbol{s}''' \ggg 1$
  end for
  $\mathcal{M}.con \leftarrow \big(x_{66}^R + s_{93}^R + s_{162}^R + s_{177}^R + s_{243}^R + s_{288}^R\big) = 1$
  return $(\mathcal{M}, s^r.F)$
end procedure

Because Trivium is a public-update cipher, during the progress of recovering the ANF coefficients of superpoly, the set $\mathbb{K}$ is always empty. We only show the propagation of $\mathbb{L}$ in Algorithm 5. The input of procedure *RoundPropagation* in Algorithm 5 is the r-th round BDPT $\mathbb{L}_r$, and the outputs is the $(r+1)$-th round BDPT $\mathbb{L}_{r+1}$. Finally, in order to recover the ANF coefficients of superpoly in R-round Trivium, we proposed Algorithm 6 which is the instantiation of Algorithm 3.

5.3 Experimental Verification

All the experiments are conducted on the following platform: Intel Core i5-4590 CPU @3.3GHz, 8.00G RAM. And the optimizer we used to search integral distinguishers is Gurobi 8.1.0 [8]. Identical to [18], we firstly use the cube $I_v = \{14, 24, 34, 44, 54, 64, 74, 84\}$ to verify our attacks and implementations.

Algorithm 5: The propagation of $\mathbb{L}$ for the round function

procedure *CorePropagation*$(\mathbb{L}, i_1, i_2, i_3, i_4, i_5)$
  Let $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)$ be the variables
  Let $\boldsymbol{y}$ be the function of $\boldsymbol{x}$, and $\boldsymbol{y} = (x_1, x_2, x_3, x_4, x_1x_2 + x_3 + x_4 + x_5)$
  $\mathbb{L}' = \emptyset$
  for $\boldsymbol{\ell}$ in $\mathbb{L}$
    for all $\boldsymbol{u} = (u_1, u_2, u_3, u_4, u_5) \in \mathbb{F}_2^5$ **do**
      if $\boldsymbol{y^u}$ contains the term $\boldsymbol{x}^{(\ell_{i_1}, \ell_{i_2}\ell_{i_3}, \ell_{i_4}, \ell_{i_5})}$ **then**
        $\boldsymbol{\ell'} = \boldsymbol{\ell}$
        $\ell'_{i_1} = u_1$, $\ell'_{i_2} = u_2$, $\ell'_{i_3} = u_3$, $\ell'_{i_4} = u_4$, $\ell'_{i_5} = u_5$
        $\mathbb{L}' \xleftarrow{x} \boldsymbol{\ell'}$
      end if
    end for
  end for
  return $\mathbb{L}'$
end procedure

procedure *RoundPropagation*$(\mathbb{L}_r)$
  initial $\mathbb{L}' = \emptyset$, $\mathbb{L}'' = \emptyset$, $\mathbb{L}''' = \emptyset$, $\mathbb{L}_{r+1} = \emptyset$
  $\mathbb{L}' =$*CorePropagation*$(\mathbb{L}_r, 66, 171, 91, 92, 93)$
  $\mathbb{L}'' =$*CorePropagation*$(\mathbb{L}', 162, 164, 175, 176, 177)$
  $\mathbb{L}''' =$*CorePropagation*$(\mathbb{L}'', 243, 69, 286, 287, 288)$
  for all $\boldsymbol{\ell}$ in $\mathbb{L}'''$ **do**
    $\mathbb{L}_{r+1} = \mathbb{L}_{r+1} \bigcup \{\boldsymbol{\ell} \ggg 1\}$
  end for
  return $\mathbb{L}_{r+1}$
end procedure

Example 1. For 591-round Trivium, when the cube is C_{I_v,J_v,K_v}, where $I_v = \{14, 24, 34, 44, 54, 64, 74, 84\}$, $J_v = \{15, 30, 33, 206, 207, 208\}$, and $K_v = \{1, 2, \cdots, 208\} - I_v - J_v$, we can get that the involved secret variables are $\{x_{23}, x_{24}, x_{25}, x_{67}\}$, the degree of superpoly is not larger than 2. Then, we use Algorithm 6 to recover all the ANF coefficients of the superpoly, which is in accordance with the following superpoly recovered by practical

$$p_{I_v,J_v,K_v}(\boldsymbol{x}) = x_{67} + x_{25} + x_{24}x_{23} + 1.$$

Example 2. For 591-round Trivium, when the cube is C_{I_v,J_v,K_v}, where $I_v = \{14, 24, 34, 44, 54, 64, 74, 84\}$, $J_v = \{14, 15, \cdots, 93, 206, 207, 208\} - I_v$, and $K_v = \{1, 2, \cdots, 208\} - I_v - J_v$, we can get that the involved secret variables are $\{x_{23}, x_{24}, x_{25}, x_{66}, x_{67}\}$, the degree of superpoly is not larger than 3. Then, we use Algorithm 6 to recover the superpoly, which is in accordance with the following superpoly recovered by practical

$$p_{I_v,J_v,K_v}(\boldsymbol{x}) = x_{66}x_{24}x_{23} + x_{66}x_{25} + x_{67}x_{66} + x_{66}.$$

To further confirm the correctness of our method, we use the cube above and conduct practical experiments on different rounds, namely 576,577,587,590 (selected from Table 2 of [18]). The superpolies got by our method are all in accordance with the real superpolies.

At CRYPTO 2017 [18], Todo *et al.* proposed a CBDP based cube attack on the 832-round Trivium. Then, at CRYPTO 2018 [19], Wang *et al.* improved the result and presented a CBDP based cube attack on 839-round Trivium. But both methods cannot ensure whether the cube attacks are key recovery attacks or not. After applying Algorithm 6 to the 832-round and 839-round Trivium, we have the following results.

Algorithm 6: Recover the ANF coefficient of superpoly in R-round Trivium

procedure *TriviumRecover*(R, I_x, I_v, $J_v^{I_x}$,$uhw_{I_x,I_v,J_v^{I_x}}$)
  Initial $\mathbb{K}_0 = \emptyset$
  Initial $\mathbb{L}_0 = \left\{(\boldsymbol{u}_{I_x}, \boldsymbol{u}_v) \,|\boldsymbol{u}_{I_v} \preceq \boldsymbol{u}_v \preceq \boldsymbol{u}_{I_v} \oplus \boldsymbol{u}_{J_v}^{I_x} \text{ and } hw\left(\boldsymbol{u}_{I_x}, \boldsymbol{u}_v\right) \leq uhw_{I_x,I_v,J_v^{I_x}}\right\}$
  Assign $s_i^0.F = \delta$ for $i \in I_x$
  Assign $s_i^0.F = 0$ for $i \in \{1, 2, \ldots, 93\} - I_x$
  Assign $s_{80+i}^0.F = \delta$ for $i \in I_v$
  Assign $s_{80+i}^0.F = 1$ for $i \in J_v^{I_x}$
  Assign $s_{80+i}^0.F = 0$ for $\{1, 2, \ldots, 208\} - I_v - J_v^{I_x}$
  for $(r = 1; r \leq R; i++)$
    Initial $\mathbb{L}' = \emptyset$
    for $\boldsymbol{\ell}$ in $\mathbb{L}_{r-1}$ **do**
      $(\mathcal{M}, s^r.F) = TriviumModel\left(r, R, \boldsymbol{\ell}, s^{r-1}.F\right)$
      if $\mathcal{M}$ is feasible
        $\mathbb{L}' = \mathbb{L}' \bigcup \boldsymbol{\ell}$
      end if
    end for
    if $\mathbb{L}' = \emptyset$
      return 0
    end if
    $\mathbb{L}^r = RoundPropagation\left(\mathbb{L}'\right)$
  end for
    return 1
end procedure

Result 1. For the cube set C_{I_v,J_v,K_v}, where $I_v = \{14, \ldots, 46, 48, \ldots, 59, 61, \ldots, 93\}$, no matter what the assignment to the non-cube IVs $\{47, 60\}$ is, the corresponding superpoly of 839-round Trivium in the paper [19] is constant. So the cube attack based on CBDP in the paper [19] is not key recovery attack.

Result 2. For the cube set C_{I_v,J_v,K_v}, where $I_v = \{14, 15, \ldots, 78, 80, 82, \ldots, 92\}$, the superpolies of some assignments to the non-cube IVs $\{79, 81, 83, 85, 87, 89, 91, 93\}$ are constant. For example, when $J_v = \{206, 207, 208\}$ and $K_v = \{1, 2, \ldots, 208\} - I_v - J_v$, the superpoly recovered is $p_{I_v,J_v,K_v}(\boldsymbol{x}) = 0$. And the superpolies of some assignments to the non-cube IVs $\{79, 81, 83, 85, 87, 89, 91, 93\}$ are non-constant. For example, when $J_v = \{81, 91, 206, 207, 208\}$ and $K_v = \{1, 2, \ldots, 208\} - I_v - J_v$, the superpoly recovered is $p_{I_v,J_v,K_v}(\boldsymbol{x}) = x_{57}x_{58}x_{59} + x_{33}x_{57} + x_{57}x_{60}$. In a word, the assignment to the non-cube IVs will affect whether the cube attack on 832-round Trivium in the paper [18] is key recovery attack or not.

5.4 Theoretical Results

From [19], we know that superpoly has sparse monomial distribution properties and many ANF coefficients can be determined by CBDP method. For the remaining coefficients, we need query Algorithm 6. In our experiments, the running time of Algorithm 6 on a common PC (Intel Core i5-4590 CPU @3.3GHz, 8.00G RAM) is practical. For example, it spends about 10 days to recover all the $\sum_{i=0}^{3}\binom{5}{i} = 26$ ANF coefficients of superpoly in Result 2.

Result 3. Let C_{I_v,J_v,K_v} be a cube set, where $I_v = \{14, 15, \ldots, 92\}$, $J_v = \{206, 207, 208\}$, and $K_v = \{1, 2, \ldots, 205\} - I_v$. Using the MILP method, we can get that the degree of superpoly in 842-round Trivium is not larger than 7. Then, we have $\sum_{i=0}^{d} \binom{|I|}{i} \leq \sum_{i=0}^{7} \binom{80}{i} \leq 2^{32}$. That means we can use no more than 2^{32} MILP propagation of BDPT to recover the exact superpoly of 842-round Trivium.

6 Conclusion

In this paper, we propose a new method to recover exact superpoly in cube attack. Our method is developed from BDPT, and as far as we know, this is the first application of BDPT to stream ciphers. For public-update ciphers, the exact ANF of superpoly can be fully recovered by exploring the propagation of BDPT. Fortunately, many present stream ciphers are public-update ciphers. To verify the correctness and effectiveness of our method, we apply it to Trivium. For the cube attack on the 832-round Trivium [18], we obtain that only some proper non-cube IV assignments can obtain non-constant superpolies. And the complexity of recovering the superpoly is reduced from 2^{77} to practical. For the cube attack on 839-round Trivium [19], our result shows that the superpoly is always constant. Because our method can determine the ANF coefficients of superpoly in practical time, we propose a theoretical cube attack on 842-round Trivium.

For secret-update ciphers, due to the influence of intermediate round keys, not all the ANF coefficients can be obtained by BDPT. From this perspective, when we design stream ciphers, the secret-update ciphers are more secure. How to recover the superpoly of secret-update ciphers is our future work.

Acknowledgements The authors would like to thank the anonymous reviewers for their detailed comments and suggestions. This work was supported by the National Natural Science Foundation of China [Grant No.61572516, 61802437].

References

[1] De Cannière, C., Preneel, B.: Trivium. In: Robshaw, M., Billet O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)

[2] Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278-299. Springer, Heidelberg (2009)

[3] Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011)

[4] Englund, H., Johansson, T., Sönmez Turan, M.: A framework for chosen IV statistical analysis of stream ciphers. In Srinathan, K., Rangan, C.P., Yung, M., (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268–281. Springer, Heidelberg (2007)

[5] Fischer, S., Khazaei, S., Meier, W.: Chosen IV statistical analysis for key recovery attacks on stream ciphers. In Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)

[6] Fouque, PA., Vannet, T.: Improving key recovery to 784 and 799 rounds of Trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2013)

[7] Fu, X., Wang, X., Dong, X., Meier, W.: A key-recovery attack on 855-round Trivium. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 160–184. Springer, Cham (2018)

[8] Gurobi: http://www.gurobi.com/

[9] Hao, Y., Jiao, L., Li, C., Meier, W., Todo, Y., Wang, Q.: Observations on the dynamic cube attack of 855-Round TRIVIUM from Crypto'18. IACR Cryptology ePrint Archive 2018:972 (2018). https://eprint.iacr.org/2018/972.pdf

[10] Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round Keccak sponge function. In: Coron, JS., Nielsen, J. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 259–288. Springer, Cham (2017)

[11] Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) CC. SISECS, vol. 276, pp. 227–233. Springer, Heidelberg (1994)

[12] Liu, M., Yang, J., Wang, W., Lin, D.: Correlation cube attacks: From weak-key distinguisher to key recovery. In: Nielsen, J., Rijmen, V. (eds.) EUROCRYPT 2018. vol. 10821, pp. 715–744. Springer, Cham (2018)

[13] Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Katz, J., Shacham, H., (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 227–249. Springer, Cham (2017)

[14] Mroczkowski, P., Szmidt, J.: The cube attack on stream cipher Trivium and quadraticity tests. Fundam. Inform. 114(3-4), 309–318 (2012).

[15] Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015)

[16] Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016)

[17] Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015)

[18] Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017)

[19] Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting low degree property of superpoly. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 275–305. Springer, Cham (2018)

[20] Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: MILP method of searching integral distinguishers based on division property using three subsets. IACR Cryptology ePrint Archive, 2018:1186 (2018). https://eprint.iacr.org/2018/1186.pdf

[21] 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., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016)

[22] Ye, C., Tian, T.: Deterministic cube attacks. IACR Cryptology ePrint Archive, 2018:1028 (2018). https://eprint.iacr.org/2018/1082.pdf