MILP Method of Searching Integral Distinguishers Based on Division Property Using Three Subsets*

Senpeng Wang[1], Bin Hu[1], Jie Guan[1], Kai Zhang[1], and Tairong Shi[1]

Zhengzhou Information Science and Technology Institute, Zhengzhou 450001, 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. The huge time and memory complexity that once restricted the applications of CBDP have been solved by Xiang et al. at ASIACRYPT 2016. They 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 can not be modeled efficiently. Thus it cannot be applied to block ciphers with block size larger than 32 bits. In this paper, we focus on the feasibility of applying MILP-aided method to search integral distinguishers based on BDPT. We firstly study how to get the BDPT propagation rules of an S-box. Based on that we can efficiently describe the BDPT propagation of cipher which has S-box. Moreover, we propose a technique called "fast propagation", which can translate BDPT into CBDP, then the balanced bits based on BDPT can be presented. Together with the propagation properties of BDPT, we can use MILP method based on CBDP to search integral distinguishers based on BDPT. In order to prove the efficiency of our method, we search integral distinguishers on SIMON, SIMECK, PRESENT, RECTANGLE, LBlock, and TWINE. For SIMON64, PRESENT, and RECTANGLE, we find more balanced bits than the previous longest distinguishers. For LBlock, we find a 17-round integral distinguisher which is one more round than the previous longest integral distinguisher, and a better 16-round integral distinguisher with less active bits can be obtain. For other ciphers, our results are in accordance with the previous longest distinguishers.

Keywords: Division property using three subsets · Integral distinguisher · MILP.

1 Introduction

Integral attack proposed by Knudsen and Wagner at FSE 2002 [7] is one of the most powerful tools used for block cipher. It is extended from square attack [4]

* Supported by organization x.

and has been applied to many block ciphers so far, such as Rijndael [9], ARIA [8] and Serpent [28]. The basic idea of integral attack is to analyze properties of the corresponding ciphertexts, such as zero-sum property.

Division property is a generalization of integral property proposed by Todo [19] at EUROCRYPT 2015. It can exploit the algebraic structure of block cipher to construct integral distinguishers even if the block cipher has non-bijective, bit-oriented, or low-degree structures. With this new technique, he detected a 6-round integral characteristic on MISTY1 [10], and achieved the first theoretical cryptanalysis of the full MISTY1 [17]. In order to exploit the concrete structure of round function, Todo and Morii [18] proposed conventional bit-based division property (CBDP), which use not only the algebraic degree but also the algebraic structure. Using CBDP, they explored a 14-round integral distinguisher of SIMON32 [1] which improved 4 rounds than [19]. Although CBDP could find more accurate integral distinguishers, it once required much time and memory complexity. They couldn't apply it to cipher whose block size was more than 32 bits. At CRYPTO 2016, Boura and Canteaut [3] introduced the parity set to exploit division property in another view. They formulated and characterized the division property, specially for the construction of division trails of S-box. They utilized the parity set to exploit further properties of the S-box and linear layer of PRESENT [2], leading to several improved distinguishers against reduced-round PRESENT. Since more properties of S-box and linear layer were utilized, parity sets could find more accurate integral characteristics. But it also required higher time and memory complexity than CBDP. In the paper [23], Xie and Tian proposed a concept called "term set" to propagate some information of Algebraic Normal Form (ANF). With term set, they improved the distinguisher searching method based on the parity set both in terms of memory and time complexity. And they found a 9-round distinguisher of PRESENT with 22 balanced output bits.

For block cipher whose block size is larger than 32, searching integral distinguishers by CBDP under Todo and Morii's framework once was infeasible just because of computational complexity. To solve the restriction of the huge complexity, Xiang et al. [22] applied MILP method to search integral distinguishers based on CBDP, which allowed them to analyze primitives whose block sizes were larger than 32. They analysed six block ciphers and found some longer integral distinguishers for SIMON, SIMECK [24], PRESENT, RECTANGLE [25], LBlock [20], and TWINE [12]. Then, Sun, Wang, and Wang gave the feasibility of MILP-aided CBDP for ciphers with non-bit-permutation linear layers [13, 14]. However the paper [26] found that the solutions of linear inequalities in [13] contained some impossible division trails, which may eventually lead the integral-trail searching come to a premature end and resulted in a shorter integral distinguisher. Then they found a way to give an accurate and compact description on the binary linear layer of block cipher. CBDP may find better integral distinguishers and it has been applied to various block ciphers. Take HIGHT as an example, in the paper [5], Funabiki, Todo, Isobe, and Morii proposed a new 19-round integral distinguisher by using the propagation of CBDP

and they improved the previous best attack by two rounds. At CRYPTO 2017, Todo, Isobe, Hao, and Meier applied CBDP in the cube attack on the stream ciphers ACORN, Grain128a, and TRIVIUM, and got the best key-recovery attack against these ciphers at that time.

CBDP proved the existence of the 14-round integral distinguisher of SIMON32 [18]. However, the experimental distinguisher covered 15 rounds [21], and there was still a one-round gap between the experiments and theoretical analysis. The paper [18] introduced a new variant called bit-based division property using three subsets (BDPT). CBDP focuses on that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is 0 or unknown, while BDPT focuses on that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is 0, 1, or unknown. Therefore, BDPT can find more accurate integral characteristics than CBDP. Using BDPT, they proved the existence of 15-round integral distinguisher of SIMON32. However, the propagation of BDPT requires more time and memory complexity than CBDP. When we evaluate BDPT propagation of an n-bit block ciphers directly, the complexity is about 2^n because BDPT has to manage the set of n-dimensional vectors whose elements take values in $\mathbb{F}_2$.

For designers, they want to know the number of rounds that has provable security against BDPT. Therefore, Todo and Morii [18] introduced a new technique called "lazy propagation" which could evaluate the number of rounds that BDPT couldn't find integral characteristics based on BDPT. But the number of rounds required to be provable secure has a gap to the existing integral distinguishers. Take SIMON128 as an example, SIMON128 has a 26-round integral distinguisher gotten by CBDP, but the result of provable security shows that 29-round SIMON128 doesn't have integral distinguishers based on BDPT. There is a question whether 27- or 28-round integral distinguishers of SIMON128 exist or not based on BDPT. For attackers, they want to find better integral distinguishers based on BDPT.

1.1 Our Contributions

In this paper, we propose a novel technique to search integral distinguishers based on BDPT by using the propagation properties of BDPT and MILP method.

BDPT Division Trails of S-box. Although the BDPT of any Boolean function can be evaluated by the propagation rules of Copy, And, and Xor, the propagation requires much time and memory complexity when Boolean function is complex. We propose an algorithm which can obtain the BDPT propagation rules of S-box according to its ANF directly.

Pruning Techniques of BDPT. When we evaluate the propagation of BDPT, there will be many vectors whose elements take values in $\mathbb{F}_2^n$. However there may be some vectors that have no impact on the output bit's balance based on BDPT. So we show the properties when the vectors of BDPT can be removed.

Fast Propagation and Stopping Rules. Inspired by the "lazy propagation", we propose the notion of "fast propagation" which translates BDPT into CBDP and can shows some bits is balanced. Then based on "lazy propagation" and "fast propagation", we obtain three stopping rules in searching integral distinguishers based on BDPT.

MILP Method of Searching Integral Distinguishers Based on BDPT. According to the propagation properties of BDPT, we design the MILP method to search integral distinguishers based on BDPT. In order to improve the efficiency, we simplify the MILP method of searching integral distinguishers based on CBDP in the paper [22] and divide the round function into several parts.

Applications to SIMON and SIMECK. Their round functions only consist of And, Rotation, and Xor. Our MILP method based on BDPT can find the 15-round integral distinguisher of SIMON32 that cannot be found by MILP method based on CBDP. It should be mentioned that, the 15-round distinguisher of SIMON32 is also the longest integral distinguisher that works for all keys which proved by experience. For 18-round SIMON64, we find 23 balanced bits which has one more bit than the previous longest integral distinguisher. For SIMON32/48/96/128 and SIMECK32/48/64, the integral distinguishers we find are in accordance with the longest existing integral distinguishers.

Applications to PRESENT and RECTANGLE. Both of them are SPN ciphers with bit permutation. 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 [23]. 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.

Applications to LBlock and TWINE. They are two generalized Feistel block ciphers. We apply our technique to these two ciphers. For LBlock, we obtain a 17-round integral distinguisher which is one more round than the previous longest integral distinguisher. Moreover, a better 16-round integral distinguisher with less active bits can be obtain. For TWINE, we find the existing longest 16-round distinguisher which is in accordance with that in [27].

The summarization of our main results is shown in Table 1. Note that all our experiments are conducted on a desktop which are very efficient, the details are listed in Table 1. In a word, our method can search integral distinguishers based on BDPT for ciphers with the block size larger than 32.

Outline of the Paper. The paper is organized as follows: Section 2 shows some notations, definitions, and basic backgrounds. In Section 3, we give some propagation properties of BDPT. Section 4 shows the method of searching integral distinguishers based on BDPT. Section 5 shows applications to SIMON,

SIMECK, PRESENT, RECTANLE, LBlock, and TWINE. Section 6 concludes the paper and summarizes our results. Some auxiliary materials are supplied in Appendix.

Table 1. Summarization of integral distinguishers for some block ciphers

Cipher	Data	Round	Number of balanced bits	Time	Reference
SIMON32	2^{31}	15	3		[18]
		15	3	1h48m	Sect. 5.1
SIMON48	2^{47}	16	24	48.2s	[22]
		16	24	1h48m	Sect. 5.1
SIMON64	2^{63}	18	22	6.7m	[22]
		18	**23**	23h31m	Sect. 5.1
SIMON96	2^{95}	22	5	17.4m	[22]
		22	5	31h25m	Sect. 5.1
SIMON128	2^{127}	26	3	58.4m	[22]
		26	3	62h16m	Sect. 5.1
SIMECK32	2^{31}	15	7	6.5s	[22]
		15	7	51m	Sect. 5.1
SIMECK48	2^{47}	18	5	56.6s	[22]
		18	5	5h3m	Sect. 5.1
SIMECK64	2^{63}	21	5	3m	[22]
		21	5	23h25m	Sect. 5.1
PRESENT	2^{60}	9	1	3.4m	[22]
		9	**4**	12h37m	Sect. 5.2
	2^{63}	9	22		[23]
		9	**28**	4h8m	Sect. 5.2
RECTANGLE	2^{60}	9	16	4.1m	[22]
		9	**27**	10m	Sect. 5.2
LBlock	2^{63}	16	32	4.9m	[22]
		17	4	98h30m	Sect. 5.3
	$\mathbf{2^{62}}$	16	3	41h29m	Sect. 5.3
TWINE	2^{63}	16	32	2.6m	[22]
		16	32	6m	Sect. 5.3

For SIMON and SIMECK, since the round key is Xored into the state after the round function, we can add one more round before the distinguishers using the technique in [21]. The results of SIMON and SIMECK family presented in the above table have been added by one round.

2 Preliminaries

2.1 Notations and Definitions

In this subsection, we present the notations and definitions used throughout this paper. Let $\mathbb{F}_2$ denote the finite field $\{0,1\}$ and $a\,[i]$ denotes the i-th bit of a , where a is an n-bit vector $(a[n-1],\ldots,a[0]) \in \mathbb{F}_2^n$. The Hamming weight of a is calculated as $w\,(a) = \sum_{i=0}^{n-1} a\,[i]$. For any $\boldsymbol{a} = (a_{m-1},\ldots,a_0) \in \mathbb{F}_2^{n_{m-1}} \times \cdots \times \mathbb{F}_2^{n_0}$, the vectorial Hamming weight of $\boldsymbol{a}$ is defined as $W\,(\boldsymbol{a}) = (w\,(a_{m-1}),\ldots,w\,(a_0)) \in \mathbb{Z}^m$, where $\mathbb{Z}$ denotes the integer ring. We use $\oplus$ and $+$ as the addition of $\mathbb{F}_2^n$ and addition of $\mathbb{Z}$.

For any $\boldsymbol{k} \in \mathbb{Z}^m$ and $\boldsymbol{k'} \in \mathbb{Z}^m$, define $\boldsymbol{k} \succeq \boldsymbol{k'}$ if $k_i \geq k_i'$ holds for all $i = 0,1,\ldots,m-1$ and $\boldsymbol{k} \succ \boldsymbol{k'}$ if $k_i > k_i'$ holds for all $i = 0,1,\ldots,m-1$. In this paper, $\mathbb{K}$ denotes the set of $\boldsymbol{k}$, and $|\mathbb{K}|$ is the number of elements in $\mathbb{K}$. Let $\emptyset$ be an empty set. We simply write $\mathbb{K} \leftarrow \boldsymbol{k}$ when $\mathbb{K} = \mathbb{K} \cup \{\boldsymbol{k}\}$ and $\mathbb{K} \rightarrow \boldsymbol{k}$ when $\mathbb{K} = \mathbb{K} \setminus \{\boldsymbol{k}\}$. Moreover, we write $\mathbb{K} \xleftarrow{x} \boldsymbol{k}$, where the new $\mathbb{K}$ is computed as

$$\mathbb{K} = \begin{cases} \mathbb{K} \cup \{\boldsymbol{k}\} \text{ if the original } \mathbb{K} \text{ does not include } \boldsymbol{k}, \\ \mathbb{K} \setminus \{\boldsymbol{k}\} \text{ if the original } \mathbb{K} \text{ includes } \boldsymbol{k}. \end{cases}$$

Definition 1. (Bit Product Function [19]). *For any $u \in \mathbb{F}_2^n$ and $x \in \mathbb{F}_2^n$, the bit product function $\pi_u : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is defined as*

$$\pi_u\,(x) = \prod_{i=0}^{n-1} x[i]^{u[i]}.$$

For any $\boldsymbol{u} = (u_{m-1},\ldots,u_0) \in \left(\mathbb{F}_2^{n_{m-1}} \times \cdots \times \mathbb{F}_2^{n_0}\right)$ and $\boldsymbol{x} = (x_{m-1},\ldots,x_0) \in \left(\mathbb{F}_2^{n_{m-1}} \times \cdots \times \mathbb{F}_2^{n_0}\right)$, the bit product function $\boldsymbol{\pi_u}\,(\boldsymbol{x})$ is defined as

$$\boldsymbol{\pi_u}\,(\boldsymbol{x}) = \prod_{i=0}^{m-1} \pi_{u_i}\,(x_i).$$

The bit product function also appears in the Algebraic Normal Form (ANF) of a Boolean function. The ANF of a Boolean function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is represented as

$$f\,(x) = \underset{u \in \mathbb{F}_2^n}{\oplus} a_u^f \left(\prod_{i=0}^{n-1} x[i]^{u[i]} \right) = \underset{u \in \mathbb{F}_2^n}{\oplus} a_u^f \pi_u\,(x)$$

where $a_u^f \in \mathbb{F}_2$ is a constant value depending on f and u.

2.2 Division Property and Conventional Bit-based Division Property

Division property [19] is a new method of finding integral characteristics. This subsection briefly shows its definition and propagation rules. For more details, please refer to [19].

Definition 2. (Conventional Division Property [19]). *Let $\mathbb{X}$ be a multiset whose elements take values from $\left(\mathbb{F}_2^{n_{m-1}} \times \cdots \times \mathbb{F}_2^{n_0}\right)$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{n_{m-1},\ldots,n_0}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose i-th element takes a value between 0 and n_i, it fulfills the following conditions:*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{u}}\left(\boldsymbol{x}\right) = \begin{cases} \textit{unknown} & \text{if there exists } \boldsymbol{k}\in\mathbb{K} \text{ satisfying } W\left(\boldsymbol{u}\right) \succeq \boldsymbol{k}, \\ 0 & \text{otherwise.} \end{cases}$$

If there are $\boldsymbol{k} \in \mathbb{K}$ and $\boldsymbol{k}' \in \mathbb{K}$ satisfying $\boldsymbol{k} \succeq \boldsymbol{k}'$ in the division property $\mathcal{D}_{\mathbb{K}}^{n_{m-1},\ldots,n_0}$, $\boldsymbol{k}$ can be removed from $\mathbb{K}$ because the vector $\boldsymbol{k}$ is redundant.

In the paper [18], Todo and Morii introduced a bit-based division property when $n_{m-1},\ldots,n_0$ were restricted to 1, i.e. $\mathcal{D}_{\mathbb{K}}^{1^m}$, we call it conventional bit-based division property (CBDP). Some propagation rules of CBDP are proven in [3, 15, 18, 19, 22].

CBDP Rule 1 (Copy). The output of a Copy function is calculated as $(y_1, y_0) = (x, x)$, where $x \in \mathbb{X}$ is the input. Assuming the input multiset $\mathbb{X}$ has division property $\mathcal{D}_{\{k\}}^{1}$, then the corresponding output multiset $\mathbb{Y}$ has division property $\mathcal{D}_{\mathbb{K}'}^{1^2}$ as follow:

$$\mathbb{K}' = \begin{cases} \{(0,0)\} & \text{if } k = 0, \\ \{(0,1),(1,0)\} & \text{if } k = 1. \end{cases}$$

CBDP Rule 2 (Xor). The output of an Xor function is calculated as $y = x_1 \oplus x_0$, where $(x_1, x_0) \in \mathbb{F}_2 \times \mathbb{F}_2$ is the input. Assuming the input multiset $\mathbb{X}$ has division property $\mathcal{D}_{\{\boldsymbol{k}\}}^{1^2}$, then the corresponding output multiset $\mathbb{Y}$ has division property $\mathcal{D}_{\mathbb{K}'}^{1}$ as follow:

$$\mathbb{K}' = \begin{cases} \{(0)\} & \text{if } \boldsymbol{k} = (0,0), \\ \{(1)\} & \text{if } \boldsymbol{k} = (0,1) \text{ or } (1,0), \\ \emptyset & \text{if } \boldsymbol{k} = (1,1). \end{cases}$$

CBDP Rule 3 (And). The output of an And function is calculated as $y = x_1 \wedge x_0$, where $(x_1, x_0) \in \mathbb{F}_2 \times \mathbb{F}_2$ is the input. Assuming the input multiset $\mathbb{X}$ has division property $\mathcal{D}_{\{\boldsymbol{k}\}}^{1^2}$, then the corresponding output multiset $\mathbb{Y}$ has division property $\mathcal{D}_{\mathbb{K}'}^{1}$ as follow:

$$\mathbb{K}' = \begin{cases} \{(0)\} & \text{if } \boldsymbol{k} = (0,0), \\ \{(1)\} & \text{otherwise.} \end{cases}$$

CBDP Rule 4 (S-box). For the propagation rules of S-box, the paper [3] introduced the table-aided CBDP to generate the propagation table of S-box. The CBDP propagation of S-box is related to its ANF. The papers [3, 15, 22] all showed the method of getting the CBDP propagation trails. Please refer to them for more information.

2.3 MILP-aided Conventional Bit-based Division Property

Although CBDP has been proved to be a powerful tool to find integral distinguishers, the time and memory complexity once restricted its applications to block ciphers whose block sizes were larger than 32 bits. At ASIACRYPT 2016, Xiang et al. [22] modeled CBDP propagations of three basic operations (Copy, And, Xor) and S-box operation by linear inequalities. Based on that they constructed a linear inequality system which could describe CBDP propagations of a block cipher given an initial division property. Then, by choosing an appropriate objective function, they converted a search algorithm under Todo's framework into an MILP problem. They solved the MILP problem by the openly available MILP optimizer Gurobi [6] to search integral distinguishers. In this subsection we will give a brief review of MILP-aided CBDP.

MILP Model 1. Let $a \to (b_1, b_0)$ be a division trail of Copy. The following inequalities are sufficient to describe the propagation of the division property for Copy.

$$\begin{cases} a - b_1 - b_0 = 0, \\ a, b_1, b_0 \text{ are binaries.} \end{cases}$$

MILP Model 2. Let $(a_1, a_0) \to b$ be a division trail of Xor. The following inequalities are sufficient to describe the propagation of the division property for Xor.

$$\begin{cases} a_1 + a_0 - b = 0, \\ a_1, a_0, b \text{ are binaries.} \end{cases}$$

MILP Model 3. Let $(a_1, a_0) \to b$ be a division trail of And. The following inequalities are sufficient to describe the propagation of the division property for And:

$$\begin{cases} a_i \leq b \text{ for all } i \in \{0, 1\}, \\ b - a_0 - a_1 \leq 0, \\ a_1, a_0, b \text{ are binaries.} \end{cases}$$

After generating the propagation property of S-box from CBDP Rule 4 in subsection 2.2, there are two main approaches that can be used to model S-box.

H-representation of the Convex Hull. By using the **inequality_generator()** function in Sage [11] software, we can get a set of linear inequalities $\mathcal{L}$. Sometimes the number of linear inequalities in the set is so large that adding all these inequalities into the MILP model will make the problem computational infeasible. Thus, Sun et al. [16] proposed an algorithm called Greedy Algorithm to reduce this set.

Logical Condition Modeling. For simplicity, if we want to remove the vector $(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) = (0, 0, 1, 1, 0, 1, 0, 1)$ from the solution space, we can specify $x_7 + x_6 - x_5 - x_4 + x_3 - x_2 + x_1 - x_0 \geq -3$, which removes this pattern while keeps all the other patterns in the solution space.

Note that MILP models for Copy, Xor, and And are sufficient to represent any circuit. Up to now, for block ciphers based on the three operations and S-box, we are able to construct a set of linear inequalities characterizing one round CBDP. Iterating this process r times, we can get a linear inequality system $\mathcal{L}$ describing r-round CBDP. All feasible solutions of $\mathcal{L}$ correspond to all r-round division trails, which are defined below.

Definition 3. (Division Trail [22]). *Let us consider the propagation of the division property* $\{\boldsymbol{k}\} \stackrel{def}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r$. *Moreover, for any vector* $\boldsymbol{k}_{i+1} \in \mathbb{K}_{i+1}$, *there must exist a vector* $\boldsymbol{k}_i \in \mathbb{K}_i$ *such that* $\boldsymbol{k}_i$ *can propagate to* $\boldsymbol{k}_{i+1}$ *for all* $i \in \{0, 1, \ldots r-1\}$, *we call* $(\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r)$ *an* r*-round division trail.*

Initial Division Property and Stopping Rule. Attackers determine indices set $I = \left\{i_{|I|-1}, i_{|I|-2}, \ldots, i_0, \right\} \subset \{0, 1, \ldots, n-1\}$ and prepare $2^{|I|}$ chosen plaintexts where variables indexed by I are taking all possible combinations of values. The division property of such chosen plaintexts is $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Then, the propagation of division property from $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$ is evaluated as

$$\{\boldsymbol{k}\} \stackrel{def}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r,$$

where $\mathcal{D}^{1^n}_{\mathbb{K}_r}$ is the division property after r-round propagation. If the division property $\mathbb{K}_r$ does not have an unit vector $\boldsymbol{e}_m$ whose only m-th element is 1, the m-th bit of r-round ciphertexts is balanced. Denote $\left(a^0_{n-1}, \ldots, a^0_0\right) \to \cdots \to \left(a^r_{n-1}, \ldots, a^r_0\right)$ an r-round division trail, and let $\mathcal{L}$ denote a linear inequality system whose feasible solutions are all division trails which start with the given initial division property. Thus, we can set the objective function as:

$$Obj: \ Min\left(a^r_{n-1} + a^r_{n-2} + \cdots + a^r_0\right).$$

Now we get a complete MILP problem by setting $\mathcal{L}$ as constraints and Obj as objective function. If $\mathbb{K}_{r+1}$ for the first time contains all the n unit vectors, the division property propagation should stop and an r-round distinguisher can be derived from $\mathcal{D}^{1^n}_{\mathbb{K}_r}$. For more details, please refer to [3, 15, 17–19, 22].

2.4 Bit-based Division Property Using Three Subsets and Lazy Propagation

CBDP focuses on that the parity $\bigoplus\limits_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{\pi_u}(\boldsymbol{x})$ is 0 or unknown. Because CBDP is insufficient to find the 15-round integral distinguisher of SIMON32, the paper [18] introduced a new variant of bit-based division property called bit-based division property using three subsets (BDPT). The new variant focuses on that the parity $\bigoplus\limits_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{\pi_u}(\boldsymbol{x})$ is 0, 1, or unknown.

Definition 4. (BDPT [18]). *Let* $\mathbb{X}$ *be a multiset whose elements take a value of* $\mathbb{F}_2^n$, *and* $\boldsymbol{k}$ *is a* n*-dimensional vector whose* i*-th element takes 0 or 1. When*

the multiset $\mathbb{X}$ *has the BDPT* $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$*, it fulfills the following conditions:*

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

When there is $\boldsymbol{k}\in\mathbb{K}$ satisfying $W(\boldsymbol{u}) \succeq \boldsymbol{k}$, $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is unknown even if there is $\boldsymbol{\ell}\in\mathbb{L}$ satisfying $W(\boldsymbol{u}) = \boldsymbol{\ell}$. We show the propagation rules of Copy, And, and Xor for BDPT, because any Boolean function can be evaluated by using these three rules. For more details, please refer to [18].

BDPT Rule 1 (Copy). Let $(x_{n-1}, x_{n-2}, \ldots, x_0) \in \mathbb{F}_2^n$ be the input of a Copy function, and $(x_{n-1}, x_{n-1}, x_{n-2}, \ldots, x_0)$ be the output. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}^{1^{n+1}}_{\mathbb{K}',\mathbb{L}'}$.

$$\mathbb{K}' \leftarrow \begin{cases} \{(0,0,k_{n-2},\ldots,k_0)\} & \text{if } k_{n-1}=0, \\ \{(1,0,k_{n-2},\ldots,k_0),(0,1,k_{n-2},\ldots,k_0)\} & \text{if } k_{n-1}=1, \end{cases}$$

$$\mathbb{L}' \leftarrow \begin{cases} \{(0,0,\ell_{n-2},\ldots,\ell_0)\} & \text{if } \ell_{n-1}=0, \\ \{(1,0,\ell_{n-2},\ldots,\ell_0),(0,1,\ell_{n-2},\ldots,\ell_0),(1,1,\ell_{n-2},\ldots,\ell_0)\} & \text{if } \ell_{n-1}=1. \end{cases}$$

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

BDPT Rule 2 (And). Let $(x_{n-1}, x_{n-2}, \ldots, x_0) \in \mathbb{F}_2^n$ be the input of And function and $(x_{n-1} \wedge x_{n-2}, \ldots, x_0)$ be the output. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}^{1^{n-1}}_{\mathbb{K}',\mathbb{L}'}$

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_{n-1}+k_{n-2}}{2} \right\rceil, k_{n-3}, \ldots, k_0\right),$$

$$\mathbb{L}' \leftarrow \left(\left\lceil \frac{\ell_{n-1}+\ell_{n-2}}{2} \right\rceil, \ell_{n-3}, \ldots, \ell_0\right),$$

where $\mathbb{K}'$ is computed from all $\boldsymbol{k}\in\mathbb{K}$ and $\mathbb{L}'$ is computed from all $\boldsymbol{\ell}\in\mathbb{L}$ satisfying $(\ell_{n-1}, \ell_{n-2}) = (0,0)$ or $(1,1)$.

BDPT Rule 3 (Xor). Let $(x_{n-1}, x_{n-2}, \ldots, x_0) \in \mathbb{F}_2^n$ be the input of Xor function and $(x_{n-1} \oplus x_{n-2}, x_{n-3}, \ldots, x_0)$ be the output. Assuming the input multiset $\mathbb{X}$ has $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$, then the output multiset $\mathbb{Y}$ has $\mathcal{D}^{1^{n-1}}_{\mathbb{K}',\mathbb{L}'}$

$$\mathbb{K}' \leftarrow (k_{n-1}+k_{n-2}, k_{n-3}, \ldots, k_0),$$

$$\mathbb{L}' \overset{x}{\leftarrow} (\ell_{n-1}+\ell_{n-2}, \ell_{n-3}, \ldots, \ell_0),$$

where $\mathbb{K}'$ is computed from all $\boldsymbol{k}\in\mathbb{K}$ satisfying $(k_{n-1}, k_{n-2}) = (0,0), (1,0)$, or $(0,1)$ and $\mathbb{L}'$ is computed from all $\boldsymbol{\ell}\in\mathbb{L}$ satisfying $(\ell_{n-1}, \ell_{n-2}) = (0,0)$, $(1,0)$, or $(0,1)$.

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

$$\begin{aligned}&\mathbb{K}' \leftarrow (\ell_{n-1}, \ell_{n-2}, \ldots, \ell_i \vee 1, \ldots, \ell_0)\,,\\ &\mathbb{L}' = \mathbb{L},\end{aligned}$$

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

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

BDPT can find more accurate integral characteristics than CBDP. However, the propagation of BDPT requires more time and memory complexity. By the "lazy propagation", Todo and Morii [18] gave the provable security of SIMON family against BDPT.

Definition 5. (Lazy Propagation). *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.*

3 The Propagation Properties of BDPT

We know that BDPT can find more accurate integral characteristics than CBDP. However, it need much time and memory complexity. In order to solve this problem, we need to explore the propagation properties of BDPT more deeply.

3.1 BDPT Division Trails of S-box

In the subsection 2.4, we have introduced the existing propagation rules of Copy, And, and Xor for BDPT. Although any Boolean function can be evaluated by using these three rules, the propagation requires much time and memory complexity when Boolean function is complex. Inspired by the algorithm of calculating CBDP division trails of S-box [22], we propose a generalized algorithm to calculate BDPT division trails of S-box . In **Algorithm 1**, $\boldsymbol{x} = (x_{n-1}, \ldots, x_0)$ and $\boldsymbol{y} = (y_{n-1}, \ldots, y_0)$ denote the input and output of an n-bit S-box respectively, and y_i is expressed as a Boolean function of $(x_{n-1}, \ldots, x_0)$.

Algorithm 1: Calculating the BDPT division trails of S-box

Input: The input BDPT of an n-bit S-box $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}\}}$, where $\boldsymbol{\ell}=(\ell_{n-1},\ldots,\ell_0)$
Output: The output BDPT $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$

1 **begin**
2 $\quad\mathbb{S}=\left\{\overline{\boldsymbol{k}}|\text{there is } \boldsymbol{k}\in\mathbb{K} \text{ satisfying } \overline{\boldsymbol{k}}\succeq\boldsymbol{k}\right\}$
3 $\quad F(X)=\left\{\boldsymbol{\pi}_{\overline{\boldsymbol{k}}}(\boldsymbol{x})\,|\overline{\boldsymbol{k}}\in\mathbb{S}\right\}$
4 $\quad\underline{\mathbb{K}}=\emptyset$ and $\underline{\mathbb{L}}=\emptyset$
5 $\quad$**for** $\boldsymbol{u}\in\mathbb{F}_2^n$ **do**
6 $\quad\quad$**if** $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial in $F(X)$ **then**
7 $\quad\quad\quad\underline{\mathbb{K}}=\underline{\mathbb{K}}\cup\{\boldsymbol{u}\}$
8 $\quad\quad$**end**
9 $\quad\quad$**if** $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contain the monomial $\boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$ **then**
10 $\quad\quad\quad\underline{\mathbb{L}}=\underline{\mathbb{L}}\cup\{\boldsymbol{u}\}$
11 $\quad\quad$**end**
12 $\quad$**end**
13 $\quad\mathbb{K}'=$***SizeReduce0***$(\underline{\mathbb{K}})$ and $\mathbb{L}'=$***SizeReduce1***$(\underline{\mathbb{K}},\underline{\mathbb{L}})$
14 $\quad$**return** $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$
15 **end**

We explain **Algorithm 1** line by line:
Line 2-3 According to the input BDPT $\mathcal{D}^{1^n}_{\mathbb{K},\{\boldsymbol{\ell}\}}$, the parity of monomial $\boldsymbol{\pi}_{\overline{\boldsymbol{k}}}(\boldsymbol{x})$ satisfying there is $\boldsymbol{k}\in\mathbb{K}$ satisfying $\overline{\boldsymbol{k}}\succeq\boldsymbol{k}$ over $\mathbb{X}$ is undetermined, and we store all these monomials in $F(X)$.
Line 4 Initialize $\underline{\mathbb{K}}$ and $\underline{\mathbb{L}}$ as empty set $\emptyset$.
Line 5-8 For any possible $\boldsymbol{u}$, if Boolean function $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial in $F(X)$, the parity of $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ over $\mathbb{X}$ is undetermined, and we store all these vectors in $\underline{\mathbb{K}}$.
Line 9-11 For any possible $\boldsymbol{u}$, if $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains the monomial $\boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$, we store all these vectors in $\underline{\mathbb{L}}$.
Line 13 The function ***SizeeReduce0*()** removes all the redundant vectors in $\underline{\mathbb{K}}$. Namely, if there is $\boldsymbol{u}',\boldsymbol{u}\in\underline{\mathbb{K}}$ satisfying $\boldsymbol{u}'\succeq\boldsymbol{u}$, the vector $\boldsymbol{u}'$ should be removed from $\underline{\mathbb{K}}$. And ***SizeReduce1*()** removes all the redundant vectors in $\underline{\mathbb{L}}$. That is, if there exist $\boldsymbol{u}\in\underline{\mathbb{K}},\boldsymbol{\ell}\in\underline{\mathbb{L}}$ satisfying $\boldsymbol{\ell}\succeq\boldsymbol{u}$, then $\boldsymbol{\ell}$ should be removed from $\underline{\mathbb{L}}$.
Line 14 Return $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$ as output.

The Precondition for the Propagation Rules of BDPT. The propagation rules of BDPT in subsection 2.4 simply regard $\underset{\boldsymbol{x}\in\mathbb{X}}{\oplus}(\boldsymbol{\pi}_{\boldsymbol{u}_1}(\boldsymbol{x})\oplus\boldsymbol{\pi}_{\boldsymbol{u}_2}(\boldsymbol{x}))$ as unknown if either $\underset{\boldsymbol{x}\in\mathbb{X}}{\oplus}\boldsymbol{\pi}_{\boldsymbol{u}_1}(\boldsymbol{x})$ or $\underset{\boldsymbol{x}\in\mathbb{X}}{\oplus}\boldsymbol{\pi}_{\boldsymbol{u}_2}(\boldsymbol{x})$ is unknown. Under this precondition, we will prove that our algorithm provides accurate propagation of BDPT.

Theorem 1. *Let $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}\}}$ be the input BDPT of an n-bit S-box, under the precondition for the propagation rules of BDPT, the output BDPT $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$ can be*

accurately described by $\mathcal{D}^{1^n}_{\underline{\mathbb{K}},\underline{\mathbb{L}}}$*, where*

$$\underline{\mathbb{K}} = \{\boldsymbol{u} | \boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \textit{ contains any momomial in } F(X)\},$$
$$\underline{\mathbb{L}} = \{\boldsymbol{u} | \boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \textit{ contains the momomial } \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})\}.$$

That is $\mathbb{K}' = \boldsymbol{SizeReduce0}(\underline{\mathbb{K}})$ *and* $\mathbb{L}' = \boldsymbol{SizeReduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}})$.

Proof. Let $\mathbb{K}'$ be the set of output BDPT that has no redundant vectors. According to the definition of BDPT and the precondition for the propagation rules, for any $\boldsymbol{u}$, if Booblean function $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial in $F(X)$, the parity $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is undetermined. Then we get $\boldsymbol{SizeReduce0}(\underline{\mathbb{K}}) \subset \mathbb{K}'$.

On the other hand, for any $\boldsymbol{u} \in \mathbb{K}'$, we have $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is undetermined. Suppose that $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ doesn't has any monomial in $F(X)$. Then if $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains the monomial $\boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$, we have $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) = 1$, or else $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) = 0$. The value of $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is determined, which is a contradiction. So for any $\boldsymbol{u} \in \mathbb{K}$, $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contain at least one monomial in $F(X)$. Therefore, $\mathbb{K}' \subset \underline{\mathbb{K}}$. Since $\mathbb{K}'$ has no redundant vectors, we can get $\mathbb{K}' \subset \boldsymbol{SizeReduce0}(\underline{\mathbb{K}})$. Altogether, we obtain $\mathbb{K}' = \boldsymbol{SizeReduce0}(\underline{\mathbb{K}})$.

Let $\mathbb{L}'$ be the set of output BDPT that has no redundant vectors. For any $\boldsymbol{u} \in \mathbb{L}'$, we have $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) = 1$. Since there is only one vector $\boldsymbol{\ell}$ in the input $\mathbb{L}$, the ANF of $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ must has the monomial $\boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$. Thus we get

$$\mathbb{L}' \subset \underline{\mathbb{L}} = \{\boldsymbol{u} | \boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \text{ contains the momomial } \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})\}.$$

Since the redundant vectors of $\underline{\mathbb{L}}$ do not affect whether the parity becomes 0, 1, or unknown, we have $\mathbb{L}' \subset \boldsymbol{SizeReduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}})$.

On the other hand, if $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains the monomial $\boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$, then

$$\begin{aligned}\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x}) \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \oplus \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})) \\ &= 1 \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \oplus \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})).\end{aligned}$$

Consider first the case $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \oplus \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$ does not contains any monomial in $F(X)$, then $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \oplus \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})) = 0$, we get $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) = 1$ and $\boldsymbol{u} \in \mathbb{L}'$. Now consider the case $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y}) \oplus \boldsymbol{\pi}_{\boldsymbol{\ell}}(\boldsymbol{x})$ contains some monomials in $F(X)$, then $\boldsymbol{u} \in \underline{\mathbb{K}}$, $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is unknown. According to the definition, when there is $\boldsymbol{k} \in \underline{\mathbb{K}}$ satisfying $W(\boldsymbol{u}) \succeq \boldsymbol{k}$, $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is unknown even if there is $\boldsymbol{\ell} \in \underline{\mathbb{L}}$ satisfying $W(\boldsymbol{u}) = \boldsymbol{\ell}$. So $\boldsymbol{u}$ is a redundant vector in $\underline{\mathbb{L}}$. Then we get $\boldsymbol{SizeReduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}}) \subset \mathbb{L}'$. Altogether, we obtain $\mathbb{L}' = \boldsymbol{SizeReduce1}(\underline{\mathbb{K}}, \underline{\mathbb{L}})$. □

we apply our algorithm to the core operation of SIMON family, the BDPT division trails we get is in accordance with [18]. Thus, we believe that our algorithm is sound. Algorithm 1 will return a table of the BDPT division trails when the input $\mathbb{L}$ has only one vector. If there are more than one vector in $\mathbb{L}$, the paper [18] showed an example on how to get its division trails from the table. Let $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}_{m-1},\boldsymbol{\ell}_{m-2},\ldots,\boldsymbol{\ell}_0\}}$ and $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}$ be the input and output BDPT of S-box,

respectively. According to the Algorithm 1, we can get the output BDPT $\mathcal{D}_{\mathbb{K}',\mathbb{L}'_i}$ from the input BDPT $\mathcal{D}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}_i\}}$, where $i = 0, 1, \ldots, m-1$. Therefore, we have

$$\mathbb{L}' = \{\boldsymbol{u} | \boldsymbol{u} \text{ appears odd times in sets } \mathbb{L}'_{m-1}, \mathbb{L}'_{m-2}, \ldots, \mathbb{L}'_0\}.$$

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

3.2 Pruning Techniques of BDPT

The previous works often divide cipher into r rounds, and investigate the CBDP or BDPT of round function. Round function 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 cipher into small parts, and after the BDPT propagation of a part we will use the BDPT properties to remove the redundant vectors. Then the propagation of the remaining vectors can continue efficiently.

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

Theorem 2. *For r-round cipher $E = Q_r \cdot Q_{r-1} \cdots \cdot Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$, if $\boldsymbol{k} \in \mathbb{K}_{i,j}$ cannot generate the output unit vector $\boldsymbol{e}_m$ of $\overline{E_{i,j}}$ based on CBDP, then whether $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ can generate the vector $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ based on BDPT or not is equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\to\boldsymbol{k},\mathbb{L}_{i,j}}$, where $\mathbb{K}_{i,j} \to \boldsymbol{k}$ denotes removing $\boldsymbol{k}$ from $\mathbb{K}_{i,j}$.*

Proof. In the subsection 2.4, we know that for public function, the 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 dependently based on CBDP, if $\boldsymbol{k} \in \mathbb{K}_r$ can not generate the output unit vector $\boldsymbol{e}_m$, then removing it from $\mathbb{K}_{i,j}$ doesn't have any impact on whether $\mathbb{K}_{r+1,0}$ has $\boldsymbol{e}_m$ or not. That is $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ has the same result with $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\to\boldsymbol{k},\mathbb{L}_{i,j}}$ on whether $\mathbb{K}_{r+1,0}$ has $\boldsymbol{e}_m$ based on BDPT or not. □

Theorem 3. *For r-round cipher $E = Q_r \cdot Q_{r-1} \cdot \cdots \cdot Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$. Then let $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$ be the initial CBDP of $\overline{E_{i,j}}$, if $\mathcal{D}^{1^n}_{\{\boldsymbol{\ell}\}}$ cannot generate the output unit vector $\boldsymbol{e}_m$ of $\overline{E_{i,j}}$ based on CBDP, we obtain whether $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ can generate the vector $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ based on BDPT or not is equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$, where $\mathbb{L}_{i,j} \to \boldsymbol{\ell}$ denotes removing $\boldsymbol{\ell}$ from $\mathbb{L}_{i,j}$.*

Proof. For the input BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ of $\overline{E_{i,j}}$, if $\mathcal{D}^{1^n}_{\{\boldsymbol{\ell}\}}$ cannot generate the output unit vector $\boldsymbol{e}_m$ of $\overline{E_{i,j}}$ based on CBDP, where $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$. According to Theorem 2, whether $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$ can generate $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ or not is equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$. Then by the definition of BDPT, $\boldsymbol{\ell}$ is unknown, even though $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$, because of $\mathbb{K}_{i,j} \leftarrow \boldsymbol{\ell}$. So whether $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$ can generate $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ based on BDPT or not is equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\leftarrow\boldsymbol{\ell},\mathbb{L}_{i,j}}$. Moreover, according to Theorem 2 again, it equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$. So we get the result, whether $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ can generate the vector $\boldsymbol{e}_m \in \mathbb{K}_{r+1,0}$ based on BDPT or not is equivalent to $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}\to\boldsymbol{\ell}}$. □

Because the propagation of CBDP can be efficiently solved by MILP model. The meaning of Theorem 2 and Theorem 3 is that we can use CBDP method to determine whether the vectors $\boldsymbol{k} \in \mathbb{K}_{i,j}$ and $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$ are redundant or not to the propagation of BDPT. Dividing a cipher into small parts guarantees that we can remove redundant vectors timely and prevent them from expanding too much. In subsection 5.1, we will compared the size of BDPT vectors obtained by our method with that in [18] to show its high efficiency.

3.3 Fast Propagation

Because it needs much time and memory complexity to evaluate the propagation of BDPT, Todo and Morii proposed the concept of "lazy propagation" to guarantee the security against BDPT. The "lazy propagation" can show that some bits are unknown. Inspired by it, we propose a notion called "fast propagation" to show the balanced bits. Combining "lazy propagation" with "fast propagation", we can obtain the balanced information of the output bits efficiently.

Definition 6. (Fast Propagation). *For r-round cipher $E = Q_r \cdot Q_{r-1} \cdots \cdot Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$, then 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 division property of $\overline{E_{i,j}}$ is computed from all vectors of $\overline{\mathbb{K}}_{i,j} = \mathbb{K}_{i,j} \cup \mathbb{L}_{i,j}$ based on CBDP.*

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 propagation problem 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 $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$, according to the definition of BDPT and CBDP, the set must also satisfies $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j}\cup\mathbb{L}_{i,j}}$,. We get $\mathcal{D}^{1^n}_{\mathbb{K}_{r+1,0},\mathbb{L}_{r+1,0}}$ and $\mathcal{D}^{1^n}_{\overline{\mathbb{K}}_{r+1,0}}$, by the propagation of BDPT and the "fast propagation", respectively. Then, the set of $\boldsymbol{u}$ satisfying parity is 0 is represented as

$$\mathbb{S}_{\mathbb{K}_{r+1,0}} = \{\boldsymbol{u} \in \mathbb{F}_2^n | \text{ there is no } \boldsymbol{k} \in \mathbb{K}_{r+1,0} \text{ satisfying } W(\boldsymbol{u}) \succeq \boldsymbol{k}\}.$$

Though the complementary set of $\mathbb{S}_{\overline{\mathbb{K}}_{r+1,0}}$ cannot completely represent the set of $\boldsymbol{u}$ that the parity is unknown, $\mathbb{S}_{\overline{\mathbb{K}}_{r+1,0}} \subseteq \mathbb{S}_{\mathbb{K}_{r+1,0}}$ always holds.

4 The Method of Searching Integral Distinguishers Based on BDPT

Based on the work of [22], 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-aided Method of CBDP

Using the method in the paper [22], we can get a linear inequality set $\mathcal{L}$ which describes the r-round CBDP division trails with the given initial input division property $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$. The former CBDP method will return a set of balanced bits. Since the absence of integral distinguisher based on division property doesn't imply the absence of integral distinguishers, the bits that are not balanced based on CBDP will be viewed as unknown. Since only one bit's balanced information is needed, our MILP model has no objective function which is added into the constrains. Let $\mathcal{L}' = \mathcal{L} \cup \left\{a^r_{n-1} = 0, \ldots, a^r_{m+1} = 0, a^r_m = 1, a^r_{m-1} = 0, \ldots, a^r_0 = 0\right\}$, we can use Gurobi to determine whether $\mathcal{L}'$ has feasible solutions or not. If it has feasible solutions, it shows that the m-th bit of the output of cipher E is unknown. Otherwise, the m-th bit is balanced. The detail information is shown in Algorithm 2.

Algorithm 2 *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     L is a linear inequality set which describe the CBDP division trails of E
      with given initial division property D_{{k}}
3     Let L' = L ∪ {a^r_{n-1} = 0, ..., a^r_{m+1} = 0, a^r_m = 1, a^r_{m-1} = 0, ..., a^r_0 = 0}
4     if L' has feasible solutions do
5         return unknown
6     else
7         return balanced
8 end
```

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.

Stopping Rule 1. *For r-round cipher $E = Q_r \cdot Q_{r-1} \cdot \cdots \cdot Q_1$, let $\mathcal{D}^{1^n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$ be the input BDPT of $\overline{E_{i,j}}$, If $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$ can generate the output unit vector $\boldsymbol{e}_m$ of $\overline{E_{i,j}}$*

based on CBDP, where $\boldsymbol{k} \in \mathbb{K}_{i,j}$*. Then 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 vector in $\mathbb{K}_{i,j}$ will be removed according to the pruning techniques. 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 techniques in subsection 3.2, if there is element* $\boldsymbol{\ell} \in \mathbb{L}_{i,j}$ *that can generate the output unit vector* $\boldsymbol{e}_m$ *of* $\overline{E_{i,j}}$ *based on CBDP, we cannot stop the procedure and* $\boldsymbol{\ell}$ *should be propagated to next part based on BDPT. If there is no element in* $\mathbb{L}_{i,j}$ *that can generate* $\boldsymbol{e}_m$*, according to "fast propagation", we can get that the m-th output bit of E is balanced.*

Different from Stopping Rule 1 which shows some bits is unknown, Stopping Rule 2 can show some bits 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* $\mathbb{L}_{r+1,0} = \{\boldsymbol{\ell}\}$*, then we find an integral distinguisher of E whose sum is 1.*

When this situation happens, we find an integral distinguisher that cannot be found by CBDP.

4.3 The MILP Method of Searching Integral Distinguishers Based on BDPT

The algorithm of searching integral distinguishers often has a given initial division property $\mathcal{D}^{1^n}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$. Attackers determine indices $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{1, 2, \ldots, n\}$ and prepare $2^{|I|}$ chosen plaintexts where variables indexed by I are taking all possible combinations of values. The CBDP of such chosen plaintexts is $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise. Then, the BDPT of such chosen plaintexts is $\mathcal{D}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$, where $\mathbb{K}_{1,0} = \{\boldsymbol{k}' | \boldsymbol{k}' \succ \boldsymbol{k}\}$ and $\mathbb{L}_{1,0} = \{\boldsymbol{k}\}$.

Using the propagation rules of BDPT in subsection 2.4 and 3.1, we can obtain output BDPT from the input when function is not very complexity. Then the pruning techniques can simplify output BDPT to avoid it expanding too much. If the BDPT satisfies any stopping rules in subsection 4.2, we obtain the balanced information of corresponding output bit. Algorithm 3 will return whether the m-th output bit is balanced or not based on BDPT.

We explain **Algorithm 3** line by line:

Line 2-3 When we describe the BDPT propagation of $E = Q_r \cdot Q_{r-1} \cdot \cdots \cdot Q_1$, there will be redundant vectors. We have to divide the cipher into small parts so that the BDPT vectors don't expand too much. For every part, we will remove the redundant BDPT vectors and consider the stopping rules.

Line 4-9 For every vector in $\mathbb{K}_{i,j}$, we use Algorithm 1 to get whether it can generate unit vector $\boldsymbol{e}_m$ based on CBDP or not. If $SCBDP\left(\overline{E_{i,j}}, \boldsymbol{k}, m\right)$ is unknown, according to Stopping Rule 1, we known that the m-th output bit is unknown

Algorithm 3 $BDPT(E, \mathbb{L}_{1,0}, \mathbb{K}_{1,0}, m)$

Input: The cipher E, the input initial BDPT $\mathcal{D}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$, and the number m
Output: Whether the m-th bit of output is balanced or not 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 balanced
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 end
```

based on BDPT. Otherwise $\boldsymbol{k}$ cannot generate the unit vector $\boldsymbol{e}_m$, 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 $SCBDP\big(\overline{E_{i,j}}, \boldsymbol{\ell}, m\big)$ can generate the unit vector $\boldsymbol{e}_m$ of the output BDPT, it doesn't satisfy the pruning technique in Theorem 3, and 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 After the process of pruning techniques and stopping rules, 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 $\mathcal{D}_{\mathbb{K}_{i+\lfloor (j+1)/l_i \rfloor,(j+1)\%l_i},\mathbb{L}_{i+\lfloor (j+1)/l_i \rfloor,(j+1)\%l_i}}$ of the next part.

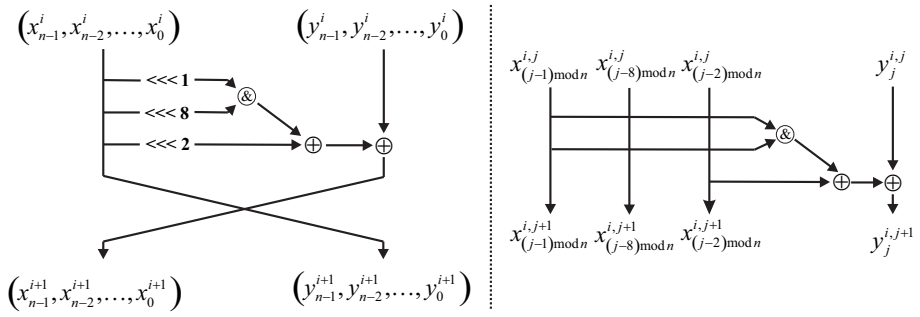
The principle of determining the size of $Q_{i,j}$ is that the BDPT set doesn't expand too much. Only in this way, can we run the searching algorithm efficiently. Algorithm 3 can show the balanced information of the m-th output bit. Therefore, we can search the integral distinguisher by exploring n-bit output of the cipher one by one.

5 Applications to SIMON, SIMECK, PRESENT, RECTANGLE, LBlock, and TWINE

In this section, we show some applications of our technique. In [22], Xiang et al. have applied MILP method to search integral distinguishers based on CBDP for 6 lightweight block ciphers SIMON, SIMECK, PRESENT, RECTANGLE, LBlock, and TWINE. We apply our technique to them too, we want to know whether BDPT can improve the integral distinguishers gotten by CBDP or not. The results listed in Table 1 show that we get improved distinguishers for SIMON64, PRESENT, RECTANGLE, and LBlock. For SIMON32/48/96/128, SIMECK32/48/64, and TWINE our results are consistent with the previous best results. Especially, for SIMON 32 we find the 14-round distinguisher that found in [18] used BDPT. It shows that our algorithm is valid. All the experiments are conducted on the following platform: Intel Celeron CPU 1007U @1.5GHz, 6.00 RAM, 64-bit Windows system. And the optimizer we used to search integral distinguishers is Gurobi 7.5.2 [6].

5.1 Applications to SIMON and SIMECK

SIMON is a lightweight block cipher family [1] 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 $n \in \{16, 24, 32, 48, 64\}$. The input of the i-th round function is denoted by $\left(x_{n-1}^i, \ldots, x_0^i, y_{n-1}^i, \ldots, y_0^i\right)$. 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. In every core operation, we only focus on four bits and evaluate the propagation independent of other $(2n-4)$ bits. The round function of SIMON2n repeats core operation for all n-bit values in the right half. For more details please refer to [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 3 to SIMON2n, we divide one-round SIMON2n into $n+1$ parts $Q_i = Q_{i,n} \cdot Q_{i,n-1} \cdot \cdots \cdot Q_{i,0}$.

When $0 \le j \le n-1$, we have

$$\begin{aligned} &Q_{i,j}\left(x_{n-1}^{i,j}, \ldots, x_0^{i,j}, y_{n-1}^{i,j}, \ldots, y_0^{i,j}\right) \\ &\quad = \left(x_{n-1}^{i,j}, \ldots, x_0^{i,j}, y_{n-1}^{i,j}, \ldots, y_{j+1}^{i,j}, Y_j^{i,j}, y_{j-1}^{i,j}, \ldots, y_0^{i,j}\right), \end{aligned}$$

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, $Q_{i,n}$ is

$$\begin{aligned} &Q_{i,n}\left(x_{n-1}^{i,n}, \ldots, x_0^{i,n}, y_{n-1}^{i,n}, \ldots, y_0^{i,n}\right) \\ &\quad = \left(y_{n-1}^{i,n} \oplus k_{n-1}^{i}, \ldots, y_0^{i,n} \oplus k_0^{i}, x_{n-1}^{i,n}, \ldots, x_0^{i,n}\right). \end{aligned}$$

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 Algorithm 1 to get its accurate BDPT propagation rules which are in accordance with that in the paper [18]. We show it in Appendix Table 8.

When we use Algorithm 3 to search the integral distinguishers of SIMON2n based on BDPT, we should call Algorithm 2 to build the MILP model based on CBDP. The paper [22] shows us how to model one-round CBDP division trail of SIMON2n.

1-round Description of SIMON Denote 1-round division 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 division 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 1-round bit-wise And operation used in SIMON2n can be modeled by the following inequalities:

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

At last, the Xor operations in SIMON2n can be modeled by the following inequalities:

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

So far, we get a description $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$ of 1-round division trails. By repeating this procedure r times, we can get a linear inequality system $\mathcal{L}$ for r-round CBDP propagation. Given the initial CBDP, we can add corresponding constrains into $\mathcal{L}$ and estimate whether an integral distinguisher exists by Algorithm 2.

How to Describe the CBDP Division Trails of Partial Round. For $\overline{E_{i,j}}$, the first round maybe a partial round $Q_{i,l_i-1} \cdot Q_{i,l_i-2} \cdot \cdots \cdot Q_{i,j}$, we can use the propagation rules of Copy, And, and Xor to obtain its linear inequality system directly. But there will be n different situations which will make program code complex. We observe that when we consider the CBDP propagation of $Q_{i,j}$, if add constrain $b_j^{i+1,j} = b_j^{i,j}$, the output vector set is the same with input vector, that is $Q_{i,j}$ is transformed into identical function. For example, when the input CBDP of core operation is $\mathcal{D}_{\{\boldsymbol{k}=(1,0,1,0)\}}$, according to the Table 7 in Appendix, the output CBDP of core operation is $\mathcal{D}_{\mathbb{K}}^{1^4}$, where $\mathbb{K} = \{(1,0,1,0), (0,0,1,1), (1,0,0,1)\}$. When adding the constrain $b_j^{i+1,j} = b_j^{i,j}$, we get $\mathbb{K} = \{(1,0,1,0)\}$. So it may make the core operation become identical function.

For partial round $Q_{i,l_i-1} \cdot Q_{i,l_i-2} \cdot \cdots \cdot Q_{i,j}$, we let the $Q_{i,j-1} \cdot Q_{i,j-2} \cdot \cdots \cdot Q_{i,0}$ be identical function. 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 get a description $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4\}$ of partial round $Q_{i,l_i-1} \cdot Q_{i,l_i-2} \cdot \cdots \cdot Q_{i,j}$. Then by repeating the constrains of one round $r - i$ times, we can get a linear inequality system $\mathcal{L}$ for $\overline{E_{i,j}}$. Given some initial division property, we can add the corresponding constrains into $\mathcal{L}$ and estimate whether a distinguisher based on CBDP exists or not by Algorithm 2.

How to Obtain the Output BDPT of $Q_{i,j}$. After the pruning techniques and stopping rules, if Algorithm 3 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 here.

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 $\boldsymbol{\ell}_1 = (1, \mathbf{0}, \mathbf{1}, 1, 1, 1, 1, 1, 1, \mathbf{1}, 1, 1, 1, 1, 1, 1, 1, 1, \mathbf{0}, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$, and $\boldsymbol{\ell}_2 = (1, \mathbf{0}, \mathbf{0}, 1, 1, 1, 1, 1, 1, \mathbf{1}, 1, 1, 1, 1, 1, 1, 1, 1, \mathbf{1}, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$. 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 8. The output vector set is $\mathbb{L}_1' = \{[0, 1, 1, 0], [0, 1, 0, 1], [0, 1, 1, 1]\}$. So the propagation from $\boldsymbol{\ell}_1$ generates three vectors as:

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

In the same way, we can obtain that the propagation from $\boldsymbol{\ell}_2$ generates only one vectors $(1,\mathbf{0},\mathbf{0},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1)$.

According to the **BDPT Rule 3 (Xor)**, the vector $(1,\mathbf{0},\mathbf{0},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1)$ should be cancelled because it is propagated from $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ twice. So 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 $\boldsymbol{\ell}_3 = (1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{0},1,1,1,1,1,1,1,1,1,1,1,1,1,1)$ and $\boldsymbol{\ell}_4 = (1,\mathbf{0},\mathbf{1},1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,\mathbf{1},1,1,1,1,1,1,1,1,1,1,1,1,1,1)$.

Finally, $Q_{1,16}$ has round keys Xored operation, So new vectors are generated from $\mathbb{L}_{1,16}$ according to the **BDPT Rule 4**, and the new vectors are inserted into $\mathbb{K}_{1,16}$. Moreover, some vectors in $\mathbb{L}_{1,16}$ becomes redundant because of the new vectors 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 $\boldsymbol{k} = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$ and $\boldsymbol{\ell}_5 = (0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1)$.

The High Efficiency of Our Algorithm. For SIMON32, we prepare chosen plaintexts such that the leftmost bit is constant and the others are active, the set of chosen plaintexts has $\mathcal{D}_{\{(1,1,1,\ldots,1)\},\{(0,1,1,\ldots,1)\}}$. Take 14-round SIMON32 as an example, if we want to obtain the balanced information of the rightmost output bit, we have to evaluate the propagation characteristic of BDPT. Table 2 shows the size of $|\mathbb{K}|$ and $|\mathbb{L}|$ in every round. The size in the paper [18] is obtained after removing redundant vectors according to the definition of BDPT, while the size in this paper is obtained after the pruning techniques. From Table 2, 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.

Table 2. 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
[18]	$\|\mathbb{L}\|$	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

The Integral Distinguishers of SIMON and SIMECK Based on BDPT. SIMECK is a family of lightweight block cipher proposed at CHES 2015 [24], and its round function is very similar to that of SIMON except the rotation constants. The rotation number is changed from $(1,8,2)$ to $(0,5,1)$. Let SIMECK2n be the SIMECK block ciphers with $2n$-bit block length, where n is chosen from 16, 24, and 32. We use Algorithm 3 to search the integral distinguishers of SIMON and SIMECK family based on BDPT. 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 SIMON32/48/96/128 and

SIMECK32/48/64, the distinguishers we find are in accordance with the previous longest distinguishers. Especially for SIMON32, our MILP algorithm finds the 14-round integral distinguisher that found in [18] by going through all the BDPT division trails. It shows the valid of our algorithm.

Since round keys are Xored after the round function in SIMON and SIMECK, we can trivially get the $(r+1)$-round integral characteristic from the r-round integral distinguisher. Like Table 1, the results of SIMON and SIMECK present in Table 3 have been added by one round. What needs to be explained is that "a" denotes active bit, "c" denotes constant bit, "?" denotes the balanced information in unknown based on BDPT, and "b" denotes the balanced bit.

Table 3. Integral distinguishers for SIMON and SIMECK

Cipher	Distinguisher
15-SIMON32	In: (caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa) Out: (????????????????, ?b??????b??????b)
16-SIMON48	In: (caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????????, bbbbbbbbbbbbbbbbbbbbbbbb)
18-SIMON64	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????????????????,bbbbbbbbbbb??b??b?????bbbbbbbbbb)
22-SIMON96	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (??, b?b????b?????????????????????????????????b????b?)
26-SIMON128	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (??, b?b???b?)
15-SIMECK32	In: (caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa) Out: (????????????????, bb???bb???bb???b)
18-SIMECK48	In: (caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????, b???bb?????????????bb???)
21-SIMECK64	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????????????,bb???b?????????????????????b???b)

5.2 Applications to PRESENT and RECTANGLE

PRESENT 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. For more details, please refer to [2].

We divide one-round PRESENT into 17 parts $Q_i = Q_{i,16} \cdot Q_{i,15} \cdot \cdots \cdot Q_{i,0}$. When $0 \leq j \leq 15$, we have

$$Q_{i,j}\left(x_{63}^{i,j}, x_{62}^{i,j}, \ldots, x_{0}^{i,j}\right) = \left(x_{63}^{i,j}, \ldots, S\left(x_{4j+3}^{i,j}, x_{4j+2}^{i,j}, x_{4j+1}^{i,j}, x_{4j}^{i,j}\right), \ldots, x_{0}^{i,j}\right),$$

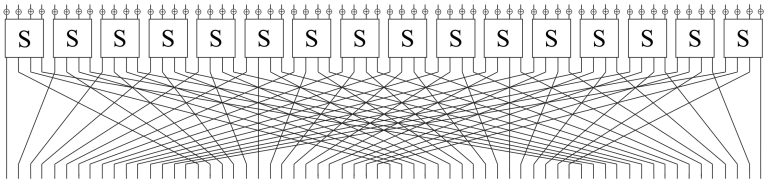


Fig. 2. One-round SP structure of PRESENT

where $S\left(x_{4j+3}^{i,j}, x_{4j+2}^{i,j}, x_{4j+1}^{i,j}, x_{4j}^{i,j}\right)$ is the S-box of PRESENT. When $j = 16$, we have

$$Q_{i,16}\left(x_{63}^{i,16}, x_{62}^{i,16}, \ldots, x_{63}^{i,16}\right) = P\left(x_{63}^{i,16}, x_{62}^{i,16}, \ldots, x_{0}^{i,16}\right) \oplus \boldsymbol{k}_i,$$

where P is the linear permutation function of PRESENT and $\boldsymbol{k}_i$ is the i-th round key.

The paper [22] has studied how to calculate the CBDP division trails of S-box and model those trails by linear inequalities. In this part, we only list the $\mathbb{L}$ propagation property of PRESENT's S-box in Appendix Table 9 gotten by Algorithm 1. The linear layer of PRESENT is a bit permutation, thus the propagation of BDPT through linear layer is just a permutation of the vector. Finally, when round keys are Xored with the middle state, new vectors are generated from $\mathbb{L}$, and the new vectors are inserted into $\mathbb{K}$. If the partial round doesn't has $Q_{i,j}$, we only need to avoid the responding linear inequalities. So we can easily complete the algorithm $SCBDP\left(\overline{E_{i,j}}, \boldsymbol{k}, m\right)$ and $BDPTP\left(Q_{i,j}, \mathcal{D}_{\emptyset,\mathbb{L}'_{i,j}}\right)$.

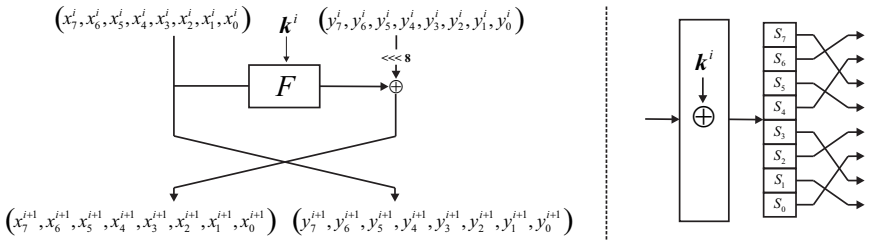
RECTANGLE is very like to PRESENT, for more detail information please refer to the paper [25]. We list the $\mathbb{L}$ propagation property of RECTANGLE's S-box in Appendix Table 10. We apply Algorithm 3 to PRESENT and RECTANGLE, and the results are listed in Table 4. For 9-round PRESENT, when the input plaintexts is 2^{60} we find a better integral distinguisher which has 3 more balanced bits than before. When 2^{63} plaintexts are used, the paper [23] gave a 9-round integral distinguisher with 22 balanced bits, while our algorithm can obtain 28 balanced bits. For RECTANGLE, when the input plaintexts is 2^{60}, the best integral distinguisher is 9 rounds with 16 balanced bits. In this paper we find a better 9-round distinguisher which has 27 balanced bits.

Table 4. Integral distinguishers for PRESENTand RECTANGLE

Cipher	Distinguisher
9-PRESENT	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaacccc) Out: (????????????????????????????????,??????????????????b???b???b???b)
9-PRESENT	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac) 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 and TWINE

LBlock is a lightweitht block cipher proposed by Wu and Zhang [20]. The block size is 64 bits and the key size is 80 bits. It employs a variant Feistel structure and consists of 32 rounds. The output nibble of every S-box in Xored with some another nibble of the right branch. The non-linear layer uses eight 4-bit S-boxes and we do not list these S-boxes due to the space limitation. One-round structure of LBlock is given in Fig. 3. For a more detailed description, please refer to [20].



(a) i-th round structure of LBlock (b) The structure of F-function

Fig. 3. Round structure of LBlock

Table 5. The diffusion function P

j	0	1	2	3	4	5	6	7
$P(j)$	2	0	3	1	6	4	7	5

We divide one-round LBlock into 9 parts $Q_i = Q_{i,8} \cdot Q_{i,7} \cdot \cdots \cdot Q_{i,0}$. When $0 \leq j \leq 7$, we have

$$\begin{aligned} &Q_{i,j}\left(x_7^{i,j}, \ldots, x_0^{i,j}, y_7^{i,j}, \ldots, y_0^{i,j}\right) \\ &\quad = \left(x_7^{i,j}, \ldots, x_0^{i,j}, y_7^{i,j}, \ldots, y_{P(j)+1}^{i,j}, Y_{P(j)}^{i,j}, y_{P(j)-1}^{i,j}, \ldots, y_0^{i,j}\right), \end{aligned}$$

where $Y_{P(j)}^{i,j} = S_j\left(x_j^{i,j} \bigoplus k_{i,j}\right) \bigoplus y_{(P(j)-2)\bmod 8}^{i,j}$, S_j is the j-th S-box of LBlock, and $P(x)$ is the nibble diffusion function as shown in Table 5. For $Q_{i,8}$, we have

$$Q_{i,8}\left(x_7^{i,8}, \ldots, x_0^{i,8}, y_7^{i,8}, \ldots, y_0^{i,8}\right) = \left(y_7^{i,8}, \ldots, y_0^{i,8}, x_7^{i,8}, \ldots, x_0^{i,8}\right).$$

For $Q_{i,j}$, $0 \leq j \leq 7$, when we consider the BDPT propagation rules of the function $BDPTP\left(Q_{i,j}, \mathcal{D}_{\emptyset,\mathbb{L}'_{i,j}}\right)$ in Algorithm 3, the 56 bits remain unchanged, and there will be 8 bits $\left(x_j^{i,j}, y_{P(j)}^{i,j}\right)$ of the BDPT vectors will be changed. We

can view it as 8-bit S-box to get its output BDPT. It should be valued that the round key Xor operation is in the front of S-box, we have to build a table to describe the impact of $\mathbb{L}$ to $\mathbb{K}$.

After generating the propagation table for every enlarged S-box, we can use Algorithm 3 to find integral distinguishers for LBlock. We find a new 17-round integral distinguisher of LBlock which is one more round than the previous longest integral distinguisher, and a better 16-round integral distinguisher with less active bits can be obtain. The detail forms of the integral distinguishers are shown in Table 6.

Table 6. Integral distinguishers for LBlock and TWINE

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

TWINE is a 64-bit lightweight block cipher supporting 80- and 128-bit keys and has the alike structure as LBlock. We get a 16-round integral distinguisher on TWINE which is the same with the previous best results [27]. We omit the details for TWINE due to the limit of space.

6 Conclusions

In this paper, we solve the complexity problem of searching integral distinguishers based on BDPT. We research the BDPT of S-box for the first time and propose an algorithm which can get the BDPT division trails from the ANF of S-box directly. Using the pruning techniques, we can removing redundant vectors which has no impact on the m-th bit's balance based on BDPT. It makes the propagation of BDPT efficient. Inspired by the "lazy propagation" which can use CBDP method to show some bits are unknown based on BDPT, we propose the notion of "fast propagation" which show some bits are balanced. Finally we propose an algorithm to estimate whether the m-th output bit is balanced or not based on BDPT.

We apply our searching algorithm to SIMON, SIMECK, PRESENT, RECTANGLE, LBlock, and TWINE. For SIMON64, PRESENT, and RECTANGLE, we find more balanced bits than the previous longest distinguishers. For LBlock, we find a 17-round integral distinguisher which is one more round than the previous longest integral distinguisher, and a better 16-round integral distinguisher with less active bits can be obtain. For other ciphers, all the integral distinguishers we get are in accordance with the existing best integral distinguishers.

It shows that BDPT is powerful in finding integral distinguishers for block ciphers, and our algorithm can search integral distinguishers based on BDPT in practical time for block ciphers with block size larger than 32, which is impractical under the traditional framework.

The absence of integral characteristic based on BDPT doesn't imply the absence of integral characteristic. The precondition for the propagation rules of BDPT regards $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{\pi}_{\boldsymbol{u}_1}(\boldsymbol{x}) \oplus \boldsymbol{\pi}_{\boldsymbol{u}_2}(\boldsymbol{x}))$ as unknown if either $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{u}_1}(\boldsymbol{x})$ or $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi}_{\boldsymbol{u}_2}(\boldsymbol{x})$ is unknown. Any improvement on the accuracy of BDPT propagation may also obtain better integral distinguishers. Moreover, our searching algorithm supposes that all round keys are randomly and secretly chosen. If consider the key scheduling algorithm, we may obtain better integral distinguishers.

References

1. 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.
2. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., 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). doi: 10.1007/978-3-540-74735-2_31
3. 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). doi: 10.1007/978-3-662-53018-4_24
4. Daemen, J., Knudsen, L., Rijmen, V.: The Block Cipher Square. In: Biham, E. (eds.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997). doi: 10.1007/BFb0052343
5. Funabiki, Y., Todo, Y., Isobe, T., Morii, M.: Improved Integral Attack on HIGHT. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 363-383. Springer, Cham (2017). doi: 10.1007/978-3-319-60055-0_19
6. Gurobi: http://www.gurobi.com/
7. Knudsen, L., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). doi: 10.1007/3-540-45661-9_9
8. Li, P., Sun, B., Li, C.: Integral Cryptanalysis of ARIA. In: Bao, F., Yung, M., Lin, D.D., Jing, J.W. (eds.) INSCRYPT 2009. LNCS, vol 6151, pp. 1–14. Springer, Heidelberg (2009). doi: 10.1109/TDC.2010.5484388
9. Li, Y.J., Wu, W.L.: Improved Integral Attacks on Rijndael. Journal of Information Science and Engineering. 27, 2031–2045 (2011).
10. Matsui, M.: New block encryption algorithm MISTY. In: Biham, E. (eds.) FSE 1997. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (1997). doi: 10.1007/BFb0052334
11. Sage: http://www.sagemath.org/
12. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE : A lightweight block cipher for multiple platforms. In: Knudsen L.R., Wu H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2012). doi: 10.1007/978-3-642-35999-6_22

13. Sun, L., Wang, W., Wang, M.: Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. IACR Cryptology ePrint Archive, 2016: 811. https://eprint.iacr.org/2016/811.pdf
14. Sun, L., Wang, W., Wang, M.: Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol 10624, pp. 128–157. Springer, Cham (2017). doi: 10.1007/978-3-319-70694-8_5 Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property
15. Sun L., Wang M.: Towards a further understanding of bit-based division property. Science China Information Sciences. **60**(12), 128101 (2017)
16. 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). doi: 10.1007/978-3-662-45611-8_9
17. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015). doi: 10.1007/978-3-662-47989-6_20
18. 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). doi: 10.1007/978-3-662-52993-5_18
19. 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). doi: 10.1007/978-3-662-46800-5_12
20. 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). doi: 10.1007/978-3-642-21554-4_19
21. 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). doi: 10.1007/978-3-319-13039-2_9
22. 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). doi: 10.1007/978-3-662-53887-6_24
23. Xie X.F., Tian T.: Improved Distinguisher Search Techniques Based on Parity Sets. Sci China Inf Sci (2018). doi: 10.1007/s11432-018-9495-x
24. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., 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)
25. 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). doi: 10.1007/s11432-015-5459-7
26. Zhang, W., Rijmen V.: Division Cryptanalysis of Block Ciphers with a Binary Diffusion Layer. IACR Cryptology ePrint Archive 2017, 188 (2017). https://eprint.iacr.org/2017/188.pdf
27. Zhang, H., Wu, W.: Structural evaluation for generalized feistel structures and applications to LBlock and TWINE. In: Biryukov, A., Goyal, V. (eds.) INDOCRYPT 2015. LNCS, vol. 9462, pp. 218–237. Springer, Cham (2015). doi: 10.1007/978-3-319-26617-6_12

28. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-Pattern Based Integral Attack. In: Nyberg, K. (eds.) FSE 2008. LNCS, vol. 2086, pp. 363–381. Springer, Heidelberg (2008). doi: 10.1007/978-3-540-71039-4_23

A The Propagation of Division Property

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

Input $\mathcal{D}^{1^4}_{\mathbb{K},\{\ell\}}$	Output $\mathcal{D}^{1^4}_{\mathbb{K}',\mathbb{L}'}$
$\boldsymbol{\ell} = [0,0,0,0]$	$\mathbb{L}' = \{[0,0,0,0]\}$
$\boldsymbol{\ell} = [1,0,0,0]$	$\mathbb{L}' = \{[1,0,0,0],[0,0,0,1]\}$
$\boldsymbol{\ell} = [0,1,0,0]$	$\mathbb{L}' = \{[0,1,0,0],[0,0,0,1]\}$
$\boldsymbol{\ell} = [1,1,0,0]$	$\mathbb{L}' = \{[1,1,0,0],[0,0,0,1]\}$
$\boldsymbol{\ell} = [0,0,1,0]$	$\mathbb{L}' = \{[0,0,1,0],[0,0,0,1]\}$
$\boldsymbol{\ell} = [1,0,1,0]$	$\mathbb{L}' = \{[1,0,1,0],[1,0,0,1],[0,0,1,1]\}$
$\boldsymbol{\ell} = [0,1,1,0]$	$\mathbb{L}' = \{[0,1,1,0],[0,0,1,1],[0,1,0,1]\}$
$\boldsymbol{\ell} = [1,1,1,0]$	$\mathbb{L}' = \{[1,1,1,0],[0,0,1,1],[1,1,0,1]\}$
$\boldsymbol{\ell} = [k_3,k_2,k_1,k_0]$	$\mathbb{L}' = \{[k_3,k_2,k_1,k_0]\}$

Table 8. The $\mathbb{L}$ propagation of BDPT for the core operation of SIMON

Input $\mathcal{D}^{1^4}_{\mathbb{K},\{\ell\}}$	Output $\mathcal{D}^{1^4}_{\mathbb{K}',\mathbb{L}'}$
$\boldsymbol{\ell} = [0,0,0,0]$	$\mathbb{L}' = \{[0,0,0,0]\}$
$\boldsymbol{\ell} = [1,0,0,0]$	$\mathbb{L}' = \{[1,0,0,0]\}$
$\boldsymbol{\ell} = [0,1,0,0]$	$\mathbb{L}' = \{[0,1,0,0]\}$
$\boldsymbol{\ell} = [1,1,0,0]$	$\mathbb{L}' = \{[1,1,0,0],[0,0,0,1],[1,0,0,1],[0,1,0,1],[1,1,0,1]\}$
$\boldsymbol{\ell} = [0,0,1,0]$	$\mathbb{L}' = \{[0,0,1,0],[0,0,0,1],[0,0,1,1]\}$
$\boldsymbol{\ell} = [1,0,1,0]$	$\mathbb{L}' = \{[1,0,1,0],[1,0,0,1],[1,0,1,1]\}$
$\boldsymbol{\ell} = [0,1,1,0]$	$\mathbb{L}' = \{[0,1,1,0],[0,1,0,1],[0,1,1,1]\}$
$\boldsymbol{\ell} = [1,1,1,0]$	$\mathbb{L}' = \{[1,1,1,0],[0,0,1,1],[1,0,1,1],[0,1,1,1],[1,1,0,1]\}$
$\boldsymbol{\ell} = [\ell_3,\ell_2,\ell_1,\ell_0]$	$\mathbb{L}' = \{[\ell_3,\ell_2,\ell_1,\ell_0]\}$

Table 9. The $\mathbb{L}$ propagation property of S-box in PRESENT

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

Table 10. The $\mathbb{L}$ propagation property of S-box in RECTANGLE

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