IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. 3, pp. 288–304. DOI:10.13154/tosc.v2020.i3.288-304

Exploring Secret Keys in Searching Integral Distinguishers Based on Division Property

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

PLA SSF Information Engineering University, Zhengzhou, China. wsp2110@126.com, hb2110@126.com, guanjie007@163.com, zhkai2010@139.com, strwanzi@163.com

Abstract. Division property proposed by Todo at EUROCRYPT 2015 is a generalized integral property. Then, conventional bit-based division property (CBDP) and bit-based division property using three subsets (BDPT) were proposed by Todo and Morii at FSE 2016. At ASIACRYPT 2016, Xiang *et al.* extended Mixed Integer Linear Programming (MILP) method to search integral distinguishers based on CBDP. And at ASIACRYPT 2019, Wang *et al.* proposed an MILP-aided method of searching integral distinguishers based on BDPT. Although BDPT is powerful in searching integral distinguishers, the accuracy is not perfect.

For block cipher SPECK32, as the block size is only 32 bits, we can experimentally observe the behaviors of all the plaintexts under a fixed key. By testing 2^{10} random secret keys, we experimentally find a better integral distinguisher of 6-round SPECK32 with 30 active bits. But this experimental integral distinguisher cannot be proved by existing methods. So there still exists a gap between the proved distinguisher and the experimental one.

To fill the gap, we explore secret keys in searching integral distinguishers based on BDPT. We put forward a situation where "Xor with The Secret Key" operation can be bypassed. Based on the new BDPT propagation rule, an improved automatic algorithm of searching integral distinguishers is proposed. For SPECK32, our improved algorithm can find the 6-round integral distinguisher with 2^{30} chosen plaintexts. The gap between the proved distinguisher and the experimental one is filled. Moreover, we apply this improved method to search the integral distinguishers of SPECK, KATAN/KTANTAN, SIMON, SIMECK, SIMON(102), PRESENT and RECTANGLE block ciphers. The integral distinguishers found by our improved method are better than or consistent with the previous longest distinguishers.

Keywords: Integral Distinguisher · Division Property · MILP · Block Cipher

1 Introduction

Integral attack proposed by Knudsen and Wagner at FSE 2002 [KW02] is one of the most powerful tools used for block ciphers. It is extended from square attack [DKR97] and has been applied to many block ciphers so far, such as Rijndael [LW11], ARIA [LSL09] and Serpent [ZRHD08]. The basic idea of integral attack is to analyze the properties of corresponding ciphertexts, such as the zero-sum property.

Division property, a generalization of integral property [KW02], was proposed by Todo at EUROCRYPT 2015 [Tod15b]. It can exploit the algebraic structure of block ciphers to construct integral distinguishers even if the block ciphers have non-bijective, bit-oriented, or low-degree structures. A remarkable application is that, at CRYPTO 2015 [Tod15a], Todo applied division property to MISTY1 and achieved the first theoretical cryptanalysis of the full-round MISTY1. Then, Sun *et al.* [SHZ+17] revisited division property and studied the property of a multiset satisfying certain division property. And at CRYPTO

Received: 2020-03-02 Revised: 2020-06-01 Accepted: 2020-08-01 Published: 2020-09-28

2016, Boura and Canteaut [BC16] introduced a new notion called *parity set* to exploit division property. They formulated and characterized the division property of S-box.

In order to exploit the concrete structure of round functions, Todo and Morii [TM16] proposed the notion of bit-based division property at FSE 2016. There are two kinds of bit-based division properties: conventional bit-based division property (CBDP) and bit-based division property using three subsets (BDPT). CBDP focuses on that $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$ is 0 or unknown, while BDPT focuses on that $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$ is 0, 1, or unknown. Although CBDP and BDPT could find accurate integral distinguishers, the huge complexity once restricted their wide applications. At ASIACRYPT 2016, Xiang *et al.* [XZBL16] proposed the concept of *CBDP trail* and applied MILP method to search integral distinguishers based on CBDP, which allowed them to analyze block ciphers with large sizes. BDPT can find more accurate integral distinguishers than CBDP. For example, CBDP cannot prove the existence of SIMON32's 15-round integral distinguisher which is obtained by BDPT [TM16]. But the MILP modeling technique cannot solve the propagation of BDPT directly. In order to solve this problem, an automatic search for a variant BDPT with STP solver was proposed in [HW19]. But the variant BDPT sacrifices some accuracy of the original BDPT. Later, at ASIACRYPT 2019, Wang *et al.* [WHG+19] proposed the pruning properties of BDPT and modeled the propagation of BDPT accurately.

Cube attack, proposed by Dinur and Shamir [DS09] at EUROCRYPT 2009, is one of the most powerful cryptanalytic techniques against symmetric cryptosystems. The target of cube attack is to recover secret variables from the simplified polynomial called *superpoly.* Because traditional cube attack [DS09] regarded a cipher as black polynomial and introduced a linearity test to recover superpoly, the cube had to be limited in practical size. It is noticeable that, at CRYPTO 2017, Todo *et al.* treated the polynomial as non-blackbox and applied CBDP to cube attacks on stream ciphers for the first time. Due to the MILP-aided CBDP, they could evaluate the algebraic normal form of the superpoly with large cube size. Then, at CRYPTO 2018, Wang [WHT+18] *et al.* improved it by introducing flag and term enumeration techniques. For CBDP based cube attacks, the superpolies of large cubes can be recovered by theoretical method. But the theory of CBDP cannot ensure that the superpoly of a cube is non-constant. Hence the key recovery attack may be just a distinguish attack. To solve this problem, at ASIACRYPT 2019, Wang *et al.* [WHG+19] proposed the cube attack based on BDPT and proved that BDPT without unknown subset can recover the accurate superpoly of cube attack. Then, at EUROCRYPT 2020, Hao *et al.* [HLM+20] proposed a new modeling method for the BDPT without unknown subset. Their algorithm is more efficient, and it can improve existing key-recovery attacks on many ciphers.

Although BDPT is powerful in searching integral distinguishers and cube attacks, the accuracy is not perfect. As pointed in [TM16], the propagation of BDPT simply regards $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{x}^{\boldsymbol{u}_1} \oplus \boldsymbol{x}^{\boldsymbol{u}_2})$ as unknown if either $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_1}$ or $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_2}$ is unknown. That is, the bit-based division property cannot exploit the following fact that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{x}^{\boldsymbol{u}_1} \oplus \boldsymbol{x}^{\boldsymbol{u}_2})$ may be always 0 or 1 although $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_1}$ and $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_2}$ are unknown. How to improve the accuracy of BDPT is an important issue worth studying.

1.1 Motivation

SPECK [BSS+13] is a family of lightweight block ciphers. And according to block size, SPECK family of ciphers are composed of SPECK32/48/64/96/128. For SPECK32, because the block size is only 32 bits, we can observe the behaviors of all the plaintexts under a fixed key. By testing 2^{10} random secret keys, we experimentally find a better integral distinguisher of 6-round SPECK32 with 30 active bits. However, this experimental integral distinguisher cannot be proved by existing methods. Namely, either the above experimental 6-round distinguisher does not work for all keys or the existing methods cannot find accurate integral distinguishers. Therefore, we want to give an improved searching method to tackle this problem.

1.2 Our Contributions

1.2.1 Bypass the Influence of Some Secret Keys on BDPT Propagation

We show that there are integral properties which are beyond BDPT. The main reason is that the unknown parity may be always 0 or 1 due to the cancellation of terms which are unknown. According to the previous propagation rule of BDPT through "Xor with The Secret Key" operation, some vectors may be put into the set $\mathbb{K}$ which represents unknown. We put forward a situation where "Xor with The Secret Key" operation can be bypassed. That is, no vectors will be put into $\mathbb{K}$. Based on the new propagation rule, an improved automatic algorithm of searching integral distinguishers is proposed.

1.2.2 Applications

For SPECK32, our new method shows that the experimental integral distinguisher with 2^{30} chosen plaintexts works for all keys. The gap between the proved distinguisher and the experimental one is filled. For SPECK48/64/96, an additional integral distinguisher which has the same length with the previous longest integral distinguishers is found, respectively.

KATAN and KTANTAN [CDK09] are two families of hardware oriented ciphers. For KATAN/KTANTAN32/48, the integral distinguishers found by us are in accordance with the previous longest distinguishers [HW19]. For KATAN/KTANTAN64, we find a 73.6-round integral distinguisher which is longer than that in [HW19].

SIMON, SIMECK and SIMON(102) are lightweight block ciphers which involve only bit-wise And, Xor, and Circular shift operations. For SIMON and SIMECK family of block ciphers, the integral distinguishers found by our method are in accordance with the previous longest distinguishers. And for SIMON(102), we determine the accurate values of constant bits in integral distinguishers.

PRESENT and RECTANGLE are two lightweight block ciphers with SPN structure. The integral distinguishers obtained by our new searching algorithm are in accordance with the previous longest integral distinguishers.

Our improved results are listed in Table 1.

2 Preliminaries

2.1 Notations

Let $\mathbb{F}_2$ denote the finite field $\{0, 1\}$ and $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_2^n$ be an n-bit vector, where a_i denotes the i-th bit of $\boldsymbol{a}$. For n-bit vectors $\boldsymbol{x}$ and $\boldsymbol{u}$, define $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=0}^{n-1} x_i^{u_i}$. Then, for any $\boldsymbol{k} \in \mathbb{F}_2^n$ and $\boldsymbol{k'} \in \mathbb{F}_2^n$, define $\boldsymbol{k} \succeq \boldsymbol{k'}$ if $k_i \geq k'_i$ holds for all $i = 0, 1, \ldots, n-1$, and define $\boldsymbol{k} \succ \boldsymbol{k'}$ if $k_i > k'_i$ holds for all $i = 0, 1, \ldots, n-1$. For a subset $I \subset \{0, 1, \ldots, n-1\}$, $\boldsymbol{u}_I$ denotes an n-dimensional bit vector $(u_0, u_1, \ldots, u_{n-1})$ satisfying $u_i = 1$ if $i \in I$ and $u_i = 0$ otherwise. We simply write $\mathbb{K} \leftarrow \boldsymbol{k}$ when $\mathbb{K} = \mathbb{K} \cup \{\boldsymbol{k}\}$ and $\mathbb{K} \rightarrow \boldsymbol{k}$ when $\mathbb{K} = \mathbb{K} \setminus \{\boldsymbol{k}\}$. And $|\mathbb{K}|$ denotes the number of elements in the set $\mathbb{K}$.

2.2 Mixed Integer Linear Programming

MILP is a kind of optimization or feasibility program whose objective function and constraints are linear, and the variables can be continuous or integers. Generally, an

Table 1: Summarization of improved integral distinguishers

Cipher	Data	Round⋆	Number of constant bits	Number of distinguisher	Time	Reference
SPECK32	2^{31}	7	1	2	3.5m	[HW19, SWW17]
	$\mathbf{2^{30}}$	7	1	1	1h5m	Sect. 4.1
SPECK48	2^{45}	7	1	1	<6m	[SWW17]
	2^{45}	7	1	**2**	12h57m	Sect. 4.1
SPECK64	2^{61}	7	1	1	<6m	[SWW17]
	2^{61}	7	1	**2**	17h40m	Sect. 4.1
SPECK96	2^{93}	7	1	1	<6m	[SWW17]
	2^{93}	7	1	**2**	7d10h25m	Sect. 4.1
KATAN/KTANTAN64	2^{63}	72.3	2	1	18m	[HW19]
		73.6	1	1	44h24m	Sect. 4.2
SIMON(102)32	2^{31}	20	3	32	25s	[HW19]
		20	**3**‡	32	22m	Sect. 4.3
SIMON(102)48	2^{47}	28	3	48	9.3s	[HW19]
		28	**3**‡	48	1h10m	Sect. 4.3
SIMON(102)64	2^{63}	36	3	64	1.1h	[HW19]
		36	**3**‡	64	3h27m	Sect. 4.3

⋆ Generally, the input multiset of BDPT should satisfy the condition that some bits are active and the others are constant. For SPECK, SIMON, SIMECK, SIMON(102) family of block ciphers, since the round keys are xored into the state after the round functions, we can add one more round before the distinguishers using the technique in [WLV+14]. And these extended integral distinguishers cannot be found by our method directly. The results of SPECK and SIMON(102) presented in the third column of Table 1 have been added by one round.

‡ For SIMON(102), the accurate values of some constant bits are unknown in previous integral distinguishers. We determine the accurate values of constant bits.

MILP model $\mathcal{M}$ consists of variables $\mathcal{M}.var$, constrains $\mathcal{M}.con$, and the objective function $\mathcal{M}.obj$. MILP models can be solved by solver like Gurobi [GRB]. If there is no feasible solution, the solver will returns *infeasible*. And if there are feasible solutions, the solver will returns the optimal value of the objective function. When there is no objective function in $\mathcal{M}$, the MILP solver will only return whether $\mathcal{M}$ is feasible or not.

2.3 Bit-based Division Property

Two kinds of bit-based division property (CBDP and BDPT) were introduced by Todo and Morii at FSE 2016 [TM16]. Their definitions are as follows.

Definition 1. (CBDP [TM16]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the CBDP $\mathcal{D}_{\mathbb{K}}^{1^n}$, where $\mathbb{K}$ denotes a set of n-dimensional bit vectors, it fulfills the following conditions:*

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

Definition 2. (BDPT [TM16]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ and $\mathbb{L}$ be two sets whose elements take n-dimensional bit vectors. When the multiset $\mathbb{X}$ has the BDPT $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, it fulfills the following conditions:*

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

According to [TM16], if there are $\boldsymbol{k} \in \mathbb{K}$ and $\boldsymbol{k}' \in \mathbb{K}$ satisfying $\boldsymbol{k} \succeq \boldsymbol{k}'$, $\boldsymbol{k}$ can be removed from $\mathbb{K}$ because the vector $\boldsymbol{k}$ is redundant. This progress is denoted as ***Reduce0***$(\mathbb{K})$. And if there are $\boldsymbol{\ell} \in \mathbb{L}$ and $\boldsymbol{k} \in \mathbb{K}$ satisfying $\boldsymbol{\ell} \succeq \boldsymbol{k}$, the vector $\boldsymbol{\ell}$ can also be removed from $\mathbb{L}$. This progress is denoted as ***Reduce1***$(\mathbb{K}, \mathbb{L})$. For any $\boldsymbol{u}$, the redundant vectors in $\mathbb{K}$ and $\mathbb{L}$ will not affect the value of $\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}}$.

The propagation rules of $\mathbb{K}$ in CBDP are the same with BDPT. So we only introduce the propagation rules of BDPT which are needed in this paper.

BDPT Rule 1 (Xor with The Secret Key [TM16]). *Let $\mathbb{X}$ be the input multiset satisfying $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$. For the input $\boldsymbol{x} \in \mathbb{X}$, the output $\boldsymbol{y} \in \mathbb{Y}$ is $\boldsymbol{y} = (x_0, \ldots, x_{i-1}, x_i \oplus r_k, x_{i+1}, \ldots, x_{n-1})$, where r_k is the secret key. Then, the output multiset $\mathbb{Y}$ has $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$, where $\mathbb{K}'$ and $\mathbb{L}'$ are computed as*

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

BDPT Rule 2 (S-box [WHG+19]). *For an S-box: $\mathbb{F}_2^n \to \mathbb{F}_2^m$, let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{m-1})$ denote the input and output variables. And every $y_i, i \in \{0, 1, \ldots, m-1\}$ can be expressed as a boolean function of $(x_0, x_1, \ldots, x_{n-1})$. If the input BDPT of S-box is $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}=\{\boldsymbol{\ell}\}}$, then the output BDPT of S-box can be calculated by $\mathcal{D}^{1^m}_{\boldsymbol{Reduce0}(\underline{\mathbb{K}}),\boldsymbol{Reduce1}(\underline{\mathbb{K}},\underline{\mathbb{L}})}$, where*

$$\begin{aligned}
&\underline{\mathbb{K}} = \{\boldsymbol{u}' \in \mathbb{F}_2^m | \text{ for any } \boldsymbol{u} \in \mathbb{K}, \text{ if } \boldsymbol{y}^{\boldsymbol{u}'} \text{ contains any term } \boldsymbol{x}^{\boldsymbol{v}} \text{satisfying } \boldsymbol{v} \succeq \boldsymbol{u}\},\\
&\underline{\mathbb{L}} = \{\boldsymbol{u} \in \mathbb{F}_2^m | \boldsymbol{y}^{\boldsymbol{u}} \text{ contains the term } \boldsymbol{x}^{\boldsymbol{\ell}}\}.
\end{aligned}$$

2.4 The MILP Model for CBDP

At ASIACRYPT 2016, Xiang *et al.* [XZBL16] applied MILP method to search integral distinguishers based on CBDP, which allowed them to analyze block ciphers with large sizes. Firstly, they introduced the concept of CBDP trail as follows.

Definition 3. (CBDP Trail [XZBL16]). *When considering the propagation of the CBDP $\{\boldsymbol{k}_I\} \overset{def}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \cdots \to \mathbb{K}_r$, for any vector $\boldsymbol{k}_{i+1} \in \mathbb{K}_{i+1}$, there must exist a vector $\boldsymbol{k}_i \in \mathbb{K}_i$ such that $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ by the propagation rules of CBDP. For $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$, if $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for all $i \in \{0, 1, \ldots r-1\}$, then $\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_r$ is called an r-round CBDP trail.*

Let E be a block cipher of size n. For indices set $I = \{i_0, \ldots, i_{|I|-1}\} \subset \{0, \ldots, n-1\}$, we can prepare $2^{|I|}$ chosen plaintexts where variables indexed by I are taking all possible combinations of values and the other variables are set to constants. The CBDP of such chosen plaintexts is $\mathcal{D}^{1^n}_{\mathbb{K}_0=\{\boldsymbol{k}_I\}}$. Let $\mathcal{L}$ denote a linear inequality system whose feasible solutions are all CBDP trails which start from the given CBDP $\mathcal{D}^{1^n}_{\mathbb{K}_0=\{\boldsymbol{k}_I\}}$. If the CBDP trail $\boldsymbol{k}_I \xrightarrow{E} \boldsymbol{e}_m$ is not the solution of $\mathcal{L}$, then the m-th output bit of cipher E is balanced. And if all the CBDP trails $\boldsymbol{k}_I \xrightarrow{E} \boldsymbol{e}_m, m \in \{0, 1, \ldots, n-1\}$ are the solutions of $\mathcal{L}$, the CBDP propagation should stop and there is no integral distinguisher.

The algorithm of searching integral distinguishers based on CBDP is denoted as $CBDP\big(E, \mathcal{D}^{1^n}_{\mathbb{K}_0}, m\big)$, where E is a cipher, $\mathcal{D}^{1^n}_{\mathbb{K}_0}$ is the input CBDP of E and m is an integer. The algorithm $CBDP\big(E, \mathcal{D}^{1^n}_{\mathbb{K}_0}, m\big)$ will return unknown or 0. That means the sum of the m-th output bit of E is unknown or 0, correspondingly. For more details, please refer to [XZBL16].

2.5 The MILP-aided Method of Searching BDPT

Recently, an MILP-aided method of searching integral distinguishers based on BDPT was proposed in [WHG+19]. The main insights are the pruning techniques and stopping rules of BDPT as follows. For more information, please refer to [WHG+19].

Let $E(\boldsymbol{x}) = f_{l-1} \circ f_{l-2} \circ \cdots \circ f_0(\boldsymbol{x})$ be a cipher, where $\boldsymbol{x} \in \mathbb{F}_2^n$ is the input variables. For the initial multiset $\mathbb{X}$ satisfying $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$, let $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}$ be the input BDPT of $\overline{E_i} = f_{l-1} \circ \cdots \circ f_i$, where $0 \le i \le l-1$. And the output BDPT of E is denoted as $\mathcal{D}^{1^n}_{\mathbb{K}_l,\mathbb{L}_l}$.

Theorem 1. (Prune $\mathbb{K}$ [WHG+19]) *For any vector $\boldsymbol{k} \in \mathbb{K}_i$, if there is no CBDP trail such that $\boldsymbol{k} \xrightarrow{\overline{E_i}} \boldsymbol{e}_m$, the BDPT propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}$ is equivalent to that of $\mathcal{D}^{1^n}_{\mathbb{K}_i \to \boldsymbol{k},\mathbb{L}_i}$ on whether $\boldsymbol{e}_m \in \mathbb{K}_l$ and $\boldsymbol{e}_m \in \mathbb{L}_l$ or not.*

Theorem 2. (Prune $\mathbb{L}$ [WHG+19]) *For any vector $\boldsymbol{\ell} \in \mathbb{L}_i$, if there is no CBDP trail such that $\boldsymbol{\ell} \xrightarrow{\overline{E_i}} \boldsymbol{e}_m$, the BDPT propagation of $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}$ is equivalent to that of $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i \to \boldsymbol{\ell}}$ on whether $\boldsymbol{e}_m \in \mathbb{K}_l$ and $\boldsymbol{e}_m \in \mathbb{L}_l$ or not.*

Stopping Rule 1. [WHG+19] *For any vector $\boldsymbol{k} \in \mathbb{K}_i$, if there is CBDP trail such that $\boldsymbol{k} \xrightarrow{\overline{E_i}} \boldsymbol{e}_m$, we stop the process and obtain that the m-th output bit of E is unknown.*

Stopping Rule 2. [WHG+19] *After considering Stopping Rule 1, we use Theorem 2 to remove the redundant vectors in the set $\mathbb{L}_i$. If there is still vector $\boldsymbol{\ell} \in \mathbb{L}_i$, we cannot stop the procedure and $\boldsymbol{\ell}$ should be propagated to next part based on BDPT propagation rules. If there is no vector in $\mathbb{L}_i$, we can get that the m-th output bit of E is balanced.*

Stopping Rule 3. [WHG+19] *If the process does not stop even the output BDPT of E is got, namely, $\mathbb{K}_l = \emptyset$ and $\mathbb{L}_l = \{\boldsymbol{e}_m\}$. Then, we find an integral distinguisher whose sum of the m-th output bit is 1.*

The algorithm of searching integral distinguishers based on BDPT is denoted as $BDPT\big(E, \mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}, m\big)$, where E is a cipher, $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$ is the input BDPT of E and m is an integer. According to the above three stopping rules, the algorithm $BDPT\big(E, \mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}, m\big)$ will return unknown or 0 or 1. That means the sum of the m-th output bit of E is unknown or 0 or 1, correspondingly.

3 Exploring Secret Keys in the Propagation of BDPT

3.1 The Integral Property Which is Beyond BDPT

As pointed in [TM16], the propagation of BDPT simply regards $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{x}^{\boldsymbol{u}_1} \oplus \boldsymbol{x}^{\boldsymbol{u}_2})$ as unknown if either $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_1}$ or $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_2}$ is unknown. That is, BDPT cannot exploit the fact that the parity $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{x}^{\boldsymbol{u}_1} \oplus \boldsymbol{x}^{\boldsymbol{u}_2})$ may be always 0 or 1 although $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_1}$ and $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}_2}$ are unknown. We show it by giving a simple example.

Example 1. Let $F(\boldsymbol{x}, r_k) = f_1(f_0(\boldsymbol{x}, r_k))$ be a function, where $r_k \in \mathbb{F}_2$ is the secret key, $\boldsymbol{x} = (x_0, x_1, x_2, x_3) \in \mathbb{F}_2^4$ and

$$\begin{aligned} \boldsymbol{y} &= f_0(\boldsymbol{x}, r_k) = (x_0, x_1, x_2 \oplus r_k, x_3), \\ \boldsymbol{z} &= f_1(\boldsymbol{y}) = (y_0y_1y_2 \oplus y_0y_1y_2y_3, y_0, y_1, y_3). \end{aligned}$$

Assuming the input multiset $\mathbb{X}$ has BDPT $\mathcal{D}^{1^4}_{\mathbb{K}_0=\emptyset,\mathbb{L}_0=\{(1,1,0,1),(1,1,0,0)\}}$. Then, according to **BDPT Rule 1 (Xor with The Secret Key)**, the BDPT of $\mathbb{Y} = \{\boldsymbol{y} | \boldsymbol{y} = f_0(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{X}\}$

is $\mathcal{D}^{1^4}_{\mathbb{K}_1=\{(1,1,1,0)\},\mathbb{L}_1=\{(1,1,0,1),(1,1,0,0)\}}$. Let the BDPT of $\mathbb{Z} = \{\boldsymbol{z}|\boldsymbol{z} = f_1(\boldsymbol{y}), \boldsymbol{y} \in \mathbb{Y}\}$ be $\mathcal{D}^{1^4}_{\mathbb{K}_2,\mathbb{L}_2}$. We can view the function $f_1(\boldsymbol{y})$ as an S-box, then we know $(1,0,0,0) \in \mathbb{K}_2$ according to the **BDPT Rule 2 (S-box)**. That means $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} F(\boldsymbol{x}, r_k)^{(1,0,0,0)}$ is unknown. However, we can get the following equation:

$$\begin{aligned}\bigoplus_{\boldsymbol{x}\in\mathbb{X}} F(\boldsymbol{x}, r_k)^{(1,0,0,0)} &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} x_0x_1(x_2 \oplus r_k) \oplus x_0x_1(x_2 \oplus r_k)x_3 \\ &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} x_0x_1x_2 \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} x_0x_1x_2x_3 \oplus r_k \bigoplus_{\boldsymbol{x}\in\mathbb{X}} (x_0x_1 \oplus x_0x_1x_3) \\ &= 0 \oplus 0 \oplus r_k(1 \oplus 1) = 0.\end{aligned}$$

Therefore, $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} F(\boldsymbol{x}, r_k)^{(1,0,0,0)}$ is always 0 not unknown.

According to **Example 1**, the two unknown parities $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} x_0x_1r_k$ and $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} x_0x_1x_3r_k$ can be cancelled with each other. But the propagation rules of division property set $\mathbb{K}$ cannot exploring the above cancelling property. From the **BDPT Rule 1 (Xor with The Secret Key)**, the set $\mathbb{L}$ may generate some vectors which should be added into $\mathbb{K}$. Therefore, if we can prevent vectors from being added to the set $\mathbb{K}$, the accuracy of BDPT will be improved.

3.2 Bypass the Influence of Some Secret Keys on BDPT

Before researching the influence of secret keys on BDPT, we firstly propose a lemma.

Lemma 1. *Let $\mathbb{X}$ be the input multiset satisfying $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$. For the input $\boldsymbol{x} \in \mathbb{X}$, the output $\boldsymbol{y} \in \mathbb{Y}$ is computed as $\boldsymbol{y} = (x_0, \ldots, x_{i-1}, x_i \oplus 0, x_{i+1}, \ldots, x_{n-1}) = \boldsymbol{x}$ and the output $\boldsymbol{z} \in \mathbb{Z}$ is computed as $\boldsymbol{z} = (x_0, \ldots, x_{i-1}, x_i \oplus 1, x_{i+1}, \ldots, x_{n-1}) = \boldsymbol{x} \oplus \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the unit vector whose only i-th bit is 1. Then, the multiset $\mathbb{Y}\bigcup\mathbb{Z}$ has BDPT $\mathcal{D}^{1^n}_{\mathbb{K}'=\mathbb{K},\mathbb{L}'=\underline{\mathbb{L}}}$, where*

$$\underline{\mathbb{L}} = \{(\ell_0, \ell_1, \ldots, \ell_i \vee 1, \ldots, \ell_{n-1}) \,|\, \boldsymbol{\ell} \in \mathbb{L} \text{ satisfying } \ell_i = 0\}.$$

Proof. For any $\boldsymbol{u} \in \mathbb{F}_2^n$, let $\underline{\boldsymbol{u}} = (u_0, \ldots, u_{i-1}, 0, u_{i+1}, \ldots, u_{n-1})$. Then, we have

$$\bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \boldsymbol{a}^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{y}\in\mathbb{Y}} \boldsymbol{y}^{\boldsymbol{u}} \oplus \bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \boldsymbol{z}^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} (\boldsymbol{x} \oplus \boldsymbol{e}_i)^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} x_i^{u_i} \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} (x_i \oplus e_i)^{u_i}.$$

When $u_i = 0$, we have

$$\bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \boldsymbol{a}^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} x_i^{u_i} \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} (x_i \oplus e_i)^{u_i} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} = 0. \tag{1}$$

Therefore, the output BDPT of the multiset $\mathbb{Y}\bigcup\mathbb{Z}$ can be obtained by the vectors $\boldsymbol{u} \in \mathbb{F}_2^n$ satisfying $u_i = 1$, denoted as $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$. When $u_i = 1$, we have

$$\bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \boldsymbol{a}^{\boldsymbol{u}} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} x_i^{u_i} \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} (x_i \oplus e_i)^{u_i} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} (x_i \oplus x_i \oplus e_i) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}}.$$

Because the multiset $\mathbb{X}$ has BDPT $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$ and $\boldsymbol{u} \succeq \underline{\boldsymbol{u}}$. Then, for any $\boldsymbol{u} \in \mathbb{F}_2^n$, if there is no $\boldsymbol{k} \in \mathbb{K}$ satisfying $\boldsymbol{u} \succeq \boldsymbol{k}$, we have $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} =$ constant (0 or 1) and $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\underline{\boldsymbol{u}}} =$ constant (0 or 1). According to the definition of BDPT, the multiset $\mathbb{Y}\bigcup\mathbb{Z}$ has $\mathcal{D}^{1^n}_{\mathbb{K}'=\mathbb{K},\mathbb{L}'}$.

For any $\underline{\boldsymbol{\ell}} \in \underline{\mathbb{L}}$, there exists an $\boldsymbol{\ell} \in \mathbb{L}$ satisfying $\underline{\boldsymbol{\ell}} = \boldsymbol{\ell} \oplus \boldsymbol{e}_i$. Since $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{\ell}} = 1$, we have

$$\begin{aligned}\bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \boldsymbol{a}^{\underline{\boldsymbol{\ell}}} &= \bigoplus_{\boldsymbol{y}\in\mathbb{Y}} \boldsymbol{y}^{\underline{\boldsymbol{\ell}}} \oplus \bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \boldsymbol{z}^{\underline{\boldsymbol{\ell}}} = \bigoplus_{\boldsymbol{y}\in\mathbb{Y}} \boldsymbol{y}^{\boldsymbol{\ell}\oplus\boldsymbol{e}_i} \oplus \bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \boldsymbol{z}^{\boldsymbol{\ell}\oplus\boldsymbol{e}_i} = \bigoplus_{\boldsymbol{y}\in\mathbb{Y}} \boldsymbol{y}^{\boldsymbol{\ell}} y_i \oplus \bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \boldsymbol{z}^{\boldsymbol{\ell}} z_i \\ &= \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{\ell}} x_i \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{\ell}} (x_i \oplus 1) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{\ell}} = 1.\end{aligned} \tag{2}$$

So $\underline{\boldsymbol{\ell}} \in \mathbb{L}'$, we have $\underline{\mathbb{L}} \subseteq \mathbb{L}'$.

For any $\boldsymbol{\ell}' \in \mathbb{L}'$, if $\ell'_i = 0$, according to Eq. (1), we have $\bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \boldsymbol{a}^{\boldsymbol{\ell}'} = 0$ which is contradict with $\boldsymbol{\ell}' \in \mathbb{L}'$. Therefore, every vector $\boldsymbol{\ell}' \in \mathbb{L}'$ meets the condition $\ell'_i = 1$. According to Eq. (2), we know that $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{x}^{\boldsymbol{\ell}} = 1$, where $\boldsymbol{\ell} = \boldsymbol{\ell}' \oplus \boldsymbol{e}_i$. Namely, there exists $\boldsymbol{\ell} \in \mathbb{L}$ satisfying $\ell_i = 0$ and $\boldsymbol{\ell}' = \boldsymbol{\ell} \oplus \boldsymbol{e}_i$. Therefore, $\mathbb{L}' \subseteq \underline{\mathbb{L}}$.

Altogether, we obtain $\mathbb{K}' = \mathbb{K}$ and $\mathbb{L}' = \underline{\mathbb{L}}$.

□

Then, based on Lemma 1, we proposed the bypass technique of secret keys in the propagation of BDPT.

Theorem 3. *Let $E(\boldsymbol{x}) = f_{l-1} \circ f_{l-2} \circ \cdots \circ f_0(\boldsymbol{x})$ be a cipher, where $\boldsymbol{x} \in \mathbb{F}_2^n$ is the input variables. If f_i is an "Xor with The Secret Key" function denoted as $\boldsymbol{z} = f_i(\boldsymbol{y}) = (y_0, \ldots, y_{j-1}, y_j \oplus r_k, y_{j+1} \ldots, y_{n-1})$, where r_k is the secret key. For the initial multiset $\mathbb{X}$ satisfying $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$, let $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}$ be the input BDPT of $\overline{E_i} = f_{l-1} \circ f_{l-2} \cdots \circ f_i$. Then, if the MILP-aided algorithm $BDPT\left(\overline{E_{i+1}}, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}'_i}, m\right)$ return 0, we have*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=0}(\boldsymbol{x})^{\boldsymbol{e}_m} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=1}(\boldsymbol{x})^{\boldsymbol{e}_m},$$

where $\mathbb{L}'_i = \{(\ell_0, \ell_1, \ldots, \ell_j \vee 1, \ldots, \ell_{n-1}) \,|\, \boldsymbol{\ell} \in \mathbb{L}_i \text{ satisfying } \ell_j = 0\}$, $\boldsymbol{e}_m$ is the unit vector whose only m-th bit is 1, $E_{r_k=0}(\boldsymbol{x})^{\boldsymbol{e}_m}$ is the m-th output bit of cipher $E(\boldsymbol{x})$ whose secret key bit r_k is fixed as 0, and $E_{r_k=1}(\boldsymbol{x})^{\boldsymbol{e}_m}$ is the m-th output bit of cipher $E(\boldsymbol{x})$ whose secret key bit r_k is fixed as 1.

Proof. The BDPT of multiset $\mathbb{W} = \{\boldsymbol{w} | \boldsymbol{w} = f_{i-1} \circ f_{i-2} \circ \cdots \circ f_0(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{X}\}$ is $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}$. Because f_i is an "Xor with The Secret Key" function. When $r_k = 0$, the output multiset of $f_i \circ f_{i-1} \circ \cdots \circ f_0(\boldsymbol{x})$ is $\mathbb{Y} = \{\boldsymbol{y} | \boldsymbol{y} = f_i(\boldsymbol{w}), \boldsymbol{w} \in \mathbb{W} \text{ and } r_k = 0\}$. And when $r_k = 1$, the output multiset of $f_i \circ f_{i-1} \circ \cdots \circ f_0(\boldsymbol{x})$ is $\mathbb{Z} = \{\boldsymbol{z} | \boldsymbol{z} = f_i(\boldsymbol{w}), \boldsymbol{w} \in \mathbb{W} \text{ and } r_k = 1\}$. According to Lemma 1, the BDPT of multiset $\mathbb{Y} \bigcup \mathbb{Z}$ is $\mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}'_i}$, where

$$\mathbb{L}'_i = \{(\ell_0, \ell_1, \ldots, \ell_j \vee 1, \ldots, \ell_{n-1}) \,|\, \boldsymbol{\ell} \in \mathbb{L}_i \text{ satisfying } \ell_j = 0\}.$$

From Section 2.5, when the MILP-aided algorithm $BDPT\left(\overline{E_{i+1}}, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}'_i}, m\right)$ return 0, it means $\bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \overline{E_{i+1}}(\boldsymbol{a})^{\boldsymbol{e}_m} = 0$. Then, we have

$$\begin{aligned}\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=0}(\boldsymbol{x})^{\boldsymbol{e}_m} \oplus \bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=1}(\boldsymbol{x})^{\boldsymbol{e}_m} &= \bigoplus_{\boldsymbol{y}\in\mathbb{Y}} \overline{E_{i+1}}(\boldsymbol{y})^{\boldsymbol{e}_m} \oplus \bigoplus_{\boldsymbol{z}\in\mathbb{Z}} \overline{E_{i+1}}(\boldsymbol{z})^{\boldsymbol{e}_m} \\ &= \bigoplus_{\boldsymbol{a}\in\mathbb{Y}\bigcup\mathbb{Z}} \overline{E_{i+1}}(\boldsymbol{a})^{\boldsymbol{e}_m} \\ &= 0.\end{aligned}$$

Therefore,

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=0}\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=1}\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m}.$$

□

Because the secret key r_k is a constant bit whose value is unknown. When the condition $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=0}\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=1}\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m}$ is satisfied, we have $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m} = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=0}\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m}$. Namely, if we want to research the integral property of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m}$, we only need to research the integral property of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} E_{r_k=0}\left(\boldsymbol{x}\right)^{\boldsymbol{e}_m}$. When $r_k = 0$, the "Xor with The Secret Key" function f_i will become the identity map and we do not need to consider the influence of the secret key bit r_k any more. That is, the influence of the secret key bit r_k can be bypassed.

In Section 3.1 we show the integral property which is beyond BDPT. In order to show that our bypass technique can find more accurate integral property than BDPT, we apply it to the same example.

Example 2. Let $F\left(\boldsymbol{x}, r_k\right) = f_1\left(f_0\left(\boldsymbol{x}, r_k\right)\right)$ be a function, where $r_k \in \mathbb{F}_2$ is the secret key, $\boldsymbol{x} = \left(x_0, x_1, x_2, x_3\right) \in \mathbb{F}_2^4$ and

$$\begin{aligned}\boldsymbol{y} &= f_0\left(\boldsymbol{x}, r_k\right) = \left(x_0, x_1, x_2 \oplus r_k, x_3\right),\\ \boldsymbol{z} &= f_1\left(\boldsymbol{y}\right) = \left(y_0y_1y_2 \oplus y_0y_1y_2y_3, y_0, y_1, y_3\right).\end{aligned}$$

Assuming the input multiset $\mathbb{X}$ has BDPT $\mathcal{D}^{1^4}_{\mathbb{K}_0=\emptyset,\mathbb{L}_0=\{(1,1,0,1),(1,1,0,0)\}}$. Because $\boldsymbol{y} = f_0\left(\boldsymbol{x}, r_k\right)$ is the "Xor with The Secret Key" function, we firstly research the MILP-aided algorithm $BDPT\left(f_1, \mathcal{D}^{1^4}_{\mathbb{K}_0,\mathbb{L}'_0}, 0\right)$, where $\mathbb{K} = \emptyset$ and $\mathbb{L}'_0 = \{(1,1,1,1), (1,1,1,0)\}$. The function $f_1\left(\boldsymbol{y}\right)$ can be seen as a 4-bit S-box. Then, according to **BDPT Rule 2 (S-box)**, when the input BDPT is $\mathcal{D}^{1^4}_{\mathbb{K}_0,\mathbb{L}'_0}$, the output BDPT of f_1 is $\mathcal{D}^{1^4}_{\emptyset,\emptyset}$. The **Stopping Rule 2** is triggered, and $BDPT\left(f_1, \mathcal{D}^{1^4}_{\mathbb{K}_0,\mathbb{L}'_0}, 0\right)$ will return 0. According to Theorem 3, we have

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} F\left(\boldsymbol{x}, r_k\right) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} f_1\left(f_0\left(\boldsymbol{x}, r_k\right)\right) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} f_1\left(f_0\left(\boldsymbol{x}, 0\right)\right) = \bigoplus_{\boldsymbol{x}\in\mathbb{X}} f_1\left(\boldsymbol{x}\right).$$

Because the input multiset $\mathbb{X}$ has BDPT $\mathcal{D}^{1^4}_{\mathbb{K}_0=\emptyset,\mathbb{L}_0=\{(1,1,0,1),(1,1,0,0)\}}$, according to **BDPT Rule 2 (S-box)**, the BDPT of $\{f_1\left(\boldsymbol{x}\right) | \boldsymbol{x} \in \mathbb{X}\}$ is $\mathcal{D}^{1^4}_{\emptyset,\{(0,1,1,1),(0,1,1,0)\}}$. From the definition of BDPT, we know that $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} F\left(\boldsymbol{x}, r_k\right)^{(1,0,0,0)} = 0$. As shown in **Example 1**, this is the integral property that cannot be found by BDPT.

3.3 A New Algorithm of Searching Integral Distinguishers

Our new algorithm of searching integral distinguishers should also have a given initial BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$. For an index subset $I = \{i_0, i_1, \ldots, i_{|I|-1}\} \subset \{0, 1, \ldots, n-1\}$, we prepare a multiset of $2^{|I|}$ chosen plaintexts, denoted as C_I, where variables indexed by I are taking all possible combinations of values and the other variables are set to constants. The BDPT of such chosen plaintexts is $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$, where $\mathbb{K}_0 = \{\boldsymbol{u} | \boldsymbol{u} \succ \boldsymbol{u}_I\}$ and $\mathbb{L}_0 = \{\boldsymbol{u}_I\}$. Moreover, if we know the values of constants, for index subsets $I = \{i_0, i_1, \ldots, i_{|I|-1}\} \subseteq \{0, 1, \ldots, n-1\}$, $J = \{j_0, j_1, \ldots, j_{|J|-1}\} \subseteq \{0, 1, \ldots, n-1\} \setminus I$ and $K = \{k_0, k_1, \ldots, k_{|K|-1}\} = \{0, 1, \ldots, n-1\} \setminus \{I \cup J\}$, we can prepare a multiset of $2^{|I|}$ chosen plaintexts, denoted as $C_{I,J,K}$, where $C_{I,J,K} = \{\boldsymbol{x} \in \mathbb{F}_2^n | x_i \in \mathbb{F}_2 \text{ if } i \in I, x_j = 1 \text{ if } j \in J, x_k = 0 \text{ if } k \in K\}$. Then, the BDPT of $C_{I,J,K}$ is $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$, where $\mathbb{K}_0 = \emptyset$ and $\mathbb{L}_0 = \{\boldsymbol{u} | \left(\boldsymbol{u}_I \oplus \boldsymbol{u}_J\right) \succeq \boldsymbol{u} \succeq \boldsymbol{u}_I\}$.

The paper [XZBL16] has shown the method of searching integral distinguishers based on CBDP, denoted as $CBDP\big(E_i, \mathcal{D}^{1^n}_{\mathbb{K}_0}, m\big)$. And the paper [WHG+19] has shown the method of building the MILP-aided method of searching integral distinguishers based on BDPT, denoted as $BDPT\big(E, \mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}, m\big)$. Moreover, according to **BDPT Rule 2 (S-box)**, the BDPT propagation of f_i can be obtained, denoted as $BDPTP\big(f_i, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}\big)$. Therefore, the whole framework of our new searching algorithm is shown in Algorithm 1.

Algorithm 1 $K\text{-}BDPT\big(E, \mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}, m\big)$

```
    Input: The cipher $E = f_{l-1} \circ \cdots \circ f_0(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{F}_2^n$ is the input variables.
           The input BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}$ and the integer number $m$.
    Output: The balanced information of the $m$-th output bit of $E$.
1   for $(i = 0; i \le l-1; i++)$ do
2       for $\boldsymbol{k}$ in $\mathbb{K}_i$ do
3           if $CBDP\Big(\overline{E_i}, \mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{k}\}}, m\Big)$ returns unknown then
4               return unknown
5           else
6               $\mathbb{K}_i \to \boldsymbol{k}$
7       end for
8       let $\mathbb{L}_i' = \emptyset$
9       for $\boldsymbol{\ell}$ in $\mathbb{L}_i$ do
10          if $CBDP\Big(\overline{E_i}, \mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{\ell}\}}, m\Big)$ returns unknown then
11              $\mathbb{L}_i' = \mathbb{L}_i' \cup \boldsymbol{\ell}$
12          end if
13      end for
14      let $\mathbb{L}_i = \mathbb{L}_i'$
15      if $\mathbb{L}_i = \emptyset$ then
16          return 0
17      end if
18      if $f_i$ is Xor with The Secret Key: $f_i(\boldsymbol{y}) = (y_0, \ldots, y_{j-1}, y_j \oplus r_k, \ldots, y_{n-1})$
19          let $\mathbb{L}_i' = \{(\ell_0, \ldots, \ell_j \vee 1, \ldots, \ell_{n-1}) \,|\, \boldsymbol{\ell} \in \mathbb{L}_i$ satisfying $\ell_j = 0\}$
20          if $BDPT\Big(\overline{E_{i+1}}, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i'}, m\Big)$ returns 0 then
21              $\mathbb{K}_{i+1} = \mathbb{K}_i$ and $\mathbb{L}_{i+1} = \mathbb{L}_i$
22          else
23              $\mathbb{K}_{i+1} = \mathbb{K}_i \cup \mathbb{L}_i'$ and $\mathbb{L}_{i+1} = \mathbb{L}_i$
24      else if $f_i$ does not involve secret keys then
25          $\mathcal{D}^{1^n}_{\mathbb{K}_{i+1},\mathbb{L}_{i+1}} = BDPTP\big(f_i, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}\big)$
26  end for
27  return 1
```

We explain Algorithm 1 line by line:

Line 1 The cipher E is divided into small parts.

Line 2-7 For any $\boldsymbol{k} \in \mathbb{K}_i$, if $CBDP\Big(\overline{E_i}, \mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{k}\}}, m\Big)$ returns unknown, according to **Stopping Rule 1**, we know that the m-th output bit is unknown. Otherwise, we remove it from $\mathbb{K}_i$ according to **Theorem 1 (Prune $\mathbb{K}$)**.

Line 8 Initialize $\mathbb{L}_i'$ to be an empty set.

Line 9-13 For any vector $\boldsymbol{\ell} \in \mathbb{L}_i$, if $CBDP\Big(\overline{E_i}, \mathcal{D}^{1^n}_{\mathbb{K}=\{\boldsymbol{\ell}\}}, m\Big)$ returns unknown, we store all these vectors in $\mathbb{L}_i'$.

Line 14 According to **Theorem 2 (Prune $\mathbb{L}$)**, let $\mathbb{L}_i = \mathbb{L}_i'$.

Line 15-17 If the set $\mathbb{L}_i$ is empty set, it satisfies **Stopping Rule 2**, that is, the m-th output bit is balanced. If we do not get the balanced information of the m-th bit, we should use the propagation rules of BDPT to get the input BDPT of the next part.

Line 18-23 When f_i is "Xor with The Secret Key", if $BDPT\left(\overline{E_{i+1}}, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}'_i}, m\right)$ returns 0, according to **Theorem 3**, the input BDPT of $\overline{E_{i+1}}$ is shown in **Line 21**. Otherwise, according to **BDPT Rule 1**, the input BDPT of $\overline{E_{i+1}}$ is shown in **Line 23**.

Line 24-25 When f_i is a function which does not involve secret keys, we can view it as an S-box and get its output BDPT according to **BDPT Rule 2 (S-box)**. This process is denoted as $\mathcal{D}^{1^n}_{\mathbb{K}_{i+1},\mathbb{L}_{i+1}} = BDPTP\left(f_i, \mathcal{D}^{1^n}_{\mathbb{K}_i,\mathbb{L}_i}\right)$.

Line 27 It triggers **Stopping Rules 3**, and the sum of the m-th output bit is 1.

It should be noted that Algorithm 1 can search the integral property of any output bit independently. Therefore, we can search the integral distinguishers of a block cipher parallelly.

Identify the Position Where the Inaccuracy of BDPT Occurs. When Algorithm 1 $K\text{-}BDPT\left(E, \mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}, m\right)$ returns "0" or "1", if $BDPT\left(E, \mathcal{D}^{1^n}_{\mathbb{K}_0,\mathbb{L}_0}, m\right)$ returns "unknown", we finds a better integral distinguisher which is beyond BDPT. Let $\boldsymbol{k} \in \mathbb{K}_i$ be the first vector that $CBDP\left(\overline{E_i}, \boldsymbol{k}, m\right)$ returns "unknown". According to **Theorem 2 (Prune $\mathbb{K}$)**, all the vectors in the set $\mathbb{K}_{i-1}$ will be removed. Therefore, the input BDPT of f_i satisfies $\mathcal{D}^{1^n}_{\mathbb{K}_{i-1}=\emptyset,\mathbb{L}_{i-1}}$. Then, the vector $\boldsymbol{k} \in \mathbb{K}_i$ can only be generated from $\mathbb{L}_{i-1}$ through "Xor with The Secret Key" function f_i. So the secret key bit in f_i is the position where the inaccuracy of BDPT occurs.

4 Applications to Block Ciphers

In this section, we apply our algorithm to SPECK, KATAN/KTANTAN, SIMON, SIMECK, SIMON(102), PRESENT and RECTANGLE block ciphers. All the experiments are conducted on the platform: Intel Core i5-9300 CPU @3.98GHz, 32.00G RAM. And the optimizer we used to solve MILP models is Gurobi 8.1.0 [GRB]. For the integral distinguishers, what needs to be explained is that 'a' denotes an active bit, 'c' denotes a constant bit, '?' denotes the bit whose balanced information is unknown, '0' denotes the bit whose sum is zero, and '1' denotes the bit whose sum is 1.

4.1 Application to SPECK

SPECK [BSS+13] is a family of lightweight block ciphers published by National Security Agency (NSA). It adopts ARX structure which takes the Modular Addition as its nonlinear operation. According to block size, SPECK family of ciphers are composed of SPECK2n, where $n \in \{16, 24, 32, 48, 64\}$. The round structure of SPECK is shown in Fig. 1, where $\left(x^i_{2n-1}, x^i_{2n-2}, \ldots, x^i_0\right)$ and $\left(x^{i+1}_{2n-1}, x^{i+1}_{2n-2}, \ldots, x^{i+1}_0\right)$ is the input and output of the i-th round function, $\left(k^i_{n-1}, k^i_{n-2}, \ldots, k^i_0\right)$ is the i-th round key, $\ggg a$ denotes right circular shift by a bits, $\lll b$ denotes left circular shift by b bits and $\boxplus$ represents the Modular Addition operation. The parameters a and b are 8 and 3, respectively, except in the case of SPECK32, where they are 7 and 2.

For SPECK32, we prepare chosen plaintexts such that the input variables x^0_5 and x^0_6 are constant and the others are active. Then, we apply Algorithm 1 to search the integral distinguisher of SPECK32 and identify a 6-round integral distinguisher. This integral distinguisher is completely the same with the experimental one which cannot be found by BDPT. Therefore, the gap between the proved distinguisher and the experimental one is filled. For SPECK48/64/96, the paper [SWW17] found only one 6-round integral distinguisher with $2^{45}/2^{61}/2^{93}$ chosen plaintexts, respectively. However, using Algorithm 1, we find 2 integral distinguishers for 6-round SPECK48/64/96 with $2^{45}/2^{61}/2^{93}$ chosen plaintexts, respectively. The detailed integral distinguishers are listed in Table 2. Moreover,

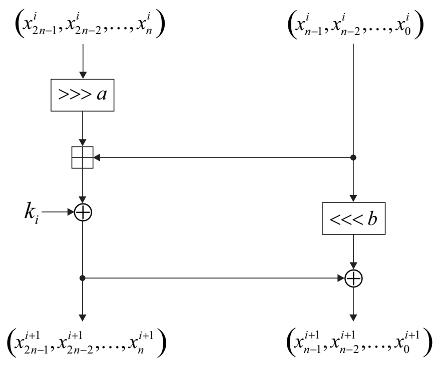
$(x^i_{2n-1}, x^i_{2n-2}, \ldots, x^i_n)$
$(x^i_{n-1}, x^i_{n-2}, \ldots, x^i_0)$
$>>> a$
k_i
$<<< b$
$(x^{i+1}_{2n-1}, x^{i+1}_{2n-2}, \ldots, x^{i+1}_n)$
$(x^{i+1}_{n-1}, x^{i+1}_{n-2}, \ldots, x^{i+1}_0)$

Figure 1: Round structure of SPECK

using the method in Section 3.3, we can give the positions where the inaccuracy of BDPT occurs. The detailed positions can be found in the third column of Table 2.

Table 2: Integral distinguishers of SPECK

Cipher	Distinguisher	Position
6-SPECK32	In: (aaaaaaaaaaaaaaaa, aaaaaaaaaccaaaaa) Out:(???????????????0, ????????????????)	k^0_7
6-SPECK48	In: (aaaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaacccaaaaa) Out:(???????????????????????0, ????????????????????????)	
	In: (aaaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaacaccaaaaa) Out:(???????????????????????0, ????????????????????????)	k^0_8
6-SPECK64	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaacccaaaaaa) Out:(???????????????????????????????0,????????????????????????????????)	
	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaacaccaaaaaa) Out:(???????????????????????????????0,????????????????????????????????)	k^0_8
6-SPECK96	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccaaaaa) Out:(???0, ??)	
	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacaccaaaaa) Out:(???0, ??)	k^0_8

4.2 Applications to KATAN and KTANTAN

KATAN and KTANTAN are two families of hardware oriented block ciphers with 32, 48 and 64-bit block size. All block ciphers share the 80-bit key size. And the only difference between KATAN and KTANTAN is the key schedule. They both take a very simple structure which is shown in Fig. 2, where IR is round constant, k_a and k_b are two subkey bits, L_1 and L_2 are LFSRs.

For KATAN/KTANTAN32, the registers L_1 and L_2 are shifted by 1 bit position, and the two new bits computed by two nonlinear functions are loaded in the least significant bits of L_1 and L_2. For KATAN/KTANTAN48, the two nonlinear functions and the update of the registers are applied twice with the same round subkey in each round, while the nonlinear functions and update of the registers are applied three times for KATAN/KTANTAN64. More details of KATAN/KTANTAN can be found in [CDK09].

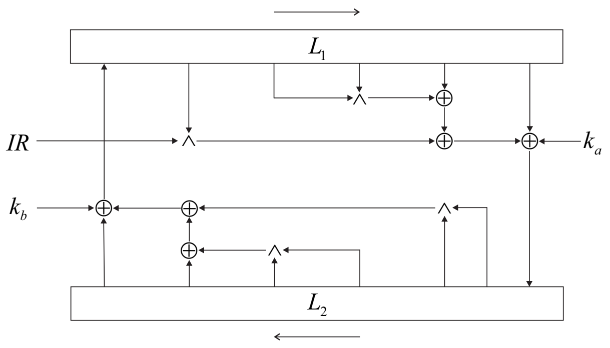

Figure 2: The structure of KATAN and KTANTAN

Applying Algorithm 1 to KATAN/KTANTAN, we can find a 101-round integral distinguisher of KATAN/KTANTAN32 and an 84-round integral distinguisher of KATAN/KTANTAN48 which are in accordance with the previous longest distinguishers [HW19]. For KATAN/KTANTAN64, we obtain a 73.6-round integral distinguisher which is longer than the previous longest integral distinguisher. The detailed forms of the integral distinguishers are listed in Table 3.

Table 3: Integral Distinguishers of KATAN/KTANTAN

Block size	Round	Distinguisher
32	101	In: (aaaaaaaaaaaaaaaaaacaaaaaaaaaaaaa) Out: (1???????????????????????????????)
48	84	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaacaaaaaaaaaaaaaaaaaaaa) Out: (0???)
64	73.6	In: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacaaaaaaaaaaaaaaaaaaaaaaaa) Out: (0???)

4.3 Applications to SIMON, SIMECK and SIMON(102)

SIMON is a lightweight block cipher family [BSS+13] based on Feistel structure which only involves bit-wise And, Xor and Circular shift operations. Let SIMON$2n$ be the SIMON cipher with $2n$-bit block length, where $n \in \{16, 24, 32, 48, 64\}$. And the left part of Fig.3 shows the round structure of SIMON$2n$. The core operation of round function is represented by the right part of Fig. 3.

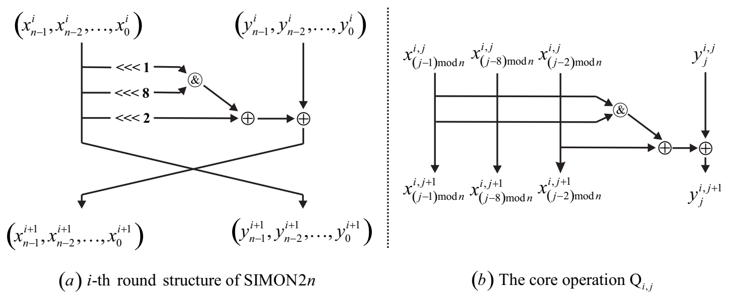
$(x^i_{n-1}, x^i_{n-2}, \ldots, x^i_0)$
$(y^i_{n-1}, y^i_{n-2}, \ldots, y^i_0)$
<<<1
<<<8
<<<2
&
$(x^{i+1}_{n-1}, x^{i+1}_{n-2}, \ldots, x^{i+1}_0)$
$(y^{i+1}_{n-1}, y^{i+1}_{n-2}, \ldots, y^{i+1}_0)$
(a) i-th round structure of SIMON$2n$
$x^{i,j}_{(j-1)\bmod n}$ $x^{i,j}_{(j-8)\bmod n}$ $x^{i,j}_{(j-2)\bmod n}$
$y^{i,j}_j$
&
$x^{i,j+1}_{(j-1)\bmod n}$ $x^{i,j+1}_{(j-8)\bmod n}$ $x^{i,j+1}_{(j-2)\bmod n}$
$y^{i,j+1}_j$
(b) The core operation $Q_{i,j}$

Figure 3: The structure of SIMON$2n$

SIMECK is a family of lightweight block cipher proposed at CHES 2015 [YZS+15]. And its round function is very similar to that of SIMON except the rotation constants. We apply our automatic search algorithm to SIMON and SIMECK family of block ciphers. All the integral distinguishers are the same with the previous longest distinguishers [XZBL16][WHG+19].

In [KLT15], another variant of SIMON-like block cipher named SIMON(102) is proposed with rotation constants (1,0,2). Then, Hu K. *et al* [HW19] presented 19-, 27- and 35-round integral distinguishers of SIMON(102)32, SIMON(102)48, and SIMON(102)64 listed in Table 4.

Table 4: Integral distinguishers of SIMON(102) in [HW19]

Cipher	Distinguisher
19-SIMON(102)32	In: (caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa) Out: (????????????????, 0*?????????????*)
27-SIMON(102)48	In: (caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????, 0*?????????????????????*)
35-SIMON(102)64	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????????????,0*?????????????????????????????*)

'*' denotes that the sum of the output bit is constant (0 or 1), but the accurate value is unknown. Knowing the constant values will be helpful to integral attacks on ciphers. Therefore, we apply Algorithm 1 to search the integral distinguishers of SIMON(102), and obtain more accurate integral distinguishers listed in Table 5. Note that the integral distinguishers in Table 5 can also be found by BDPT.

Table 5: More accurate integral distinguishers of SIMON(102)

Cipher	Distinguisher
19-SIMON(102)32	In: (caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa) Out: (????????????????, 01?????????????1)
27-SIMON(102)48	In: (caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????, 01?????????????????????1)
35-SIMON(102)64	In: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa) Out: (????????????????????????????????,01?????????????????????????????1)

4.4 Applications to PRESENT and RECTANGLE

PRESENT [BKL+07] is a lightweight block cipher with SPN structure and consists of 31 rounds. The block length is 64 bits and two key lengths of 80 and 128 bits are supported. In order to improve the hardware efficiency, it use a fully wired diffusion layer. Fig. 4 illustrates one-round structure of PRESENT.

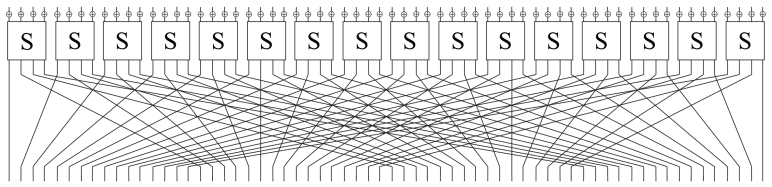

Figure 4: One-round structure of PRESENT

The structure of RECTANGLE [ZBL+15] is very like that of PRESENT. As applications, all the integral distinguishers obtained by Algorithm 1 are accordance with that in [WHG+19]. The detailed form of the integral distinguishers are listed in Table 6.

Table 6: Integral distinguishers of PRESENT and 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 Conclusions

In this paper, a new BDPT propagation rule of "Xor with The Secret Key" is proposed. It shows that some secret key bits can be bypassed. Then, we propose an improved automatic algorithm of searching integral distinguishers and apply it to some block ciphers. With this algorithm, some better integral distinguishers have been found.

It is worth noting that the integral distinguishers found by our algorithm have no significant improvement compared to the previous results. The reason may be that the previous integral distinguishers have already closed to the best distinguishers (the longest distinguishers that work for all keys). For SPECK32, SIMON32, SIMECK32, SIMON(102)32 and KATAN/KTANTAN32, because the block size is only 32 bits, we can observe the behaviors of all the plaintexts under fixed keys. Then, we experimentally proved that the integral distinguishers found by our automatic algorithm are the best integral distinguishers that work for all secret keys. That means our algorithm is very accurate in searching integral distinguishers. For ciphers with large block size, due to the limit of computing resources, we cannot give the experimental proof. As far as we know, there is still no result on the provable security against integral attack. This will be our future work.

Acknowledgments

We would like to thank the anonymous reviewers for their detailed comments and suggestions. This work was supported by the National Natural Science Foundation of China [Grant No.61572516, 61802437, 61902428].

References

[BC16] Christina Boura and Anne Canteaut. Another view of the division property. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 654–682. Springer, 2016.

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[CDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In Christophe Clavier and Kris Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 272–288. Springer, 2009.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *FSE '97*, volume 1267 of *LNCS*, pages 149–165. Springer, 1997.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[GRB] Zonghao Gu, Edward Rothberg, and Robert Bixby. Gurobi optimizer. `http://www.gurobi.com/`.

[HLM+20] Yonglin Hao, Gregor Leander, Willi Meier, Yosuke Todo, and Qingju Wang. Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 466–495. Springer, 2020.

[HW19] Kai Hu and Meiqin Wang. Automatic search for a variant of division property using three subsets. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 412–432. Springer, 2019.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015.

[KW02] Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.

[LSL09] Ping Li, Bing Sun, and Chao Li. Integral cryptanalysis of ARIA. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology*, volume 6151 of *LNCS*, pages 1–14. Springer, 2009.

[LW11] Yan-Jun Li and Wen-Ling Wu. Improved integral attacks on rijndael. *J. Inf. Sci. Eng.*, 27(6):2031–2045, 2011.

[SHZ+17] Bing Sun, Xin Hai, Wenyu Zhang, Lei Cheng, and Zhichao Yang. New observation on division property. *Sci. China Inf. Sci.*, 60(9):98102, 2017.

[SWW17] Ling Sun, Wei Wang, and Meiqin Wang. Automatic search of bit-based division property for ARX ciphers and word-based division property. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 128–157. Springer, 2017.

[TM16] Yosuke Todo and Masakatu Morii. Bit-based division property and application to simon family. In Thomas Peyrin, editor, *FSE 2016*, volume 9783, pages 357–377. Springer, 2016.

[Tod15a] Yosuke Todo. Integral cryptanalysis on full MISTY1. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume 9215 of *LNCS*, pages 413–432. Springer, 2015.

[Tod15b] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 287–314. Springer, 2015.

[WHG+19] SenPeng Wang, Bin Hu, Jie Guan, Kai Zhang, and Tairong Shi. Milp-aided method of searching division property using three subsets and applications. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019*, volume 11923 of *LNCS*, pages 398–427. Springer, 2019.

[WHT+18] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 275–305. Springer, 2018.

[WLV+14] Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of reduced-round SIMON32 and SIMON48. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 143–160. Springer, 2014.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 648–678, 2016.

[YZS+15] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The simeck family of lightweight block ciphers. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 307–329. Springer, 2015.

[ZBL+15] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Sci. China Inf. Sci.*, 58(12):1–15, 2015.

[ZRHD08] Muhammad Reza Z'aba, Håvard Raddum, Matthew Henricksen, and Ed Dawson. Bit-pattern based integral attack. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 363–381. Springer, 2008.