MILP-aided Method of Searching Division Property Using Three Subsets and Applications

Senpeng Wang(✉), Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi

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

Abstract. Division property is a generalized integral property proposed by Todo at EUROCRYPT 2015, and then conventional bit-based division property (CBDP) and bit-based division property using three subsets (BDPT) were proposed by Todo and Morii at FSE 2016. At the very beginning, the two kinds of bit-based division properties once couldn't be applied to ciphers with large block size just because of the huge time and memory complexity. At ASIACRYPT 2016, Xiang *et al.* extended Mixed Integer Linear Programming (MILP) method to search integral distinguishers based on CBDP. BDPT can find more accurate integral distinguishers than CBDP, but it couldn't be modeled efficiently.
This paper focuses on the feasibility of searching integral distinguishers based on BDPT. We propose the pruning techniques and fast propagation of BDPT for the first time. Based on these, an MILP-aided method for the propagation of BDPT is proposed. Then, we apply this method to some block ciphers. For SIMON64, PRESENT, and RECTANGLE, we find more balanced bits than the previous longest distinguishers. For LBlock, we find a better 16-round integral distinguisher with less active bits. For other block ciphers, our results are in accordance with the previous longest distinguishers.
Cube attack is an important cryptanalytic technique against symmetric cryptosystems, especially for stream ciphers. And the most important step in cube attack is superpoly recovery. Inspired by the CBDP based cube attack proposed by Todo at CRYPTO 2017, we propose a method which uses BDPT to recover the superpoly in cube attack. We apply this new method to round-reduced 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 841 round.

Keywords: Integral distinguisher · Division property · MILP · Block cipher · Cube attack · Stream cipher.

1 Introduction

Division property, a generalization of integral property [11], was proposed by Todo at EUROCRYPT 2015 [22]. 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 [20], Todo applied this new technique to MISTY1 and achieved the first theoretical cryptanalysis of the full-round MISTY1. Sun *et al.* [18], revisited division property, and they studied the property of a set (multiset) satisfying certain division property. At CRYPTO 2016 [4], Boura and Canteaut introduced a new notion called *parity set* to exploit division property. They formulated and characterized the division property of S-box and found better integral distinguisher of PRESENT [3]. But it required large time and memory complexity. To solve this problem, Xie and Tian [28] proposed another concept called *term set*, based on which they found a 9-round distinguisher of PRESENT with 22 balanced bits.

In order to exploit the concrete structure of round function, Todo and Morii [21] 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 of SIMON32 while BDPT found the 15-round integral distinguisher of SIMON32 [21].

Although CBDP and BDPT could find accurate integral distinguishers, the huge complexity once restricted their wide applications. At ASIACRYPT 2016, Xiang *et al.* [27] applied MILP method to search integral distinguishers based on CBDP, which allowed them to analyze block ciphers with large sizes. But there was still no MILP method to model the propagation of BDPT.

Cube attack, proposed by Dinur and Shamir [6] at EUROCRYPT 2009, is one of the general cryptanalytic techniques against symmetric cryptosystems. For a cipher with n secret variables $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and m public variables $\boldsymbol{v} = (v_0, v_1, \ldots, v_{m-1})$, the output bit can be denoted as a polynomial $f(\boldsymbol{x}, \boldsymbol{v})$. The core idea of cube attack is to simplify $f(\boldsymbol{x}, \boldsymbol{v})$ 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 called *superpoly*. In the original paper of cube attack [6], the authors regarded stream cipher as a blackbox polynomial and introduced a linearity test to recover superpoly. Recently, many variants of cube attacks were put forward such as dynamic cube attacks [7], conditional cube attacks [14], correlation cube attacks [15], CBDP based cube attacks [23, 26], and deterministic cube attacks [30].

At EUROCRYPT 2018 [15], Liu *et al.* proposed *correlation cube attack*, which could mount to 835-round Trivium using small dimensional cubes. Then, in [30], 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 [16]. 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 corre-

sponding 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. Moreover, at CRYPTO 2018 [9], Fu *et al.* proposed a key recovery attack on 855-round Trivium which somewhat resembled dynamic cube attacks. For the attack in [9], the paper [12] pointed out that there was possibility that the correct key guesses and the wrong ones shared the same zero-sum property. It means that the key recovery attack may degenerate to distinguish attack.

It is noticeable that, at CRYPTO 2017 [23], 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 could evaluate the algebraic normal form (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 [26], Wang *et al.* improve the CBDP based cube attack and gave a 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 cannot ensure that the superpoly of a cube is non-constant. Hence the key recovery attack may be just a distinguish attack. BDPT can exploit the integral distinguisher whose sum is 1, which means BDPT may show a determined key recovery attack. However, compared with the propagation of CBDP, the propagation of BDPT is more complicated and cannot be modeled by MILP method directly. An automatically searching for a variant three-subset division property with STP solver was proposed in [13], but the variant is weaker than the original BDPT. How to trace the propagation of BDPT is an open problem.

1.1 Our Contributions

In this paper, we propose an MILP-aided method for BDPT. Then, we apply it to search integral distinguishers of block ciphers and recover superpolies of stream ciphers.

1.1.1 MILP-aided Method for BDPT

Pruning Properties of BDPT. When we evaluate the propagation of BDPT, there may be some vectors that have no impact on the BDPT of output bit. So we show the pruning properties when the vectors of BDPT can be removed.

Fast Propagation and Stopping Rules. Inspired by the "lazy propagation" in [21], we propose the notion of "fast propagation" which can translate BDPT into CBDP and show some bits are balanced. Then, based on "lazy propagation" and "fast propagation", we obtain three stopping rules. Finally, an MILP-aided method for the propagation of BDPT is proposed.

1.1.2 Searching Integral Distinguishers of Block Ciphers

We apply our MILP-aided method to search integral distinguishers of some block ciphers. The main results are shown in Table 1.

ARX Ciphers. For SIMON32, we find the 15-round integral distinguisher that cannot be found by CBDP. For 18-round SIMON64, we find 23 balanced bits which has one more bit than the previous longest integral distinguisher.

SPN Ciphers. For PRESENT, when the input data is 2^{60}, our method can find 3 more balanced bits than the previous longest integral distinguisher. Moreover, when the input data is 2^{63}, the integral distinguisher we got has 6 more balanced bits than that got by term set in the paper [28]. For RECTANGLE, when the input data is 2^{60}, our method can also obtain 11 more balanced bits than the previous longest 9-round integral distinguisher.

Generalized Feistel Cipher. For LBlock, we obtain a 17-round integral distinguisher which is the same with the previous longest integral distinguisher. Moreover, a better 16-round integral distinguisher with less active bits can also be obtained.

1.1.3 Recovering Superpoly of Stream Cipher

Using BDPT to Recover the ANF Coefficient of Superpoly. Inspired by the CBDP based cube attack in [23, 26], our new method is based on the propagation of BDPT which can find integral distinguisher whose sum is 0 or 1. But it's nontrivial to recover the superpoly by integral distinguishers based on BDPT. Therefore, we proposed the notion of *similar polynomial*. We can recover the ANF coefficient of superpoly by researching the BDPT propagation of corresponding similar polynomial. 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.

Application to Trivium. In order to verify the correctness and effectiveness of our method, we apply BDPT to recover the superpoly of round-reduced Trivium which is a public cipher. 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 841 round. The detailed information is shown in Table 2. And 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.

Table 1. Summarization of integral distinguishers

Cipher	Data	Round	Number of balanced bits	Time	Reference
SIMON32	2^{31}	15	3		[21]
		15	3	2m	Sect. 5.1
SIMON64	2^{63}	18	22	6.7m	[27]
		18	**23**	1h41m	Sect. 5.1
PRESENT	2^{60}	9	1	3.4m	[27]
		9	**4**	56m	Sect. 5.2
	2^{63}	9	22		[28]
		9	**28**	10m	Sect. 5.2
RECTANGLE	2^{60}	9	16	4.1m	[27]
		9	**27**	10m	Sect. 5.2
LBlock	2^{63}	16	32	4.9m	[27]
		17	4		[8]
		17	4	10h25m	Sect. 5.3
	$\mathbf{2^{62}}$	16	18	6h49m	Sect. 5.3

Table 2. Superpoly recovery of Trivium

Rounds	Cube size	Exact superpoly	Complexity	Reference
832	72	yes	2^{77}	[23]
			$2^{76.7}$	[26]
			practical	Sect. 7.3
835	36/37	no		[15]
838	37	yes	practical	[30]
839	78	yes	2^{79}	[26]
			practical	Sect. 7.3
841	78	yes	$2^{41} \cdot c$	Sect. 7.4

1.2 Outline of the Paper.

This paper is organized as follows: Sect. 2 provides the background of MILP, division property, and cube attacks etc. In Sect. 3, some new propagation properties of BDPT are given. In Sect. 4, we propose an MILP-aided method for BDPT. Sect. 5 shows applications to block ciphers. In Sect. 6, we use BDPT to recover the superpoly in cube attack. Sect. 7 shows the application to Trivium. Sect. 8 concludes the paper. Some auxiliary materials are supplied in Appendix.

2 Preliminaries

2.1 Notations

Let $\mathbb{F}_2$ denote the finite field $\{0,1\}$ and $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_2^n$ be an n-bit vector, where a_i denotes the i-th bit of $\boldsymbol{a}$. For n-bit vectors $\boldsymbol{x}$ and $\boldsymbol{u}$, define $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i^{u_i}$. Then, for any $\boldsymbol{k} \in \mathbb{F}_2^n$ and $\boldsymbol{k'} \in \mathbb{F}_2^n$, define $\boldsymbol{k} \succeq \boldsymbol{k'}$ if

$k_i \geq k'_i$ holds for all $i = 0, 1, \ldots, n-1$ and define $\boldsymbol{k} \succ \boldsymbol{k'}$ if $k_i > k'_i$ holds for all $i = 0, 1, \ldots, n-1$. For a subset $I \subset \{0, 1, \ldots, n-1\}$, $\boldsymbol{u}_I$ denotes an n-dimensional bit vector $(u_0, u_1, \ldots, u_{n-1})$ satisfying $u_i = 1$ if $i \in I$ and $u_i = 0$ otherwise. We simply write $\mathbb{K} \leftarrow \boldsymbol{k}$ when $\mathbb{K} := \mathbb{K} \cup \{\boldsymbol{k}\}$ and $\mathbb{K} \rightarrow \boldsymbol{k}$ when $\mathbb{K} := \mathbb{K} \setminus \{\boldsymbol{k}\}$. And $|\mathbb{K}|$ denotes the number of elements in the set $|\mathbb{K}|$.

2.2 Mixed Integer Linear Programming

MILP is a kind of optimization or feasibility program whose objective function and constraints are linear, and the variables are restricted to be integers. Generally, an MILP model $\mathcal{M}$ consists of variables $\mathcal{M}.var$, constrains $\mathcal{M}.con$, and the objective function $\mathcal{M}.obj$. MILP models can be solved by solver like Gurobi [10]. If there is no feasible solution, the solver will returns *infeasible*. When there is no objective function in $\mathcal{M}$, the MILP solver will only return whether $\mathcal{M}$ is feasible or not.

2.3 Bit-based Division Property

Two kinds of bit-based division property (CBDP and BDPT) were introduced by Todo and Morii at FSE 2016 [21]. In this subsection, we will briefly introduce them and their propagation rules.

Definition 1. (CBDP [21]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the CBDP $\mathcal{D}_{\mathbb{K}}^{1^n}$, 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:*

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

Definition 2. (BDPT [21]). *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:*

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

According to [21], 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. We denote this progress as $\boldsymbol{Reduce0}(\mathbb{K})$. If there are $\boldsymbol{\ell} \in \mathbb{L}$ and $\boldsymbol{k} \in \mathbb{K}$ satisfying $\boldsymbol{\ell} \succeq \boldsymbol{k}$, the vector $\boldsymbol{\ell}$ can also be removed from $\mathbb{L}$. We denote this progress as $\boldsymbol{Reduce1}(\mathbb{K}, \mathbb{L})$. For any $\boldsymbol{u}$, the redundant vectors in $\mathbb{K}$ and $\mathbb{L}$ will not affect the value of $\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$.

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 [21].

BDPT Rule 1 (Copy [21]). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be a copy function, where $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$, and the output is calculated as $\boldsymbol{y} = (x_0, x_0, x_1, \ldots, x_{n-1})$. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^{n+1}}$, where*

$$\mathbb{K}' \leftarrow \begin{cases} (0, 0, k_1, \ldots, k_{n-1}), & \text{if } k_0 = 0 \\ (1, 0, k_1, \ldots, k_{n-1}), (0, 1, k_1, \ldots, k_{n-1}), & \text{if } k_0 = 1 \end{cases},$$

$$\mathbb{L}' \leftarrow \begin{cases} (0, 0, \ell_1, \ldots, \ell_{n-1}), & \text{if } \ell_0 = 0 \\ (1, 0, \ell_1, \ldots, \ell_{n-1}), (0, 1, \ell_1, \ldots, \ell_{n-1}), (1, 1, \ell_1, \ldots, \ell_{n-1}), & \text{if } \ell_0 = 1 \end{cases},$$

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

BDPT Rule 2 (And [21]). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be a function compressed by an And, where the input $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$, and the ouput is calculated as $\boldsymbol{y} = (x_0 \wedge x_1, x_2, \ldots, x_{n-1}) \in \mathbb{F}_2^{n-1}$. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^{n-1}}$, where $\mathbb{K}'$ is computed from all $\boldsymbol{k} \in \mathbb{K}$ as*

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_0 + k_1}{2} \right\rceil, k_2, \ldots, k_{n-1} \right),$$

and $\mathbb{L}'$ is computed from all $\boldsymbol{\ell} \in \mathbb{L}$ satisfying $(\ell_0, \ell_1) = (0,0)$ or $(1,1)$ as

$$\mathbb{L}' \leftarrow \left(\left\lceil \frac{\ell_0 + \ell_1}{2} \right\rceil, \ell_2, \ldots, \ell_{n-1} \right).$$

BDPT Rule 3 (Xor [21]). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be a function compressed by an Xor, where the input $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$, and the output is calculated as $\boldsymbol{y} = (x_0 \oplus x_1, x_2, \ldots, x_{n-1}) \in \mathbb{F}_2^{n-1}$. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^{n-1}}$, where $\mathbb{K}'$ is computed from all $\boldsymbol{k} \in \mathbb{K}$ satisfying $(k_0, k_1) = (0,0), (1,0),$ or $(0,1)$ as*

$$\mathbb{K}' \leftarrow (k_0 + k_1, k_2, \ldots, k_{n-1}),$$

$\mathbb{L}'$ is computed from all $\boldsymbol{\ell} \in \mathbb{L}$ satisfying $(\ell_0, \ell_1) = (0,0)$, $(1,0)$, or $(0,1)$ as

$$\mathbb{L}' \xleftarrow{x} (\ell_0 + \ell_1, \ell_2, \ldots, \ell_{n-1}).$$

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 Key [21]). *Let $\mathbb{X}$ be the input multiset satisfying $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$. For the input $\boldsymbol{x} \in \mathbb{X}$, the output $\boldsymbol{y} \in \mathbb{Y}$ is computed as $\boldsymbol{y} = (x_0, \ldots, x_{i-1}, x_i \oplus r_k, x_{i+1}, \ldots, x_{n-1})$, where r_k is the secret key. Then, the output multiset $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^n}$, where $\mathbb{K}'$ and $\mathbb{L}'$ are computed as*

$$\begin{aligned} &\mathbb{L}' \leftarrow \boldsymbol{\ell}, \text{ for } \boldsymbol{\ell} \in \mathbb{L}, \\ &\mathbb{K}' \leftarrow \boldsymbol{k}, \text{ for } \boldsymbol{k} \in \mathbb{K}, \\ &\mathbb{K}' \leftarrow (\ell_0, \ell_1, \ldots, \ell_i \vee 1, \ldots, \ell_{n-1}), \text{ for } \boldsymbol{\ell} \in \mathbb{L} \text{ satisfying } \ell_i = 0. \end{aligned}$$

CBDP Rule 5 (S-box [4, 27]). *Let $\boldsymbol{y} = f(\boldsymbol{x})$ be a function of S-box, where the input $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_2^n$, and the output $\boldsymbol{y} = (y_0, y_1, \ldots, y_{m-1}) \in \mathbb{F}_2^m$. Then, every $y_i, i \in \{0, 1, \ldots, m-1\}$ can be expressed as a Boolean function of $(x_0, \ldots, x_{n-1})$. For the input CBDP $\mathbb{K}$, the output CBDP $\mathbb{K}'$ is a set of vectors as follows:*

$$\mathbb{K}' = \{\boldsymbol{u}' \in \mathbb{F}_2^m | \text{ for any } \boldsymbol{u} \in \mathbb{K}, \text{ if } \boldsymbol{y}^{\boldsymbol{u}'} \text{ contains any term } \boldsymbol{x}^{\boldsymbol{v}} \text{ satisfying } \boldsymbol{v} \succeq \boldsymbol{u}\}.$$

When there was no effective way to model the propagation of BDPT, Todo and Morii [21] proposed the notion of 'lazy propagation" to give the provable security of SIMON family against BDPT.

Definition 3. (Lazy Propagation [21]). *Let $D^{1^n}_{\mathbb{K}_i, \mathbb{L}_i}$ be the input BDPT of the i-th round function and $D^{1^n}_{\overline{\mathbb{K}}_{i+1}, \overline{\mathbb{L}}_{i+1}}$ be the BDPT from the lazy propagation. Then, $\overline{\mathbb{K}}_{i+1}$ is computed from only a part of vectors in $\mathbb{K}_i$, and $\overline{\mathbb{L}}_{i+1}$ always becomes the empty set $\emptyset$. Therefore, if the lazy propagation creates $\mathcal{D}^{1^n}_{\overline{\mathbb{K}}_r, \emptyset}$, where $\overline{\mathbb{K}}_r$ has n distinct vectors whose Hamming weight is one, the accurate propagation also creates the same n distinct vectors in the same round.*

2.4 The MILP Representation of CBDP

For an r-round iterative cipher of size n, attackers determine indices set $I = \{i_0, i_1, \ldots, i_{|I|-1}, \} \subset \{0, 1, \ldots, n-1\}$ and prepare $2^{|I|}$ chosen plaintexts where variables indexed by I are taking all possible combinations of values and the other variables are set to constants. The CBDP of such chosen plaintexts is $\mathcal{D}^{1^n}_{\mathbb{K}_0 = \{\boldsymbol{k}_I\}}$. Based on the propagation rules, the propagation of CBDP from $\mathcal{D}^{1^n}_{\{\boldsymbol{k}_I\}}$ can be evaluated as $\{\boldsymbol{k}_I\} \stackrel{def}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \cdots \to \mathbb{K}_r$, where $\mathcal{D}^{1^n}_{\mathbb{K}_r}$ is the CBDP after r-round propagation. If the set $\mathbb{K}_r$ doesn't have the unit vector $\boldsymbol{e}_m \in \mathbb{F}_2^n$ whose only m-th element is 1, the m-th output bit of r-round ciphertexts is balanced. At ASIACRYPT 2016, Xiang *et al.* [27] applied MILP method to the propagation of CBDP. They first introduced the concept of CBDP trail, which is defined as follows.

Definition 4. (CBDP Trail [27]). *Let us consider the propagation of the CBDP $\{\boldsymbol{k}_I\} \stackrel{def}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \cdots \to \mathbb{K}_r$. 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 CBDP trail.*

In [27], the authors modeled CBDP propagations of basic operations (Copy, Xor, And) and S-box by linear inequalities. Therefore, they could build an MILP model to cover all the possible CBDP trails generated from a given initial CBDP. Here, we introduce the MILP models for **Copy**, **Xor**, **And** and S-box.

Model 1 (Copy [27]). *Let* $a \xrightarrow{Copy} (b_0, b_1, \ldots, b_{n-1})$ *be a CBDP trail of Copy. The following inequalities are sufficient to describe its CBDP propagation*

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

Model 2 (Xor [27]). *Let* $(a_0, a_1, \ldots, a_{n-1}) \xrightarrow{Xor} b$ *be a division trail of Xor. The following inequalities are sufficient to describe its CBDP propagation*

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

Model 3 (And [27]). *Let* $(a_0, a_1, \ldots, a_{n-1}) \xrightarrow{And} b$ *be a division trail of And. The following inequalities are sufficient to describe its CBDP propagation*

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

Model 4 (S-box [27]). The CBDP Rule 5 in Sect. 2.3 can generate the CBDP propagation property of S-box. Then, we can using the **inequality_generator()** function in Sage software [17] to get a set of linear inequalities. Sometimes the number of linear inequalities in the set is large. Thus, some Greedy Algorithms [1, 19] were proposed to reduced this set.

2.5 Cube Attack

Cube attack was proposed by Dinur and Shamir at EUROCRYPT 2009 [6]. For a cipher with n secret variables $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and m public variables $\boldsymbol{v} = (v_0, v_1, \ldots, v_{m-1})$, the output bit can be represented as $f(\boldsymbol{x}, \boldsymbol{v})$. Attackers determine an indices subset $I_v = \{i_0, i_1, \ldots, i_{|I_v|-1}\} \subset \{0, 1, \ldots, m-1\}$, 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 $p(\boldsymbol{x}, \boldsymbol{v})$ is called the *superpoly* of C_{I_v, J_v, K_v} in $f(\boldsymbol{x}, \boldsymbol{v})$, and every term in $q(\boldsymbol{x}, \boldsymbol{v})$ misses at least one variable from $\{v_{i_0}, v_{i_1}, \ldots, v_{i_{|I_v|-1}}\}$.

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 $J_v \subset \{0, 1, \ldots, m-1\} - I_v$ are set to constant 1, and public variables indexed by $K_v = \{0, 1, \cdots, m-1\} - I_v - J_v$ are set to constant 0. Just as follows

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

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

$$\bigoplus_{\boldsymbol{v} \in C_{I_v, J_v, K_v}} f(\boldsymbol{x}, \boldsymbol{v}) = p_{I_v, J_v, K_v}(\boldsymbol{x}). \quad (2)$$

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 set C_{I_v, J_v, K_v} to get the equation with secret variables.

2.6 The Cube Attack Based on CBDP

At CRYPTO 2017 [23], Todo *et al.* successfully applied CBDP to cube attack. They use CBDP to analyze the ANF coefficients of superpoly.

Lemma 1. [23] *Let* $f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{u}\in\mathbb{F}_2^n} a_{\boldsymbol{u}}^f \cdot \boldsymbol{x}^{\boldsymbol{u}}$ *be a polynomial from* $\mathbb{F}_2^n$ *to* $\mathbb{F}_2$ *and* $a_{\boldsymbol{u}}^f \in \mathbb{F}_2$ *be the ANF coefficients. Let* $\boldsymbol{k}$ *be an* n*-dimensional bit vector. If there is no CBDP trail such that* $\boldsymbol{k} \xrightarrow{f} 1$*, then* $a_{\boldsymbol{u}}^f$ *is always 0 for* $\boldsymbol{u} \succeq \boldsymbol{k}$ *.*

Proposition 1. [23] *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 an* n*-bit unit vector whose only* i*-th element is 1. If there is no CBDP trail such that* $(\boldsymbol{e}_i, \boldsymbol{u}_{I_v}) \xrightarrow{f} 1$ *, 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 output bit of target cipher, we can use MILP method to identify the involved keys set I by checking whether there is division trial $\{(\boldsymbol{e}_i, \boldsymbol{u}_{I_v})\} \xrightarrow{f} 1$ for $i = 0, 1, \cdots, n-1$. Then, at CRYPTO 2018 [26], Wang *et al.* proposed the degree bounding and term enumeration techniques to further reduce the complexity of recovering superpoly. The degree evaluation of superpoly is based on the following proposition.

Proposition 2. [26] *For a set* $I_x = \{i_0, i_1, \ldots, i_{|I_x|-1}\} \subset \{0, 1, \ldots, n-1\}$*, if there is no CBDP trail such that* $(\boldsymbol{u}_{I_x}, \boldsymbol{u}_{I_v}) \xrightarrow{f} 1$*, 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 keys 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 $\boldsymbol{x}$, 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 The Propagation Properties of BDPT

In this section, we will explore some new propagation properties of BDPT.

3.1 The BDPT Propagation of S-box

In the Sect. 2.3, we have introduced the existing BDPT propagation rules of Copy, And, and Xor. Although any Boolean function can be evaluated by using these three rules, the propagation requires large time and memory complexity when the Boolean function is complex. Here, we propose a generalized method to calculate the BDPT propagation of S-box.

Theorem 1. *For an S-box:* $\mathbb{F}_2^n \to \mathbb{F}_2^m$, *let* $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ *and* $\boldsymbol{y} = (y_0, y_1, \ldots, y_{m-1})$ *denote the input and output. Every* $y_i, i \in \{0, 1, \ldots, m-1\}$ *can be expressed as a boolean function of* $(x_0, x_1, \ldots, x_{n-1})$. *If the input BDPT of S-box is* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}\}}$, *then the output BDPT of S-box can be calculated by* $\mathcal{D}^{1^m}_{\boldsymbol{Reduce0}(\underline{\mathbb{K}}),\boldsymbol{Reduce1}(\underline{\mathbb{K}},\underline{\mathbb{L}})}$, *where*

$$\underline{\mathbb{K}} = \{\boldsymbol{u}' \in \mathbb{F}_2^m | \text{ for any } \boldsymbol{u} \in \mathbb{K}, \text{ if } \boldsymbol{y}^{\boldsymbol{u}'} \text{ contain any term } \boldsymbol{x}^{\boldsymbol{v}} \text{ satisfying } \boldsymbol{v} \succeq \boldsymbol{u}\}.$$

$$\underline{\mathbb{L}} = \{\boldsymbol{u} \in \mathbb{F}_2^m | \boldsymbol{y}^{\boldsymbol{u}} \text{contains the term } \boldsymbol{x}^{\boldsymbol{\ell}}\}.$$

Proof. Let $\mathbb{K}'$ be the set of output BDPT that has no redundant vectors. According to the CBDP rules 5 in Sect. 2.3, we know that $\mathbb{K}' = \boldsymbol{Reduce0}(\underline{\mathbb{K}})$.

Let $\mathbb{L}'$ be the set of output BDPT that has no redundant vectors. For any $\boldsymbol{u} \in \mathbb{L}'$, we have $\bigoplus_{\boldsymbol{y} \in \mathbb{Y}} \boldsymbol{y}^{\boldsymbol{u}} = 1$. Since there is only one vector $\boldsymbol{\ell}$ in the input $\mathbb{L}$, the ANF of $\boldsymbol{y}^{\boldsymbol{u}}$ must has the monomial $\boldsymbol{x}^{\boldsymbol{\ell}}$. Thus, we get $\mathbb{L}' \subset \underline{\mathbb{L}}$. Because the function ***Reduce1*** only removes the vectors satisfying $\bigoplus_{\boldsymbol{y} \in \mathbb{Y}} \boldsymbol{y}^{\boldsymbol{u}} = unknown$, we have $\mathbb{L}' \subset \boldsymbol{Reduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}})$.

On the other hand, if $\boldsymbol{y}^{\boldsymbol{u}}$ contains the monomial $\boldsymbol{x}^{\boldsymbol{\ell}}$, we have $\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{y}^{\boldsymbol{u}}$ equals unknown or 1. For the set $\underline{\mathbb{L}}$, the function ***Reduce1*** will remove all the vectors satisfying $\bigoplus_{\boldsymbol{y} \in \mathbb{Y}} \boldsymbol{y}^{\boldsymbol{u}} = unknown$. So all the remaining vectors satisfying $\bigoplus_{\boldsymbol{y} \in \mathbb{Y}} \boldsymbol{y}^{\boldsymbol{u}} = 1$. Then, we get $\boldsymbol{Reduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}}) \subset \mathbb{L}'$.

Altogether, we obtain $\mathbb{L}' = \boldsymbol{Reduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}})$. □

We apply Theorem 1 to the core operation of SIMON family, the obtained BDPT propagation rules are in accordance with that in [21]. Note that Theorem 1 can get the BDPT propagation rules when the input $\mathbb{L}$ has only one vector. If there are more vectors in $\mathbb{L}$, the paper [21] has showed an example on how to get its BDPT propagation rules. Let $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}_0,\boldsymbol{\ell}_1,\ldots,\boldsymbol{\ell}_{r-1}\}}$ and $\mathcal{D}^{1^m}_{\mathbb{K}',\mathbb{L}'}$ be the input and output BDPT of S-box, respectively. According to Theorem 1, we can get the output BDPT $\mathcal{D}^{1^m}_{\mathbb{K}',\mathbb{L}'_i}$ from the corresponding input BDPT $\mathcal{D}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}_i\}}$, where $i = 0, 1, \ldots, r-1$. Then,

$$\mathbb{L}' = \{\boldsymbol{\ell} | \boldsymbol{\ell} \text{ appears odd times in sets } \mathbb{L}'_0, \mathbb{L}'_1, \ldots, \mathbb{L}'_{r-1}\}.$$

And we also give an example in Sect. 5.1 to help readers understand the propagation of BDPT.

3.2 Pruning Techniques of BDPT

The previous works often divide ciphers into r rounds, and investigate the CBDP or BDPT of round functions. Round functions often have too many operations which will generate many redundant intermediate vectors of division property. When the round number or block size grows, it will make propagation impossible just because of complexity. In order to solve this problem, we divide the ciphers

into small parts. And after getting the BDPT propagation of a part, we will use the pruning techniques to remove the redundant vectors. Then, the remaining vectors in BDPT can continue to propagate efficiently.

Let Q_i be the i-th round function of an r-round cipher $E = Q_r \circ Q_{r-1} \circ \cdots \circ Q_1$, then we divide Q_i into l_i parts $Q_i = Q_{i,l_i-1} \circ Q_{i,l_i-2} \circ \cdots \circ Q_{i,0}$. Let $E_{i,j} = (Q_{i,j-1} \circ Q_{i,j-2} \circ \cdots \circ Q_{i,0}) \circ (Q_{i-1} \circ Q_{i-2} \circ \cdots \circ Q_1)$ and $\overline{E_{i,j}} = (Q_r \circ Q_{r-1} \circ \cdots \circ Q_{i+1})\,(Q_{i,l_i-1} \circ Q_{i,l_i-2} \circ \cdots \circ Q_{i,j})$, then $E = \overline{E_{i,j}} \circ E_{i,j}$, where $1 \le i \le r, 0 \le j \le l_i - 1$ and $E_{1,0}$ is identity function.

Theorem 2. (Prune $\mathbb{K}$) *For r-round cipher $E = Q_r \circ Q_{r-1} \circ \cdots \circ Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$. For any vector $\boldsymbol{k} \in \mathbb{K}_{i,j}$, if there is no CBDP trail such that $\boldsymbol{k} \xrightarrow{\overline{E_{i,j}}} \boldsymbol{e}_m$, the BDPT propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ is equivalent to that of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\to\boldsymbol{k},\mathbb{L}_{i,j}}$ on whether $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ and $\boldsymbol{e}_m \in \mathbb{L}_{r+1,0}$ or not.*

Proof. In Sect. 2.3, we know that for public function, the BDPT propagation of $\mathbb{K}_{i,j}$ and $\mathbb{L}_{i,j}$ is independent. Only when the secret round key is Xored, some vectors of $\mathbb{L}_{i,j}$ will affect $\mathbb{K}_{i,j}$, but they only adds some vectors into $\mathbb{K}_{i,j}$. Because every vector $\boldsymbol{k} \in \mathbb{K}_{i,j}$ is propagated independently based on CBDP, if there is no CBDP trail such that $\boldsymbol{k} \xrightarrow{\overline{E_{i,j}}} \boldsymbol{e}_m$, then removing it from $\mathbb{K}_{i,j}$ doesn't have any impact on whether $\mathbb{K}_{r+1,0}$ includes $\boldsymbol{e}_m$ or not. That means $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ has the same result with $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\to\boldsymbol{k},\mathbb{L}_{i,j}}$ on whether $\mathbb{K}_{r+1,0}$ includes $\boldsymbol{e}_m$ or not.

Because all the vectors of $\mathbb{L}_{r+1,0}$ are generated from $\mathbb{L}_{i,j}$, that is, removing $\boldsymbol{k}$ from $\mathbb{K}_{i,j}$ has no impact on the generation of $\boldsymbol{e}_m \in \mathbb{L}_{r+1,0}$. On the other hand, we have got that removing $\boldsymbol{k}$ from $\mathbb{K}_{i,j}$ doesn't have any impact on whether $\mathbb{K}_{r+1,0}$ includes $\boldsymbol{e}_m$ or not. So it has no impact on the reduction of $\boldsymbol{e}_m \in \mathbb{L}_{r+1,0}$. That means $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ has the same result with $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\to\boldsymbol{k},\mathbb{L}_{i,j}}$ on whether $\mathbb{L}_{r+1,0}$ includes $\boldsymbol{e}_m$ or not. □

Theorem 3. (Prune $\mathbb{L}$) *For r-round cipher $E = Q_r \circ Q_{r-1} \circ \cdots \circ Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$. For any vector $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$, if there is no CBDP trail such that $\boldsymbol{\ell} \xrightarrow{\overline{E_{i,j}}} \boldsymbol{e}_m$, the BDPT propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ is equivalent to that of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$ on whether $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ and $\boldsymbol{e}_m \in \mathbb{L}_{r+1,0}$ or not.*

Proof. For any vector $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$, if there is no CBDP trail such that $\boldsymbol{\ell} \xrightarrow{\overline{E_{i,j}}} \boldsymbol{e}_m$, according to Theorem 2, the BDPT propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ is equivalent to that of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}}$ on whether $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ and $\boldsymbol{e}_m \in \mathbb{L}_{r+1,0}$ or not.

Because $\mathbb{K}_{i,j} \leftarrow \boldsymbol{\ell}$, the vector $\boldsymbol{\ell}$ can be removed from $\mathbb{L}_{i,j}$ according to the definition of BDPT. So the BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}}$ is completely equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$.

According to Theorem 2 again, the BDPT propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$ is equivalent to that of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$ on whether $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ and $\boldsymbol{e}_m \in \mathbb{L}_{r+1,0}$ or not. □

The propagation of CBDP can be efficiently solved by MILP model. Therefore, the meaning of Theorem 2 and Theorem 3 is that we can use CBDP method to reduce the BDPT sets $\mathbb{K}_{i,j}$ and $\mathbb{L}_{i,j}$.

3.3 Fast Propagation

Inspired by the notion of "lazy propagation", we propose a notion called "fast propagation" to show the balanced information of output bits.

Definition 5. (Fast Propagation). *For r-round cipher $E = Q_r \circ Q_{r-1} \circ \cdots \circ Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$. Under fast propagation, we translate the BDPT into CBDP $\mathcal{D}^{1^n}_{\overline{\mathbb{K}}_{i,j}}$, where $\overline{\mathbb{K}}_{i,j} = \mathbb{K}_{i,j} \cup \mathbb{L}_{i,j}$. The output CBDP of $\overline{E_{i,j}}$ is computed from $\mathcal{D}^{1^n}_{\overline{\mathbb{K}}_{i,j}}$.*

The "fast propagation" removes all vectors from $\mathbb{L}_{i,j}$, and get the union set $\mathbb{K}_{i,j} \cup \mathbb{L}_{i,j}$. By its nature, "fast propagation" translate BDPT into CBDP. We can use the MILP method to solve the CBDP propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\cup\mathbb{L}_{i,j}}$,. Let us consider the meaning of "fast propagation". Assuming the input set of $\overline{E_{i,j}}$ has BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$, according to the definition of BDPT and CBDP, this set must also has CBDP $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\cup\mathbb{L}_{i,j}}$,. If for any $\boldsymbol{k} \in \mathbb{K}_{i,j} \cup \mathbb{L}_{i,j}$, there is no CBDP trial such that $\boldsymbol{k} \xrightarrow{\overline{E_{i,j}}} \boldsymbol{e}_m$, then the m-th output bit of $\overline{E_{i,j}}$ is balanced.

4 The MILP-aided Method for BDPT

Based on the work of [27], we first simplify the MILP algorithm of searching integral distinguishers based on CBDP to improve efficiency. Then, we show three stopping rules and propose an algorithm to search integral distinguishers based on BDPT.

4.1 Simplify the MILP Method of CBDP

Using the method in the paper [27], we can get a linear inequality set which describes the r-round CBDP division trails with the given initial CBDP $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$. The former CBDP method will return a set of balanced bits. Because only one bit's balanced information is needed, our MILP model has no objective function which is added into the constrains. We can use the solver Gurobi [10] to determine whether the MILP model has feasible solutions or not. If it has feasible solutions, it shows that the m-th bit of the output is unknown. Otherwise, the m-th bit is balanced. The detail information is shown in Algorithm 1.

4.2 Stopping Rules

Based on "lazy propagation" and "fast propagation", in this subsection, we propose three stopping rules in searching integral distinguishers based on BDPT.

Algorithm 1 $SCBDP(E, \boldsymbol{k}, m)$

Input: The cipher E, the initial CBDP vector $\boldsymbol{k}$, and the number m
Output: Whether the m-th bit of the output is balanced or not based on CBDP
1 **begin**
2 $\mathcal{L}$ is a linear inequality set which describe the CBDP division trails such that $\boldsymbol{k} \xrightarrow{E} \boldsymbol{e}_m$
3 **if** $\mathcal{L}$ has feasible solutions **do**
4 **return** unknown
5 **else**
6 **return** 0
7 **end**

Stopping Rule 1. *For an r-round cipher $E = Q_r \circ Q_{r-1} \circ \cdots \circ Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$. For any vector $\boldsymbol{k} \in \mathbb{K}_{i,j}$, if there is CBDP trail such that $\boldsymbol{k} \xrightarrow{\overline{E_{i,j}}} \boldsymbol{e}_m$, according to "lazy propagation", we stop the process and obtain that the m-th output bit of E is unknown.*

After Stopping Rule 1, if the searching procedure doesn't stop, all the vectors in $\mathbb{K}_{i,j}$ will be removed according to the pruning technique in Theorem 2. Then, we consider the following Stopping Rule 2.

Stopping Rule 2. *After removing the redundant vectors in the set $\mathbb{L}_{i,j}$ by the pruning technique in Theorem 3, if there is still vector $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$, we cannot stop the procedure and $\boldsymbol{\ell}$ should be propagated to next part based on BDPT. If there is no vector in $\mathbb{L}_{i,j}$, according to "fast propagation", we can get that the m-th output bit of E is balanced.*

Different from Stopping Rule 1 which shows the m-th bit is unknown, Stopping Rule 2 can show the m-th bit is balanced based on BDPT. If the process doesn't stop even we get the output BDPT of E, Stopping Rule 3 can explain this situation.

Stopping Rule 3. *If $\mathbb{K}_{r+1,0} = \emptyset$ and $\mathbb{L}_{r+1,0} = \{\boldsymbol{e}_m\}$, then we find an integral distinguisher whose sum of the m-th output bit is 1.*

4.3 The MILP-aided Method of Searching Integral Distinguishers Based on BDPT

The algorithm of searching integral distinguishers often has a given initial BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$. For an indices set $I = \{i_0, i_1, \ldots, i_{|I|-1}\} \subset \{0, 1, \ldots, n-1\}$, attackers prepare $2^{|I|}$ chosen plaintexts where variables indexed by I are taking all possible combinations of values and the other variables are set to constants. The CBDP of such chosen plaintexts is $\mathcal{D}^{1^n}_{\{\boldsymbol{u}_I\}}$. Then, the BDPT of such chosen plaintexts is $\mathcal{D}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$, where $\mathbb{K}_{1,0} = \{\boldsymbol{u}' \in \mathbb{F}_2^n | \boldsymbol{u}' \succ \boldsymbol{u}_I\}$ and $\mathbb{L}_{1,0} = \{\boldsymbol{u}_I\}$. We illustrate the whole framework in Algorithm 2.

Algorithm 2 $BDPT(E, \mathbb{L}_{1,0}, \mathbb{K}_{1,0}, m)$

Input: The cipher E, the input BDPT $\mathcal{D}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$, and the number m
Output: The balanced information of the m-th output bit based on BDPT

```
1  begin
2      for (i = 1; i ≤ r; i++) do
3          for (j = 0; j ≤ l_i − 1; j++) do
4              for k in K_{i,j}
5                  if SCBDP(E_{i,j}, k, m) is unknown
6                      return unknown
7                  else
8                      K_{i,j} → k
9              end
10             L'_{i,j} = ∅
11             for ℓ in L_{i,j} do
12                 if SCBDP (E_{i,j}, ℓ, m) is unknown
13                     L'_{i,j} = L'_{i,j} ∪ ℓ
14                 end
15             end
16             if L'_{i,j} = ∅
17                 return 0
18             end
19             D_{K_{i+⌊(j+1)/l_i⌋,(j+1) mod l_i}, L_{i+⌊(j+1)/l_i⌋,(j+1) mod l_i}} = BDPTP(Q_{i,j}, D_{∅,L'_{i,j}})
20         end
21     end
22     return 1
23 end
```

We explain **Algorithm 2** line by line:
Line 2-3 The cipher E is divided into small parts.
Line 4-9 For every $\boldsymbol{k} \in \mathbb{K}_{i,j}$, if $SCBDP\left(\overline{E_{i,j}}, \boldsymbol{k}, m\right)$ is unknown (Algorithm 1), according to Stopping Rule 1, we know that the m-th output bit is unknown based on BDPT. Otherwise, we remove it from $\mathbb{K}_{i,j}$ according to the pruning technique in Theorem 2.
Line 10 Initialize $\mathbb{L}'_{i,j}$ to be an empty set.
Line 11-15 For any vector $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$, if $SCBDP\left(\overline{E_{i,j}}, \boldsymbol{\ell}, m\right)$ can generate the unit vector $\boldsymbol{e}_m$, we store all these vectors in $\mathbb{L}'_{i,j}$.
Line 16-18 If the set $\mathbb{L}'_{i,j}$ is empty set, it satisfies Stopping Rule 2, that is, the m-th output bit is balanced.
Line 19 If we don't get the balanced information of the m-th bit, we should use the propagation rules of BDPT to get the input BDPT of the next part.
Line 22 It triggers Stopping Rules 3, and the sum of the m-th output bit is 1.

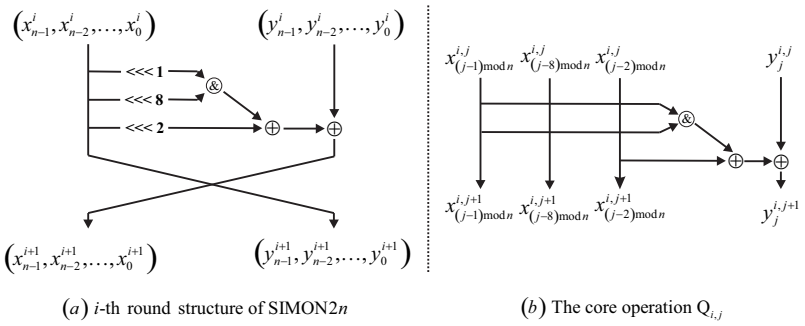
The principle of dividing the round function Q_i is that the vectors of BDPT don't expand too much. Only in this way can we run the searching algorithm efficiently. Algorithm 2 can show the balanced information of any output bit. Therefore, we can search the integral distinguishers of cipher in parallel.

5 Applications to Block Ciphers

In this section, we apply our algorithm to SIMON, SIMECK, PRESENT, RECTANGLE, and LBlock. All the experiments are conducted on the platform: Intel Core i5-4590 CPU @3,3GHz, 8.00G RAM. And the optimizer we used to solve MILP models is Gurobi 8.1.0 [10]. For the integral distinguishers, what needs to be explained is that "a" denotes active bit, "c" denotes constant bit, "?" denotes the balanced information is unknown, and "b" denotes the balanced bit.

5.1 Applications to SIMON and SIMECK

SIMON is a lightweight block cipher family [2] based on Feistel structure which only involves bit-wise And, Xor, and Circular shift operations. Let SIMON2n be the SIMON cipher with $2n$-bit block length, where $n \in \{16, 24, 32, 48, 64\}$. And the left part of Fig.1 shows the round structure of SIMON2n. The core operation of round function is represented by the right part of Fig. 1.



(a) i-th round structure of SIMON2n (b) The core operation $Q_{i,j}$

Fig. 1. The structure of SIMON2n

When we apply Algorithm 2 to SIMON2n, we divide one-round SIMON2n into $n+1$ parts $Q_i = Q_{i,n} \circ Q_{i,n-1} \circ \cdots \circ Q_{i,0}$. And the input of $Q_{i,j}$ is denoted as $(\boldsymbol{x}^{i,j}, \boldsymbol{y}^{i,j}) = \left(x_{n-1}^{i,j}, \ldots, x_0^{i,j}, y_{n-1}^{i,j}, \ldots, y_0^{i,j}\right)$. When $0 \le j \le n-1$, we have

$$Q_{i,j}\left(\boldsymbol{x}^{i,j}, \boldsymbol{y}^{i,j}\right) = \left(\boldsymbol{x}^{i,j}, y_{n-1}^{i,j}, \ldots, y_{j+1}^{i,j}, Y_j^{i,j}, y_{j-1}^{i,j}, \ldots, y_0^{i,j}\right),$$

where $Y_j^{i,j} = \left(x_{(j-1)\bmod n}^{i,j} \& x_{(j-8)\bmod n}^{i,j}\right) \bigoplus x_{(j-2)\bmod n}^{i,j}$.

Moreover, $Q_{i,n}\left(\boldsymbol{x}^{i,n}, \boldsymbol{y}^{i,n}\right) = \left(\boldsymbol{y}^{i,n} \oplus \boldsymbol{k}^i, \boldsymbol{x}^{i,n}\right)$, where $\boldsymbol{k}^i$ is the i-th round key of SIMON2n.

For $Q_{i,j}, 0 \le j \le n-1$, when we consider the BDPT propagation rules of the function $BDPTP\left(Q_{i,j}, \mathcal{D}_{\emptyset,\mathbb{L}'_{i,j}}\right)$, $(2n-4)$ bits remain unchanged. Thus, only 4-bit $\left(x_{(j-1)\bmod n}^{i,j}, x_{(j-2)\bmod n}^{i,j}, x_{(j-8)\bmod n}^{i,j}, y_{(j)\bmod n}^{i,j}\right)$ of the BDPT vectors will be

changed. We can view it as 4-bit S-box and use Theorem 1 to get its accurate BDPT propagation rules which are in accordance with that in the paper [21]. We show it in Appendix Table 7.

When we use Algorithm 2 to search the integral distinguishers of SIMON2n based on BDPT, we should call Algorithm 1 to build the MILP model based on CBDP. The paper [27] has showed us how to model CBDP division trails of 1-round SIMON2n. We introduce it as follows.

1-round Description of SIMON2n**.** Denote 1-round CBDP trail of SIMON2n by $\left(a_{n-1}^i,\ldots,a_0^i,b_{n-1}^i,\ldots,b_0^i\right) \rightarrow \left(a_{n-1}^{i+1},\ldots,a_0^{i+1},b_{n-1}^{i+1},\ldots,b_0^{i+1}\right)$. In order to get a linear description of all CBDP trails of 1-round SIMON2n, we introduce four vectors of auxiliary variables which are $\left(u_{n-1}^i,\ldots,u_0^i\right)$, $\left(v_{n-1}^i,\ldots,v_0^i\right)$, $\left(w_{n-1}^i,\ldots,w_0^i\right)$ and $\left(t_{n-1}^i,\ldots,t_0^i\right)$. We denote $\left(u_{n-1}^i,\ldots,u_0^i\right)$ the input CBDP of the left circular shift by 1 bit. Similarly, denote $\left(v_{n-1}^i,\ldots,v_0^i\right)$ and $\left(w_{n-1}^i,\ldots,w_0^i\right)$ the input CBDP of the left circular shift by 8 bits and 2 bits, respectively. Let $\left(t_{n-1}^i,\ldots,t_0^i\right)$ denote the output CBDP of bit-wise And operation. The following inequalities are sufficient to model the Copy operation used in SIMON2n:

$$\mathcal{L}_1 : a_j^i - u_j^i - v_j^i - w_j^i - b_j^{i+1} = 0 \text{ for } j \in \{0,1,\ldots,n-1\}.$$

Then, the bit-wise And operation used in SIMON2n can be modeled by:

$$\mathcal{L}_2 = \begin{cases} t_j^i - u_{(j-1)\bmod n}^i \geq 0, & \text{for } j \in \{0,1,\ldots,n-1\}, \\ t_j^i - v_{(j-8)\bmod n}^i \geq 0, & \text{for } j \in \{0,1,\ldots,n-1\}, \\ t_j^i - u_{(j-1)\bmod n}^i - v_{(j-8)\bmod n}^i \leq 0, & \text{for } j \in \{0,1,\ldots,n-1\}. \end{cases}$$

At last, the Xor operation in SIMON2n can be modeled by:

$$\mathcal{L}_3 : a_j^{i+1} - b_j^i - t_j^i - w_{(j-2)\bmod n}^i = 1 \text{ for } j \in \{0,1,\ldots,n-1\}.$$

So far, we get a description $\{\mathcal{L}_1,\mathcal{L}_2,\mathcal{L}_3\}$ of 1-round CBDP trails.

How to Describe the CBDP Propagation of Partial Round. For $\overline{E_{i,j}}$, the first round maybe a partial round $Q_{i,l_i-1} \circ Q_{i,l_i-2} \circ \cdots \circ Q_{i,j}$. when considering the CBDP propagation of $Q_{i,j}$, if add constrain $b_j^{i+1,j} = b_j^{i,j}$, the output vector is the same with the input vector. Namely, $Q_{i,j}$ is transformed into identity function.

For 1-round SIMON2n, by adding the following constrains

$$\mathcal{L}_4 : a_j^{i+1} - b_j^i = 0 \text{ for } j \in \{0,1,\ldots,j-1\},$$

we obtain a description $\{\mathcal{L}_1,\mathcal{L}_2,\mathcal{L}_3,\mathcal{L}_4\}$ of partial round $Q_{i,l_i-1} \circ Q_{i,l_i-2} \circ \cdots \circ Q_{i,j}$. Then, by repeating the constrains of 1-round $(r-i)$ times, we can get a linear inequality system $\mathcal{L}$ for $\overline{E_{i,j}}$.

How to Obtain the Output BDPT of $Q_{i,j}$. After the pruning techniques and stopping rules, if Algorithm 2 doesn't stop, we know that $\mathbb{K}_{i,j} = \emptyset$ and $\mathbb{L}_{i,j} \neq \emptyset$. In order to help readers understand our algorithm, we show an example of the propagation of BDPT.

For SIMON32, if the input BDPT of $Q_{1,15}$ is $\mathcal{D}_{\mathbb{K}_{1,15}=\emptyset,\mathbb{L}_{1,15}=\{\boldsymbol{\ell}_1,\boldsymbol{\ell}_2\}}$, where $\boldsymbol{\ell}_1 = (1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{0},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$, $\boldsymbol{\ell}_2 = (1,\mathbf{0},\mathbf{0},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$. The 4 bits of $\boldsymbol{\ell}_1$ that may be updated by $Q_{1,15}$ is $(0,1,1,0)$. Then, according to the BDPT propagation rules of core operation in Table 7. The output vector set is $\mathbb{L}' = \{[0,1,1,0],[0,1,0,1],[0,1,1,1]\}$. So $\boldsymbol{\ell}_1$ generates three vectors as:

$$(1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{0},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$$
$$(1,\mathbf{0},\mathbf{0},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$$
$$(1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$$

In the same way, we can obtain that $\boldsymbol{\ell}_2$ generates only one vector as

$$(1,\mathbf{0},\mathbf{0},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)\,.$$

According to **BDPT Rule 3**, the vector $(1,\mathbf{0},\mathbf{0},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$ should be canceled because it is propagated from $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ twice. The output BDPT of $Q_{1,15}$ is $\mathcal{D}_{\mathbb{K}_{1,16}=\emptyset,\mathbb{L}_{1,16}=\{\boldsymbol{\ell}_3,\boldsymbol{\ell}_4\}}$, where

$$\boldsymbol{\ell}_3 = (1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{0},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)\,,$$
$$\boldsymbol{\ell}_4 = (1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)\,.$$

Then, $Q_{1,16}$ has round keys Xored operation. So a new vector is generated from $\boldsymbol{\ell}_3$ and inserted into $\mathbb{K}_{1,16}$ according to the **BDPT Rule 4**. Moreover, a vector in $\mathbb{L}_{1,16}$ becomes redundant because of the new vector of $\mathbb{K}_{1,16}$. After the swapping, the output BDPT of $Q_{1,16}$ is $\mathcal{D}_{\mathbb{K}_{2,0}=\{\boldsymbol{k}\},\mathbb{L}_{2,0}=\{\boldsymbol{\ell}_5\}}$, where

$$\boldsymbol{k} = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1),$$
$$\boldsymbol{\ell}_5 = (0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1).$$

The High Efficiency of Our Algorithm. For 14-round SIMON32, we prepare chosen plaintexts such that the leftmost bit is constant and the others are active. Then, the BDPT of chosen plaintexts is $\mathcal{D}_{\mathbb{K}=\{(1,1,1,\ldots,1)\},\mathbb{L}=\{(0,1,1,\ldots,1)\}}$. Table 3 shows the sizes of $|\mathbb{K}|$ and $|\mathbb{L}|$ in every round. The sizes in the paper [21] are obtained after removing redundant vectors according to the definition of BDPT, while the sizes in this paper are obtained after the pruning techniques. From Table 3, we find that $|\mathbb{L}|$ of the 5-th round in this paper becomes 0, it triggers Stopping Rule 2, and we obtain that the rightmost bit is balanced. Our pruning techniques can reduce the size of BDPT greatly.

Integral Distinguishers. SIMECK is a family of lightweight block cipher proposed at CHES 2015 [29], and its round function is very similar to that of SIMON except the rotation constants. We use Algorithm 2 to search the integral distinguishers of SIMON and SIMECK family based on BDPT. For SIMON32,

Table 3. Sizes of $\mathcal{D}_{\mathbb{K},\mathbb{L}}$ in obtaining balanced information of the rightmost output bit

Reference	BDPT	Size in every round															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
[21]	$\mathbb{L}$	1	1	5	19	138	2236	89878	4485379	47149981	2453101	20360	168	8	0	0	0
	$\mathbb{K}$	1	1	1	6	43	722	23321	996837	9849735	2524718	130724	7483	852	181	32	32
This paper	$\mathbb{L}$	1	1	1	2	2	0	0	0	0	0	0	0	0	0	0	0
	$\mathbb{K}$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

our MILP algorithm finds the 14-round integral distinguisher that found in [21] by going through all the BDPT division trails. For 17-round SIMON64, we find an integral distinguisher with 23 balanced bits which has one more bit than the previous longest integral distinguisher. For SIMON48/96/128 and SIMECK32/48/64, the distinguishers we find are in accordance with the previous longest distinguishers that found in [27]. The detailed integral distinguishers of SIMON32 and SIMON64 are listed in Table 4. And all the integral distinguishers in Table 4 can be extended one more round by the technique in [25].

Table 4. Integral distinguishers of SIMON32 and SIMON64

Cipher	Distinguisher
14-SIMON32	In: (caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa) Out: (????????????????, ?b??????b??????b)
17-SIMON64	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????????????,bbbbbbbbbbb??b??b?????bbbbbbbbbb)

5.2 Applications to PRESENT and RECTANGLE

PRESENT [3] has an SPN structure and uses 80- and 128-bit keys with 64-bit blocks through 31 rounds. In order to improve the hardware efficiency, it use a fully wired diffusion layer. Fig. 2 illustrates one-round structure of PRESENT.

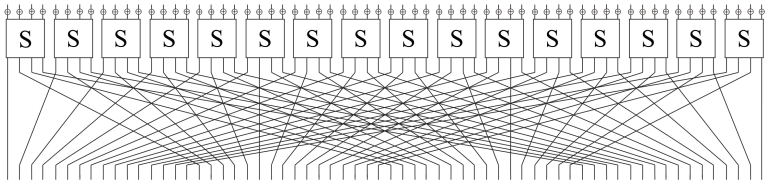


Fig. 2. One-round SPN structure of PRESENT

We divide one-round PRESENT into 17 parts $Q_i = Q_{i,16} \circ \cdots \circ Q_{i,0}$. When $0 \leq j \leq 15$, we have $Q_{i,j}\left(x_0^{i,j}, \ldots, x_{63}^{i,j}\right) = \left(x_0^{i,j}, \ldots, S\left(x_{4j}^{i,j}, \ldots, x_{4j+3}^{i,j}\right), \ldots, x_{63}^{i,j}\right)$, where $S\left(x_{4j}^{i,j}, \ldots, x_{4j+3}^{i,j}\right)$ is the S-box of PRESENT.

Moreover, $Q_{i,16}\left(x_0^{i,16}, \ldots, x_{63}^{i,16}\right) = P\left(x_0^{i,16}, x_1^{i,16}, \ldots, x_{63}^{i,16}\right) \oplus \boldsymbol{k}^i$, where P is the linear permutation of PRESENT and $\boldsymbol{k}^i$ is the i-th round key.

RECTANGLE [31] is very like PRESENT. We apply Algorithm 2 to PRESENT and RECTANGLE, and the results are listed in Table 5.

Table 5. Integral distinguishers of PRESENT and RECTANGLE

Cipher	Distinguisher
9-PRESENT	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,cccc) Out: (????????????????????????????????,??????????????????b???b???b???b)
9-PRESENT	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaac) Out: (???b???b???bbbbb???b???b???bbbbb, ???b???b???bbbbb???b???b???bbbbb)
9-RECTANGLE	In: (caaaaaaaaaaaaaaa,caaaaaaaaaaaaaaa,caaaaaaaaaaaaaaa,caaaaaaaaaaaaaaa) Out: (bbbbbbbbbbbbbbbb,bbbb??bb???bbbbb,????????????????,????????????????)

5.3 Applications to LBlock

LBlock is a lightweitht block cipher proposed by Wu and Zhang [24]. The block size is 64 bits and the key size is 80 bits. It employs a variant Feistel structure and consists of 32 rounds. One-round structure of LBlock is given in Fig. 3.

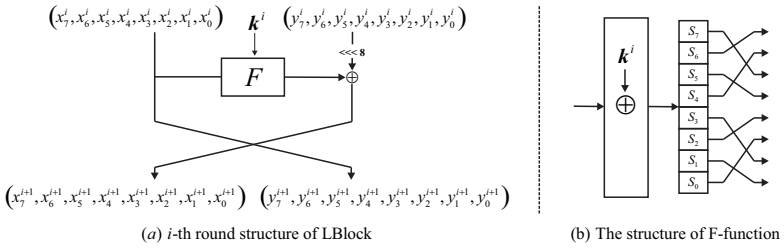


Fig. 3. Round structure of LBlock

We divide one-round LBlock into 9 parts $Q_i = Q_{i,8} \circ \cdots \circ Q_{i,0}$. And the input of $Q_{i,j}$ is denoted as $\left(\boldsymbol{x}^{i,j}, \boldsymbol{y}^{i,j}\right) = \left(\boldsymbol{x}_7^{i,j}, \ldots, \boldsymbol{x}_0^{i,j}, \boldsymbol{y}_7^{i,j}, \ldots, \boldsymbol{y}_0^{i,j}\right)$. When $0 \leq j \leq 7$, we have $Q_{i,j}\left(\boldsymbol{x}^{i,j}, \boldsymbol{y}^{i,j}\right) = \left(\boldsymbol{x}^{i,j}, \boldsymbol{y}_7^{i,j}, \ldots, \boldsymbol{y}_{P(j)+1}^{i,j}, Y_{P(j)}^{i,j}, \boldsymbol{y}_{P(j)-1}^{i,j}, \ldots, \boldsymbol{y}_0^{i,j}\right)$, where $Y_{P(j)}^{i,j} = S_j\left(\boldsymbol{x}_j^{i,j} \bigoplus k_{i,j}\right) \bigoplus \boldsymbol{y}_{(P(j)-2) \bmod 8}^{i,j}$, S_j is the j-th S-box of LBlock, and $P(x)$ is the nibble diffusion function. Moreover, $Q_{i,8}\left(\boldsymbol{x}^{i,8}, \boldsymbol{y}^{i,8}\right) = \left(\boldsymbol{y}^{i,8}, \boldsymbol{x}^{i,8}\right)$.

Using Algorithm 2, we find a 17-round integral distinguisher of LBlock which is in accordance with the previous longest integral distinguisher [8], and a better 16-round integral distinguisher with less active bits. The detail forms of the integral distinguishers are shown in Table 6.

Table 6. Integral distinguishers of LBlock

Cipher	Distinguisher
17-LBlock	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (??????????????????????????????????,??bb????????????????????????????bb)
16-LBlock	In: (aaccaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (??????????????????????????????????,??bbbbbbbbbb?b?bb?b?bbbb????????)

6 Using BDPT to Recover the Superpoly in Cube Attack

In this section, we analyze the ANF coefficients of non-blackbox polynomial and superpoly in cube attack. Then, we show an MILP-aided method based on BDPT to recover the ANF coefficients of superpoly.

6.1 Analyze the ANF Coefficients of Polynomial

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 $I_v \subset \{0, 1, \cdots, m-1\}$ are chosen as cube variables, the public variables indexed by $J_v \subset \{0, 1, \cdots, m-1\} - I_v$ are set to 1, and the remaining public variables $K_v = \{0, 1, \cdots, m-1\} - I_v - J_v$ are set to 0. Then, the ANF of $f_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ can be represented as follows

$$f_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v}) = \bigoplus_{\boldsymbol{u}_x \in \mathbb{F}_2^n, \boldsymbol{u}_v \preceq \boldsymbol{u}_I} a_{(\boldsymbol{u}_x,\boldsymbol{u}_v)}^{f_{I_v,J_v,K_v}} \cdot (\boldsymbol{x}, \boldsymbol{v})^{(\boldsymbol{u}_x,\boldsymbol{u}_v)}.$$

where $a_{(\boldsymbol{u}_x,\boldsymbol{u}_v)}^{f_{I_v,J_v,K_v}}$ is the ANF coefficient of term $(\boldsymbol{x}, \boldsymbol{v})^{(\boldsymbol{u}_x,\boldsymbol{u}_v)}$ in $f_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$.

For polynomial $f_{I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ and an index subset $I_x \subset \{0, 1, \cdots, n-1\}$, if fixing all the secret variables $\{x_k | k \in \{0, 1, \cdots, n-1\} - I_x\}$ to 0, we can get a new polynomial denoted as $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$.

Definition 6. (Similar Polynomial). *For subsets of indices $I'_x \subset I_x$, the polynomial $f_{I'_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ is called the similar polynomial of $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$.*

Lemma 2. *If $f_{I'_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$is the similar polynomial of $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$, then the value of ANF coefficient $a^{f_{I'_x,I_v,J_v,K_v}}_{\left(\boldsymbol{u}_{I'_x},\boldsymbol{u}_{I_v}\right)}$ in $f_{I'_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$ is equal to the value of ANF coefficients $a^{f_{I_x,I_v,J_v,K_v}}_{\left(\boldsymbol{u}_{I'_x},\boldsymbol{u}_{I_v}\right)}$ in $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x}, \boldsymbol{v})$.*

Proof. For $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$, if all the variables of $\{x_i|i \in I_x - I'_x\}$ are assigned 0, it becomes the function $f_{I'_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$. Compared with the ANF of $f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$, the ANF of $f_{I'_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ only misses terms that contain any variables of $\{x_i|i \in I_x - I'_x\}$. Moreover, $\boldsymbol{x}^{\boldsymbol{u}_{I'_x}}$ doesn't contain any variables of $\{x_i|i \in I_x - I'_x\}$, so $a^{f_{I'_x,I_v,J_v,K_v}}_{\left(\boldsymbol{u}_{I'_x},\boldsymbol{u}_{I_v}\right)} = a^{f_{I_x,I_v,J_v,K_v}}_{\left(\boldsymbol{u}_{I'_x},\boldsymbol{u}_{I_v}\right)}$.

6.2 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.

Let C_{I_v,J_v,K_v} be a cube set defined as Eq. (1) in Sect. 2.5. For polynomial $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$, where $\boldsymbol{x} \in \mathbb{F}_2^n$ and $\boldsymbol{v} \in \mathbb{F}_2^m$, it can be unique represented as

$$f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v}) = \boldsymbol{v}^{\boldsymbol{u}_{I_v}} \cdot p_{I_v,J_v,K_v}(\boldsymbol{x}) \oplus q_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v}). \tag{3}$$

where $p_{I_v,J_v,K_v}(\boldsymbol{x})$ does not contain any variable in $\{v_i|i \in I_v\}$, and each term of $q_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ is not divisible by $\boldsymbol{v}^{\boldsymbol{u}_{I_v}}$. Then, $p_{I_v,J_v,K_v}(\boldsymbol{x})$ is called the superpoly of C_{I_v,J_v,K_v} in $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$.

Definition 7. *Let C_{I_x,I_v,J_v,K_v} be the set of $(\boldsymbol{x},\boldsymbol{v})$ satisfying secret variables $\{x_i|i \in I_x\}$ are taking all possible combinations of values, secret variables $\{x_i|i \in \{0,1,\ldots,n-1\} - I_x\}$ are set to constant 0, public variables $\{v_i|i \in I_v\}$ are taking all possible combinations of values, public variables $\{v_j|j \in J_v\}$ are set to constant 1, and public variables $\{v_k|k \in K_v\}$ are set to constant 0.*

Here, we propose a method to calculate the ANF coefficient of superpoly.

Proposition 3. *For any index subset $I_x \subset \{0,1,\ldots,n-1\}$, the ANF coefficient of term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in the superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be calculated as*

$$a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}} = \bigoplus_{(\boldsymbol{x},\boldsymbol{v}) \in C_{I_x,I_v,J_v,K_v}} f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v}).$$

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

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

So, the ANF coefficient of $(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}$ 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 $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ can be unique represented as Eq. (3) and 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. (3), 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,

$$a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}} = a^{f_{I_v,J_v,K_v}}_{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}. \tag{4}$$

From the Definition 6, we know that f_{I_x,I_v,J_v,K_v} is the similar polynomial of f_{I_v,J_v,K_v}. And according to Lemma 2, we obtain that

$$a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}} = a^{f_{I_v,J_v,K_v}}_{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})} = a^{f_{I_x,I_v,J_v,K_v}}_{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})}. \tag{5}$$

Then, we have

$$\begin{aligned}
&\bigoplus_{(\boldsymbol{x},\boldsymbol{v})\in C_{I_x,I_v,J_v,K_v}} f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v}) \\
&\quad= \bigoplus_{(\boldsymbol{x},\boldsymbol{v})\in C_{I_x,I_v,J_v,K_v}} \bigoplus_{\boldsymbol{u}_x\preceq\boldsymbol{u}_{I_x},\boldsymbol{u}_v\preceq\boldsymbol{v}_{I_v}} a^{f_{I_x,I_v,J_v,K_v}}_{(\boldsymbol{u}_x,\boldsymbol{u}_v)}\cdot(\boldsymbol{x},\boldsymbol{v})^{(\boldsymbol{u}_x,\boldsymbol{u}_v)} \\
&\quad= a^{f_{I_x,I_v,J_v,K_v}}_{(\boldsymbol{u}_{I_x},\boldsymbol{u}_{I_v})} = a^{p_{I_v,J_v,K_v}}_{\boldsymbol{u}_{I_x}}.
\end{aligned}$$

□

6.3 The Algorithm to Recover Superpoly

The set C_{I_x,I_v,J_v,K_v} can be viewed as a cube set, according to the definition of BDPT, we know that the BDPT of C_{I_x,I_v,J_v,K_v} is $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$, where $\mathbb{K}=\emptyset$, and $\mathbb{L}=\{(\boldsymbol{u}_{I_x},\boldsymbol{u}_v)\,|\boldsymbol{u}_{I_v}\preceq\boldsymbol{u}_v\preceq\boldsymbol{u}_{I_v}\oplus\boldsymbol{u}_{J_v}\}$. Then, we can use MILP-aided method (Algorithm 2) to research the propagation of $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$. The integral distinguisher got by BDPT recover the ANF coefficient of $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$. For example, if Algorithm 2 $BDPT(f_{I_x,I_v,J_v,K_v},\mathbb{K},\mathbb{L},0)$ return 1, it means that $\bigoplus_{(\boldsymbol{x},\boldsymbol{v})\in C_{I_x,I_v,J_v,K_v}} f_{I_x,I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})=1$. According to Proposition 3, we know that the ANF coefficient of $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ in superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ equals 1. We illustrate the whole framework in Algorithm 3.

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

Definition 8. *For a function $f:\mathbb{F}_2^n\to\mathbb{F}_2^m$, if the ANF of f is definite, we call it public function. Let $E=Q_r\circ Q_{r-1}\circ\cdots\circ Q_1(\boldsymbol{x},\boldsymbol{v})$ be an r-round cipher, where Q_i is the i-th round update function, $\boldsymbol{x}$ denotes the secret variables, and $\boldsymbol{v}$ denotes the public variables. If all the round update functions Q_i, $i\in\{1,2,\cdots,r\}$ are public functions, the cipher E is public-update cipher. Otherwise we call it secret-update cipher.*

Proposition 4. *For a public-update cipher $f_{I_v,J_v,K_v}(\boldsymbol{x},\boldsymbol{v})$ and cube set C_{I_v,J_v,K_v}, the superpoly $p_{I_v,J_v,K_v}(\boldsymbol{x})$ can be fully recovered by the propagation of BDPT.*

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, I_v, J_v, K_v)
2   Initial 𝕂 = ∅, 𝕃 = {(u_{I_x}, u_v) | u_{I_v} ⪯ u_v ⪯ u_{I_v} ⊕ u_{J_v}}
3   if BDPT(f_{I_x,I_v,J_v,K_v}, 𝕂, 𝕃, 0) return unknown
4       return unknown
5   else if BDPT(f_{I_x,I_v,J_v,K_v}, 𝕂, 𝕃, 0) return 1
6       return 1
7   else
8       return 0
9 end procedure
```

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 research the propagation of BDPT $\mathcal{D}^{1^{n+m}}_{\mathbb{K},\mathbb{L}}$, where $\mathbb{K}=\emptyset$ and $\mathbb{L}=\{(\boldsymbol{u}_{I_x},\boldsymbol{u}_v)\,|\boldsymbol{u}_{I_v}\preceq\boldsymbol{u}_v\preceq\boldsymbol{u}_{I_v}\oplus\boldsymbol{u}_{J_v}\}$. 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. The output set $\mathbb{K}'=\emptyset$ and the return value of Algorithm 3 is constant (0 or 1). So the ANF coefficient of arbitrary term $\boldsymbol{x}^{\boldsymbol{u}_{I_x}}$ can be recovered by BDPT. □

According to Sect. 2.6, for 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 bounding degree of superpoly. We denote the involved secret variables indices set as I and the upper bounding 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 $hw(\boldsymbol{u})\le d$.

Analysis of Public-update Cipher. According to Proposition 4, we can query the Algorithm 3 $\sum_{i=0}^{d}\binom{|I|}{i}$ times to recover all the ANF coefficients of superpoly. The complexity is $c\cdot\sum_{i=0}^{d}\binom{|I|}{i}$, where c is the average computational complexity of Algorithm 3. 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.

Analysis of 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 the output BDPT set $\mathbb{K}'=\emptyset$ 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. 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 complexity of recovering superpoly is $\left\{c \cdot \sum_{i=0}^{d} \binom{|I|}{i} + N \cdot 2^{|I_v|}\right\}$.

7 Application to Trivium

In order to verify the correctness and effectiveness of our method, we apply it to Trivium [5] which is a public-update cipher.

7.1 Descriptions of Trivium

Trivium [5] is a bit-oriented stream cipher with 288-bit internal state denoted by $\boldsymbol{s} = (s_0, s_1, \ldots, s_{287})$. To outline our technique more conveniently, we describe Trivium using the following expression. Let $\boldsymbol{x} = (x_0, x_1, \cdots, x_{79})$ denote the secret variables (80-bit Key), and $\boldsymbol{v} = (v_0, v_1, \cdots, v_{207})$ denote the public variables. For public variables, $v_{13}, v_{14}, \cdots, v_{92}$ are the IV variables whose values can be chosen by attackers (80-bit IV), $\{v_{205}, v_{206}, v_{207}\}$ are set to 1, and others are set to 0. 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.

$(s_0, s_1, \ldots, s_{92}) \leftarrow (x_0, \ldots, x_{79}, v_0, \ldots, v_{12})$
$(s_{93}, s_{94}, \ldots, s_{176}) \leftarrow (v_{13}, \ldots, v_{96})$
$(s_{177}, s_{178}, \ldots, s_{287}) \leftarrow (v_{97}, \ldots, v_{207})$
for $i = 1$ **to** N **do**
  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 if
  $t_1 \leftarrow s_{65} \oplus s_{90} \cdot s_{91} \oplus s_{92} \oplus s_{170}$
  $t_2 \leftarrow s_{161} \oplus s_{174} \cdot s_{175} \oplus s_{176} \oplus s_{263}$
  $t_3 \leftarrow s_{242} \oplus s_{285} \cdot s_{286} \oplus s_{287} \oplus s_{68}$
  $(s_0, s_1, \ldots, s_{92}) \leftarrow (t_2, s_0, \ldots, s_{91})$
  $(s_{93}, s_{94}, \ldots, s_{176}) \leftarrow (t_0, s_{93}, \ldots, s_{175})$
  $(s_{177}, s_{178}, \ldots, s_{287}) \leftarrow (t_1, s_{177}, \ldots, s_{286})$
end for

7.2 The MILP-aided Algorithm for Trivium

Because Trivium is a public-update cipher, during the progress of recovering the ANF coefficients of superpoly, the set $\mathbb{K}$ is always empty. The papers [23, 26] have showed the method on how to build the CBDP model of Trivium. Here, we propose Algorithm 4 to get the $\mathbb{L}$'s propagation of Trivium's round function. The

Algorithm 4: The propagation of $\mathbb{L}$ for the round function

```
1  procedure CorePropagation(𝕃, i_0, i_1, i_2, i_3, i_4)
2      Let x = (x_0, x_1, x_2, x_3, x_4) be the variables
3      Let y be the function of x, and y = (x_0, x_1, x_2, x_3, x_0x_1 + x_2 + x_3 + x_4)
4      𝕃' = ∅
5      for ℓ in 𝕃
6          for all u = (u_0, u_1, u_2, u_3, u_4) ∈ F_2^5 do
7              if y^u contains the term x^(ℓ_{i_0}, ℓ_{i_1}ℓ_{i_2}, ℓ_{i_3}, ℓ_{i_4}) then
8                  ℓ' = ℓ
9                  ℓ'_{i_0} = u_0, ℓ'_{i_1} = u_1, ℓ'_{i_2} = u_2, ℓ'_{i_3} = u_3, ℓ'_{i_4} = u_4
10                 𝕃' <-x- ℓ'
11             end if
12         end for
13     end for
14     return 𝕃'
15 end procedure

1  procedure RoundPropagation(𝕃_r)
2      initial 𝕃' = ∅, 𝕃'' = ∅, 𝕃''' = ∅, 𝕃_{r+1} = ∅
3      𝕃' = CorePropagation(𝕃_r, 65, 170, 90, 91, 92)
4      𝕃'' = CorePropagation(𝕃', 161, 163, 174, 175, 176)
5      𝕃''' = CorePropagation(𝕃'', 242, 68, 285, 286, 287)
6      for all ℓ in 𝕃''' do
7          𝕃_{r+1} = 𝕃_{r+1} ⋃ {ℓ ⋙ 1}
8      end for
9      return 𝕃_{r+1}
10 end procedure
```

input of procedure *RoundPropagation* in Algorithm 4 is the r-th round BDPT set $\mathbb{L}_r$, and the outputs is the $(r+1)$-th round BDPT set $\mathbb{L}_{r+1}$.

At CRYPTO 2017 [23], Todo *et al.* proposed a CBDP based cube attack on the 832-round Trivium. Then, at CRYPTO 2018 [26], 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 3 to the 832-round and 839-round Trivium, we have the following results.

Result 1. *For cube set C_{I_v,J_v,K_v}, where $I_v = \{13, \ldots, 45, 47, \ldots, 58, 60, \ldots, 92\}$, no matter what the assignment to the non-cube IVs $\{46, 59\}$ is, the corresponding superpoly of 839-round Trivium in the paper [26] is constant. So the cube attack based on CBDP in the paper [26] is not key recovery attack.*

Result 2. *For the cube set C_{I_v,J_v,K_v}, where $I_v = \{13, 14, \ldots, 77, 79, 81, \ldots, 91\}$, the superpolies of some assignments are constant. For example, when $J_v = \{205, 206, 207\}$ and $K_v = \{0, 1, \ldots, 207\} - I_v - J_v$, the superpoly recovered is*

$p_{I_v,J_v,K_v}(\boldsymbol{x}) = 0$. *And the superpolies of some assignments are non-constant. For example, when* $J_v = \{80, 90, 205, 206, 207\}$ *and* $K_v = \{0, 1, \ldots, 207\} - I_v - J_v$, *the superpoly recovered is* $p_{I_v,J_v,K_v}(\boldsymbol{x}) = x_{56}x_{57}x_{58} + x_{32}x_{56} + x_{56}x_{59}$. *In a word, the assignment to the non-cube IVs will affect whether the cube attack on 832-round Trivium in the paper [23] is key recovery attack or not.*

7.3 Theoretical Result

Result 3. *Let* C_{I_v,J_v,K_v} *be a cube set, where* $I_v = \{13, 14, \ldots, 89, 91\}$, $J_v = \{205, 206, 207\}$, *and* $K_v = \{0, 1, \ldots, 204\} - I_v$. *Using the degree bounding technique in the paper [26], we can get that the degree of superpoly in 841-round Trivium is not larger than 10. Then, we have* $\sum_{i=0}^{d} \binom{|I|}{i} \leq \sum_{i=0}^{10} \binom{80}{i} \leq 2^{41}$. *That means we can use no more than* 2^{41} *MILP-aided propagation of BDPT to recover the exact superpoly of 841-round Trivium.*

Because our computing resources are limited, the exact superpoly of 841-round Trivium cannot be recovered in practical time. On our common PC (Intel Core i5-4590 CPU @3.3GHz, 8.00G RAM) , it takes about 18 days to complete the MILP-aided propagation of BDPT 100 times.

8 Conclusions

This paper is committed to solve the complexity problem of searching integral distinguishers based on BDPT. In order to make the propagation of BDPT efficient, we show the pruning techniques which can removing redundant vectors in time. Then, an algorithm is designed to estimate whether the m-th output bit is balanced or not based on BDPT. We apply the searching algorithm to some blocks, and the obtained integral distinguishers are the same or better than the previous longest integral distinguishers. It should be noted that the absence of integral distinguishers based on BDPT doesn't imply the absence of integral distinguishers. Any improvement on the accuracy of BDPT propagation may obtain better integral distinguishers. Moreover, our searching algorithm supposes that all round keys are chosen randomly. If consider the key scheduling algorithm, we may obtain better integral distinguishers.

Moreover, we apply BDPT to recover the superpoly in cube attack. 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. To verify the correctness and effectiveness of our method, we apply it to Trivium. For the cube attack on the 832-round Trivium [23], we obtain that only some proper non-cube IV assignments can obtain non-constant superpolies. For the cube attack on 839-round Trivium [26], 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 superpoly recovery of 841-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.

Acknowledgement. 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. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, M.: MILP modeling for (Large) S-boxes to optimize probability of differential characteristics. IACR Transactions on Symmetric Cryptology. **2017**(4), 99–129 (2017)
2. Beaulieu, R., Shors, D., Smith, J., Treatman–Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive 2013:404 (2013). http://eprint.iacr.org/2013/404
3. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
4. Boura, C., Canteaut, A.: Another view of the division property. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 654–682. Springer, Heidelberg (2016)
5. 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)
6. 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)
7. 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)
8. Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. In: Cid, C., Jacobson, Jr. M. (eds.) SAC 2018. LNCS, vol. 11349, pp. 115–138. Springer, Cham (2018)
9. 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)
10. Gurobi: http://www.gurobi.com/
11. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
12. 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
13. Hu, K., Wang, M.: Automatic search for a variant of division property using three subsets. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 412–432. Springer, Cham (2019)
14. 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)

15. 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)
16. 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)
17. Sage: http://www.sagemath.org/
18. Sun, B., Hai, X, Zhang, W,. Cheng, L., Yang, Z.: New observation on division property. Science China (Information Sciences), **2017**(09), 274–276 (2017)
19. 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. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014)
20. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015)
21. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (eds.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016)
22. 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)
23. 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)
24. Wu, W., Zhang, L.: LBlock: A lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
25. Wang, Q., Liu, Z., Varici, K., Sasaki, Y., Rijmen, V., Todo, Y.: Cryptanalysis of reduced-round SIMON32 and SIMON48. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 143–160. Springer, Cham (2014)
26. 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)
27. 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)
28. Xie, X., Tian, T.: Improved distinguisher search techniques based on parity sets. Sci China Inf Sci (2018)
29. Yang, G., Zhu, B., Suder, V., Aagaard, M., Gong, G.: The simeck family of lightweight block ciphers. In: Gneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 307–329. Springer, Heidelberg (2015)
30. Ye, C., Tian, T.: Deterministic cube attacks. IACR Cryptology ePrint Archive, 2018:1028 (2018). https://eprint.iacr.org/2018/1082.pdf
31. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences. **58**(12), 1–15 (2015)

Appendix

Table 7. The $\mathbb{L}$ propagation of BDPT for the core operation of SIMON

Input $\mathcal{D}^{1^4}_{\mathbb{K},\{\ell\}}$	Output $\mathcal{D}^{1^4}_{\mathbb{K}',\mathbb{L}'}$
$\boldsymbol{\ell} = [0,0,0,0]$	$\mathbb{L}' = \{[0,0,0,0]\}$
$\boldsymbol{\ell} = [1,0,0,0]$	$\mathbb{L}' = \{[1,0,0,0]\}$
$\boldsymbol{\ell} = [0,1,0,0]$	$\mathbb{L}' = \{[0,1,0,0]\}$
$\boldsymbol{\ell} = [1,1,0,0]$	$\mathbb{L}' = \{[1,1,0,0],[0,0,0,1],[1,0,0,1],[0,1,0,1],[1,1,0,1]\}$
$\boldsymbol{\ell} = [0,0,1,0]$	$\mathbb{L}' = \{[0,0,1,0],[0,0,0,1],[0,0,1,1]\}$
$\boldsymbol{\ell} = [1,0,1,0]$	$\mathbb{L}' = \{[1,0,1,0],[1,0,0,1],[1,0,1,1]\}$
$\boldsymbol{\ell} = [0,1,1,0]$	$\mathbb{L}' = \{[0,1,1,0],[0,1,0,1],[0,1,1,1]\}$
$\boldsymbol{\ell} = [1,1,1,0]$	$\mathbb{L}' = \{[1,1,1,0],[0,0,1,1],[1,0,1,1],[0,1,1,1],[1,1,0,1]\}$
$\boldsymbol{\ell} = [\ell_0,\ell_1,\ell_2,1]$	$\mathbb{L}' = \{[\ell_0,\ell_1,\ell_2,1]\}$

Experimental Verification

Example 1. For 591-round Trivium and cube set C_{I_v,J_v,K_v}, where $I_v = \{13, 23, 33, 43, 53, 63, 73, 83\}$, $J_v = \{14, 29, 32, 205, 206, 207\}$ and $K_v = \{0, 1, \cdots, 207\} - I_v - J_v$, we can get that the involved secret variables are $\{x_{22}, x_{23}, x_{24}, x_{66}\}$, the degree of superpoly is not larger than 2. Then, we use Algorithm 3 to recover all the ANF coefficients of the superpoly, which is in accordance with the practically recovered superpoly as follows:

$$p_{I_v,J_v,K_v}(\boldsymbol{x}) = x_{66} + x_{24} + x_{23}x_{22} + 1.$$

Example 2. For 591-round Trivium and cube set C_{I_v,J_v,K_v}, where $I_v = \{13, 23, 33, 43, 53, 63, 73, 83\}$, $J_v = \{29, 32, 82, 205, 206, 207\}$, and $K_v = \{0, 1, \cdots, 207\} - I_v - J_v$, we can get that the involved secret variables are $\{x_{22}, x_{23}, x_{24}, x_{65}, x_{66}\}$, the degree of superpoly is not larger than 3. Then, we use Algorithm 3 to recover the superpoly, which is in accordance with the practically recovered superpoly as follows:

$$p_{I_v,J_v,K_v}(\boldsymbol{x}) = x_{65}x_{23}x_{22} + x_{65}x_{24} + x_{66}x_{65} + x_{65}.$$