A Model Set Method to Search Integral Distinguisher Based on Division Property for Block Ciphers

Liu Zhang[1,2], Huawei Liu[1,2], and Zilong Wang[1,2]

[1] School of Cyber Engineering, Xidian University, Xi'an, China,
{liuzhang@stu., liuhw@stu.,zlwang@}xidian.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China

Abstract. In this paper, we focus on constructing an automatic search model that greatly improves efficiency with little loss of accuracy and obtains some better results in the construction of integral distinguishers for block ciphers. First, we define a new notion named *BDPT Trail*, which divides BDPT propagation into three parts: the *division trail* for $\mathbb{K}$, *division trail*for $\mathbb{L}$, and Key-Xor operation. Secondly, we improve the insufficiency of the previous methods of calculating division trails and propose an effective algorithm that can obtain more valid division trails for $\mathbb{L}$ of the S-box operation. Third, we propose a new algorithm that models each Key-Xor operation based on the MILP technique for the first time. Based on this, we can accurately characterize the Key-Xor operation by solving these MILP models. After that, by selecting the appropriate initial BDPT and stopping rules, we construct an automatic search model. As a result, our automatic search model is applied to search for integral distinguishers for some block ciphers. For GIFT-64, we find a 11-round integral distinguisher, which is one more round than the previous best results. For Rectangle, we find a better 10-round integral distinguisher with 9 balanced bits, which has eight more bits than the previous best results. For Simon64, we can find more balanced bits than the previous longest distinguishers.

Keywords: Division Property · Three-subset · MILP · Block Ciphers · Cross Propagation

1 Introduction

Integral cryptanalysis is one of the most powerful cryptanalysis techniques [9]. For a set of chosen plaintexts, attackers encrypt them r rounds and calculate the value of the XOR of all ciphertexts. If the value is 0, we say that the cipher has a r-round integral distinguisher.

Division property, a generalization of the integral property, which was proposed by Todo at EUROCRYPT 2015 [19], could explicitly describe the properties hidden between the traditional integral ALL and BALANCE properties. Later in CRYPTO 2015, Todo [18] applied the division property to MISTY1 and achieved the first theoretical integral attack of full-round MISTY1, which proves the superiority of the division property. Sun *et al.* [13] reviewed the division property and studied the property of a multiset satisfying certain division properties. At CRYPTO 2016, Boura and Canteaut proposed a new notion called *parity set* to characterize the division property of the S-box, based on which they found a better integral distinguisher for PRESENT [4].

In order to exploit the algebraic structure of the round function, Todo and Morii [20] proposed the bit-based division property, which treats each bit of the target primitive independently. The bit-based division property can be divided into two categories: the conventional bit-based division property (CBDP) and the bit-based division property using three subsets (BDPT). The CBDP classify all vectors $\boldsymbol{u} \in \mathbb{F}_2^n$ into two subsets such that the parity of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is 0 or *unknown*, while BDPT divides all vectors $\boldsymbol{u} \in \mathbb{F}_2^n$ into three subsets such that the parity of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is 0, 1 or *unknown*. Essentially, the set *unknown* in CBDP is divided into sets 1 and *unknown* in BDPT. Therefore, the BDPT can characterize the integral property of the primitive with more precision. For example, CBDP has found a 14-round integral distinguisher of SIMON32 while BDPT has found a 15-round integral distinguisher of SIMON32 [20]. However, the complexity of utilizing CBDP or BDPT is upper bounded by 2^n, where n denotes the block size.

Automatic Searching Integral Distinguishers Based on CBDP. To solve the restriction of huge complexity, Xiang *et al.* used the Mixed Integral Linear Programming (MILP) technique to construct an automatic search model, which was successfully applied to search integral distinguishers for lightweight ciphers whose block sizes were greater than 32 at ASIACRYPT 2016 [26]. By extending and improving the method, integral attacks have been applied to many ciphers and many better integral distinguishers have been found [14,5,21,15].

Automatic Searching Integral Distinguishers Based on BDPT. There are two problems in constructing an automatic search model based on BDPT.

1. **Feasibility and Efficiency.** The automatic search model should be solved in practical time by openly available solvers.

2. **Accuracy and Completeness.** The automatic search model needs to accurately and completely characterize the complex propagation of BDPT, which means the set $\mathbb{K}$, $\mathbb{L}$, and the influence of the set $\mathbb{L}$ on the set $\mathbb{K}$ should be traced.

To address the above problems, Hu *et al.* [8] proposed an automatic search model for a variant three-subset division property and applied the method to improve some integral distinguishers. Later in ASIACRYPT 2019, Wang *et al.* [24] proposed the pruning technique that removes redundant vectors, and a new concept ("fast propagation") that can translate BDPT into CBDP. Then, they constructed a new automatic search model based on the above techniques to search integral distinguishers. However, their methods sacrifice some accuracy of the original BDPT. The three-subset division property without unknown subset presented at EUROCRYPT 2020 [6] and the monomial prediction presented at ASIACRYPT 2020 [7] are two accurate techniques for the division property. These two techniques provide an essential description of the division property from the propagation [6] and algebraic [7] perspectives, respectively. Their main idea is to determine the parity of a public function by counting the number of solutions (or the number of monomial trails). It is the simplest solution to overcome the unknown-producing property caused by the Key-Xor operation and the cancellation property caused by the XOR operation [6]. However, when applying the technique to block ciphers directly, especially S-box-based ciphers, it is difficult to count the number of solutions in practical time by openly available solvers and the computational cost is too high.

Our Contributions. In this paper, we construct an automatic search model that greatly improves the efficiency for block ciphers with a slight loss of accuracy and obtains some better results in the construction of integral distinguishers for block ciphers. The details of our contributions are summarized as follows.

– **BDPT Trail.** We define a new notion named *BDPT Trail* to completely and accurately characterize the BDPT propagation. The *BDPT Trail* divides the BDPT propagation into three parts: the propagation of the set $\mathbb{K}$, the propagation of the set $\mathbb{L}$, and the Key-Xor operation. Furthermore, we introduce two notions named *division trail* for $\mathbb{K}$ and *division trail* for $\mathbb{L}$ to illustrate the propagation of sets $\mathbb{K}$ and $\mathbb{L}$, respectively. With these notions, building an automatic search model that characterizes the BDPT propagation is equivalent to modeling the *division trail* for $\mathbb{K}$, *division trail* for $\mathbb{L}$, and Key-Xor operation.

– **Model the BDPT propagation of Nonlinear Layer.** We first propose an "S-box" technique, which treats the nonlinear layer of a block cipher as a Blackbox, focusing only on its input and output, not on specific operations. More precisely, the "S-box" technique treats the basic operations that provide nonlinearity in non-S-box-based ciphers as an S-box. Using the "S-box" technique, we construct a generalized model that reduces the number of basic operations and models the nonlinear layer uniformly. More specifically, we transform the BDPT modeling of the nonlinear layer into the BDPT modeling of the S-box. To characterize the BDPT propagation of an S-box, we apply the method in [4,26] to calculate all the division trails for $\mathbb{K}$ of the S-box, and then we study the method to calculate all valid division trails for $\mathbb{L}$ of the S-box. We show that the method in [8] only finds a part of all valid division trails for $\mathbb{L}$, and the method in [24] finds some extra invalid division trails for $\mathbb{L}$. Then we present a theorem that can accurately find all valid division trails for $\mathbb{L}$ of the S-box according to its ANF directly. Based on this, we propose an effective algorithm that can obtain more valid division trails for $\mathbb{L}$ of the S-box. To model the division trails for $\mathbb{K}$ and division trails for $\mathbb{L}$ of the S-box by a set of linear inequalities whose feasible solutions are exactly these division trails, we use SageMath [17] to generate an initial set of linear inequalities and then apply a reduction algorithm to reduce the initial set such that these division trails can be modeled by the minimum number of linear inequalities [12].

– **Model the BDPT propagation of Key-Xor operation.** When a Key-Xor operation is applied, new vectors generated from the set $\mathbb{L}$ will be added to the set $\mathbb{K}$. Therefore, how to accurately characterize the Key-Xor operation is a complex problem. To solve this problem, we propose a new algorithm that models each Key-Xor operation based on the MILP technique for the first time. Based on this, we can accurately characterize the Key-Xor operation by solving these MILP models. Finally, by selecting appropriate initial BDPT and stopping rules, we can construct an automatic search model that greatly improves the efficiency for block ciphers at a little loss of accuracy, and obtain some better results.

– **Applications.** We apply our automatic search model to search integral distinguishers of SIMON [2], SIMECK [27], RECTANGLE [28], PRESENT [3] and GIFT-64 [1]. The results are shown in Table 1.

1. **For non-S-box-based block ciphers.** For SIMON64, we can find a better 17-round integral distinguisher with 27 balanced bits, which has four more bits than the previous longest distinguisher [24]. For SIMON32, 48, 96, 128 and SIMECK32, 48, 64, the distinguishers we find are in accordance with the previous longest distingui-shers [24].
2. **For S-box-based block ciphers.** For GIFT-64, we find an 11-round integral distinguisher which is one round more than the previous best results. For RECTANGLE, we find a 10-round integral distinguisher with 9 balanced bits, which has eight more bits than the best integral distinguisher in [11]. For PRESENT, the distinguishers we find are in agreement with the previous longest distinguishers [24]. For the above five block ciphers, our automatic search model reduces the time complexity of searching integral distinguishers compared to the previous methods.

Organization. This paper is organized as follows: In Section 2 we briefly review some basic background knowledge about the bit-based division property. Section 3 studies how to model these operations used in the round function of a block cipher by the MILP technique. Section 4 studies the initial and stopping rules and the search algorithm. Section 5 shows the applications of some lightweight block ciphers, and we conclude our work in Section 6. Some auxiliary materials are provided in the Appendix.

Table 1: Summarization of integral distinguishers

Cipher	Data	Round	$NBB^{\star}$	Time	Ref.†
SIMON32	2^{31}	15	3	1h48m	[23]
				2.0m	[24]
		15	3	1.6m	Sect. 5
SIMON48	2^{47}	16	24	1h48m	[23]
		16	24	8.4m	Sect. 5
SIMON64	2^{63}	18	23	23h31m	[23]
				1h41m	[24]
		18	**27**	1h8m	Sect. 5
SIMON96	2^{95}	22	5	31h25m	[23]
		22	5	5h55m	Sect. 5
SIMON128	2^{127}	26	3	62h16m	[23]
		26	3	21h7m	Sect. 5
SIMECK32	2^{31}	15	7	51m	[23]
		15	7	1.3m	Sect. 5
SIMECK48	2^{47}	18	5	5h3m	[23]
		18	5	12.9m	Sect. 5
SIMECK64	2^{63}	21	5	23h25m	[23]
		21	5	47.3m	Sect. 5
PRESENT	2^{63}	9	28	4h8m	[23]
				10m	[24]
	2^{63}	9	28	4.6m	Sect. 5
RECTANGLE	2^{63}	10	1	6.7m	[11]
	2^{63}	10	**9**	5.3m	Sect. 5
GIFT-64	2^{63}	10	32	——	[1]
	2^{63}	**11**	16	39.1m	Sect. 5

1. † : The paper [23] in IACR Cryptology ePrint Archive is the preprint of [24]. The results of the two papers are consistent except for the time complexity.
2. * : The balanced bit is divided into `0` and `1`, where "`0`" represents the bit whose sum is 0, "`1`" represents the bit whose sum is 1. The details results are shown in Appendix F.
3. NBB: the number of balanced bits.

2 Preliminaries

2.1 Notations

Let $\mathbb{F}_2$ be the finite field $\{0,1\}$ and $\mathbb{F}_2^n$ be the n-bit string over $\mathbb{F}_2$. For any $a \in \mathbb{F}_2^n$, let $a[i]$ be the i-th bit of a, and the Hamming weight of a is calculated as $\sum_{i=0}^{n-1} a[i]$. For any $\boldsymbol{a} = (a_0, \dots, a_{m-1}) \in \mathbb{F}_2^{n_0} \times \cdots \times \mathbb{F}_2^{n_{m-1}}$, the vectorial Hamming weight of $\boldsymbol{a}$ is defined as $W(\boldsymbol{a}) = (w(a_0), \dots, w(a_{m-1})) \in \mathbb{Z}^m$, where $w(a_i)$ is the Hamming weight of a_i, and $\mathbb{Z}$ denotes the integer ring. For any $\boldsymbol{k} \in \mathbb{Z}^m$ and $\boldsymbol{k}' \in \mathbb{Z}^m$, we define $\boldsymbol{k} \succeq \boldsymbol{k}'$ if $k_i \geqslant k_i'$ for all $i = 0, 1, \dots, m-1$. Otherwise, $\boldsymbol{k} \not\succeq \boldsymbol{k}'$. Let $\mathbb{K}$ be the set of $\boldsymbol{k}$, and let $|\mathbb{K}|$ be the number of vectors in $\mathbb{K}$. Moreover, we simply write $\mathbb{K} \leftarrow \boldsymbol{k}$ when $\mathbb{K} := \mathbb{K} \cup \{\boldsymbol{k}\}$.

Bit Product Function [19] For any $u \in \mathbb{F}_2^n$, let $x \in \mathbb{F}_2^n$ be the input. The function $\pi_u(x) : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as $\pi_u(x) := \prod_{i=0}^{n-1} x[i]^{u[i]}$

For any $\boldsymbol{u} = (u_0, u_1, \dots, u_{m-1}) \in \mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_{m-1}}$, let $\boldsymbol{x} = (x_0, x_1, \dots, x_{m-1}) \in \mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_{m-1}}$be the input. The $\pi_{\boldsymbol{u}}(\boldsymbol{x}) : (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_{m-1}}) \to \mathbb{F}_2$ is defined as $\pi_{\boldsymbol{u}}(\boldsymbol{x}) := \prod_{i=0}^{m-1} \pi_{u_i}(x_i)$.

2.2 Bit-Based Division Property

Todo and Morii proposed two types of bit-based division property (CBDP and BDPT) at FSE 2016 [20].

Definition 1. *((Bit-based) division property(**CBDP**)[20]) Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, it satisfies the following conditions:*

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

where $\boldsymbol{u} \succeq \boldsymbol{k}$ if $u_i \geq k_i$ for all i.

The bit-based division property using three subsets (BDPT) limits the underlying space to binary domains and further expands the search scope. Namely, it introduces a new set $\mathbb{L}$, which is the set of $\boldsymbol{u}$ with $\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = 1$.

Definition 2. *(**BDPT [20]**) Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the bit-based division property using three subsets $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, it satisfies the following conditions:*

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

According to [20], if there are $\boldsymbol{k}, \boldsymbol{k}' \in \mathbb{K}$ satisfying $\boldsymbol{k} \succeq \boldsymbol{k}'$, $\boldsymbol{k}$ can be removed from $\mathbb{K}$ because the vector $\boldsymbol{k}$ is redundant. Moreover, if there are $\boldsymbol{\ell} \in \mathbb{L}$ and $\boldsymbol{k} \in \mathbb{K}$ that satisfy $\boldsymbol{\ell} \succeq \boldsymbol{k}$, the vector $\boldsymbol{\ell}$ is redundant. For any $\boldsymbol{u}$, the redundant vectors in $\mathbb{K}$ and $\mathbb{L}$ will not affect the parity of $\bigoplus_{\boldsymbol{x} \in \mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$.

Propagation Rules. We introduce only a few of the propagation rules used in the following sections. For more details, see [20].

Rule 1 (Key-Xor [20]) *Let the input and output division property of the Key-Xor operation be $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$ and $\mathcal{D}^{1^n}_{\mathbb{K}',\mathbb{L}'}$, respectively. Assume that the round key is XORed with the i-th bit, $\mathbb{K}'$ and $\mathbb{L}'$ are computed as*

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

Boura *et al.* presented the propagation rules of S-box for $\mathbb{K}$ at bit-level in [4] for the first time, which is summarized in Rule 2.

Rule 2 (S-box for $\mathbb{K}$ [4]) *Let $F : \mathbb{F}_2^m \to \mathbb{F}_2^n$ be a vectorial Boolean function composed of $(f_0, f_1, \ldots, f_{n-1})$, where each $f_i : \mathbb{F}_2^m \to \mathbb{F}_2$ is a Boolean function. Assuming the input to the function F is $\boldsymbol{x} = (x_0, x_1, \ldots, x_{m-1}) \in \mathbb{F}_2^m$, the output $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ is calculated as*

$$\begin{aligned}y_0 &= f_0(x_0, x_1, \ldots, x_{m-1}),\\ y_1 &= f_1(x_0, x_1, \ldots, x_{m-1}),\\ &\vdots\\ y_{n-1} &= f_{n-1}(x_0, x_1, \ldots, x_{m-1}).\end{aligned}$$

For each vector $\boldsymbol{k}$ in the input division property $\mathbb{K}$, check each vector $\boldsymbol{u} \in \mathbb{F}_2^n$ whether the polynomial $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial $\pi_{\boldsymbol{k}'}(\boldsymbol{x})$ satisfying $\boldsymbol{k}' \succeq \boldsymbol{k}$. If so, $(\boldsymbol{k}, \boldsymbol{u})$ is a valid division trail for the S-box function.

2.3 MILP-Based Bit-Based Division Property

At ASIACRYPT 2016, Xiang *et al.* [26] applied the MILP method to the search for CBDP for the first time. With the help of the Gurobi MILP solver, they can find CBDP for block ciphers with larger block sizes, e.g. Simon128 or Present. They introduced the definition of the CBDP trail, which is defined below.

Definition 3. *(**CBDP Trail [26]**) Let f_r denote the round function of an iterated block cipher. Assume that the input multiset of the block cipher has the initial division property $\mathcal{D}^{1^n}_{\{\boldsymbol{k}\}}$, and denote the division property after i-round propagation through f_r by $\mathcal{D}^{1^n}_{\mathbb{K}_i}$. Thus, we have the following chain of division property propagations:*

$$\{\boldsymbol{k}\} \triangleq \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots$$

Furthermore, for any vector $\boldsymbol{k}_i^* \in \mathbb{K}_i (i \geqslant 1)$, there must exist a vector $\boldsymbol{k}_{i-1}^* \in \mathbb{K}_{i-1}$ such that $\boldsymbol{k}_{i-1}^*$ can propagate to $\boldsymbol{k}_i^*$ using the propagation rules of division properties. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$, if $\boldsymbol{k}_{i-1}$ can propagate to $\boldsymbol{k}_i$ for all $i \in \{1, 2, \ldots, r\}$, we call $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r)$ an r-round division trail.

Xiang *et al.* [26] modeled CBDP propagations of basic operations (Copy, Xor, And) and the S-box by linear inequalities. Iterating this process r times, they could build a MILP model to cover all the possible CBDP trails generated from a given initial CBDP. In our model, we will treat these basic operations as an S-box. Therefore, we only introduce the MILP models for S-box.

Model 1 (S-box [26]) *The CBDP Rule 2 in Sect. 2.2 can generate the CBDP propagation property of the S-box. Then, we use the **inequality_generator** function in SageMath [17] to get a set of linear inequalities. Sometimes, the number of linear inequalities in the set is large. Thus, a Greedy Algorithm [16] was proposed to reduce this set.*

3 Modeling BDPT Propagations

Suppose that an iterated block cipher of the round function f_r consists of a nonlinear layer, a linear layer, and a Key-Xor operation. Let f_k be the Key-Xor operation, and let f_e be the rest of the operations in the round function f_r. To model the propagation of BDPT for the operation f_e and f_k, we define a new notion named *BDPT trail.*

Definition 4. ***(BDPT Trail)*** *Assume that the input multiset $\mathbb{X}$ to the block cipher has initial BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$ and denote the BDPT after r-round propagation through f_e and f_k by $\mathcal{D}^{1^n}_{\mathbb{K}_r,\mathbb{L}_r}$, where $r \geqslant 1$. Thus we have the following chain of BDPT propagations:*

$$\begin{tikzcd}
\{\boldsymbol{k}\} \triangleq \mathbb{K}_0 \arrow[r, "f_e"] & \mathbb{K}_1 \arrow[r, "f_e"] & \mathbb{K}_2 \arrow[r, "f_e"] & \cdots \arrow[r, "f_e"] & \mathbb{K}_{r-1} \arrow[r, "f_e"] & \mathbb{K}_r \\
 & & & \cdots & & \\
\{\boldsymbol{\ell}\} \triangleq \mathbb{L}_0 \arrow[r, "f_e"] & \mathbb{L}_1 \arrow[uu, "f_k"'] \arrow[r, "f_e"] & \mathbb{L}_2 \arrow[uu, "f_k"'] \arrow[r, "f_e"] & \cdots \arrow[r, "f_e"] & \mathbb{L}_{r-1} \arrow[uu, "f_k"'] \arrow[r, "f_e"] & \mathbb{L}_r \arrow[uu, "f_k"']
\end{tikzcd}$$

where $\mathbb{K}_i = f_e(\mathbb{K}_{i-1}) \cup f_k(\mathbb{L}_i) = f_e(\mathbb{K}_{i-1}) \cup f_k \circ f_e(\mathbb{L}_{i-1})$, $\mathbb{L}_i = f_e(\mathbb{L}_{i-1})$, $1 \leqslant i \leqslant r$.

Moreover, for any vector tuple $(\boldsymbol{k}_i^*, \boldsymbol{\ell}_i^*)$, $\boldsymbol{k}_i^* \in \mathbb{K}_i$ and $\boldsymbol{\ell}_i^* \in \mathbb{L}_i (1 \leqslant i \leqslant r)$, there must exist a vector tuple $(\boldsymbol{k}_{i-1}^*, \boldsymbol{\ell}_{i-1}^*)$, $\boldsymbol{k}_{i-1}^* \in \mathbb{K}_{i-1}$ and $\boldsymbol{\ell}_{i-1}^* \in \mathbb{L}_{i-1}$, such that $(\boldsymbol{k}_{i-1}^*, \boldsymbol{\ell}_{i-1}^*)$ can propagate to $(\boldsymbol{k}_i^*, \boldsymbol{\ell}_i^*)$ by BDPT propagation rules. Furthermore, for $((\boldsymbol{k}_0, \boldsymbol{\ell}_0), (\boldsymbol{k}_1, \boldsymbol{\ell}_1), \ldots, (\boldsymbol{k}_r, \boldsymbol{\ell}_r)) \in \mathbb{K}_0 \times \mathbb{L}_0 \times \mathbb{K}_1 \times \mathbb{L}_1 \times \cdots \times \mathbb{K}_r \times \mathbb{L}_r$, if $(\boldsymbol{k}_{i-1}, \boldsymbol{\ell}_{i-1})$ can propagate to $(\boldsymbol{k}_i, \boldsymbol{\ell}_i)$ for all $i \in \{1, 2, \ldots, r\}$, we call $(\boldsymbol{k}_0, \boldsymbol{\ell}_0) \xrightarrow{f_e, f_k} (\boldsymbol{k}_1, \boldsymbol{\ell}_1) \xrightarrow{f_e, f_k} \cdots \xrightarrow{f_e, f_k} (\boldsymbol{k}_r, \boldsymbol{\ell}_r)$ an r-round BDPT trail.

By ignoring the Key-Xor operation (which causes the vector $\boldsymbol{\ell} \in \mathbb{L}_i$ to be added to the set $\mathbb{K}_i$), we can get the following two chains, which reflect the propagation property of f_e. Note that the purpose of 'ignoring' is to define two new notions to describe the BDPT propagation.

$$\{\boldsymbol{k}\} \triangleq \mathbb{K}'_0 \xrightarrow{f_e} \mathbb{K}'_1 \xrightarrow{f_e} \mathbb{K}'_2 \xrightarrow{f_e} \cdots \xrightarrow{f_e} \mathbb{K}'_{r-1} \xrightarrow{f_e} \mathbb{K}'_r$$
$$\{\boldsymbol{\ell}\} \triangleq \mathbb{L}_0 \xrightarrow{f_e} \mathbb{L}_1 \xrightarrow{f_e} \mathbb{L}_2 \xrightarrow{f_e} \cdots \xrightarrow{f_e} \mathbb{L}_{r-1} \xrightarrow{f_e} \mathbb{L}_r$$

where $\mathbb{K}'_i = f_e(\mathbb{K}'_{i-1})$, $\mathbb{L}_i = f_e(\mathbb{L}_{i-1})$, $1 \leqslant i \leqslant r$.

Thus, 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* $\boldsymbol{k}'_{i-1}$ *can propagate to* $\boldsymbol{k}'_i$ *for all* $i \in \{1, 2, \ldots, r\}$*, we call* $(\boldsymbol{k}'_0, \boldsymbol{k}'_1, \ldots, \boldsymbol{k}'_r)$ *an* r*-round division trail for* $\mathbb{K}$*. Similarly, for* $(\boldsymbol{\ell}_0, \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_r) \in \mathbb{L}_0 \times \mathbb{L}_1 \times \cdots \times \mathbb{L}_r$*, if* $\boldsymbol{\ell}_{i-1}$ *can propagate to* $\boldsymbol{\ell}_i$ *for all* $i \in \{1, 2, \ldots, r\}$*, we call* $(\boldsymbol{\ell}_0, \boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_r)$ *an* r*-round division trail for* $\mathbb{L}$*.*

Similar to the methods in [26], for an initial BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$, we determine whether there exist useful integral distinguishers after r-round encryption, by finding all r-round BDPT trails that start with the vector tuple $(\boldsymbol{k}, \boldsymbol{\ell})$. Thus, we need to accurately describe all valid division trails of the vectors $\boldsymbol{k}$ and $\boldsymbol{\ell}$ through f_e and f_k. For the operation f_e, we model the division trail for $\mathbb{K}$ and the division trail for $\mathbb{L}$, respectively. For the operation f_k, we construct a new MILP model to characterize the process in which part of vectors $\boldsymbol{\ell} \in \mathbb{L}_i$ is added to the set $\mathbb{K}_i$ by the Key-Xor operation.

3.1 Treat Nonlinear Layer as "S-box"

We classify block ciphers into two categories based on whether there is an S-box in the nonlinear layer. When we apply BDPT to non-S-box-based ciphers, we usually need to consider each of its specific operations for primitives. Taking the Simon family as an example, we have to consider how to represent these basic operations with a set of linear inequalities, such as Copy, Xor, And. We aim to construct a generalized model that reduces the number of basic operations and to model the non-linear layer uniformly. The intuitive idea is to regard these basic operations that provide nonlinearity as an S-box, which is named the "S-box". Theoretically, the core operation of the Simon family is represented by Fig. 1. We refer to the part surrounded by the red dotted line as the "S-box".

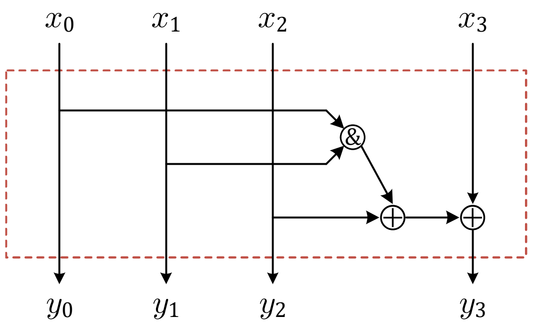

Fig. 1: Core operation of the Simon family [20] and "S-box"

We represent the input to the "S-box" as $\boldsymbol{x} = (x_0, x_1, x_2, x_3)$, and the corresponding output as $\boldsymbol{y} = (y_0, y_1, y_2, y_3)$, the algebraic normal form (ANF) of the "S-box" is listed as follows:

$$\begin{aligned} y_0 &= x_0, \quad y_1 = x_1, \\ y_2 &= x_2, \quad y_3 = x_0x_1 \oplus x_2 \oplus x_3. \end{aligned} \tag{1}$$

The S-box is an important component for most S-box-based block ciphers because it is the only nonlinear part. For non-S-box-based block ciphers, the "S-box" serves the same purpose. On the basis of this, we transform the BDPT modeling of the nonlinear layer into the BDPT modeling of the S-box.

3.2 Limitation of Previously BDPT Modeling of an S-box

In [4,26], the rule was presented to calculate all the division trails for $\mathbb{K}$ of an S box. We study the rule to find all valid division trails for $\mathbb{L}$ of an S-box.

We assume an n-bit S-box: $\mathbb{F}_2^n \to \mathbb{F}_2^n$ is composed of $(f_0, f_1, \ldots, f_{n-1})$, where the input $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_2^n$ and the output $\boldsymbol{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_2^n$. Every y_i can be expressed as a Boolean function of $(x_0, \ldots, x_{n-1})$, where $i \in \{0, \ldots, n-1\}$.

Theorem 1 ([8,24]). *If the input BDPT of the S-box is $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$ where $\boldsymbol{k} = (k_0, \ldots, k_{n-1}), \boldsymbol{\ell} = (\ell_0, \ldots, \ell_{n-1})$, then the output BDPT of the S-box can be calculated by $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}_1}$ or $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}_2}$, where*

$$\begin{aligned}
&\mathbb{K} = \{\boldsymbol{u} \in \mathbb{F}_2^n \mid \pi_{\boldsymbol{u}}(\boldsymbol{y}) \text{ contains any monomial } \pi_{\bar{\boldsymbol{k}}}(\boldsymbol{x}) \text{ satisfying } \bar{\boldsymbol{k}} \succeq \boldsymbol{k}\} \\
&\mathbb{L}_1 = \{\boldsymbol{u} \in \mathbb{F}_2^n \mid \pi_{\boldsymbol{u}}(\boldsymbol{y}) \text{ does not contain any monomial } \pi_{\bar{\boldsymbol{\ell}}}(\boldsymbol{x}) \text{ satisfying } \bar{\boldsymbol{\ell}} \succ \boldsymbol{\ell} \\
&\qquad \text{and } \pi_{\boldsymbol{u}}(\boldsymbol{y}) \text{ contains } \pi_{\boldsymbol{\ell}}(\boldsymbol{x})\}. \\
&\mathbb{L}_2 = \{\boldsymbol{u} \in \mathbb{F}_2^n \mid \pi_{\boldsymbol{u}}(\boldsymbol{y}) \text{ contains } \pi_{\boldsymbol{\ell}}(\boldsymbol{x})\}.
\end{aligned}$$

Remark 1. The rules $\mathbb{K}$ and $\mathbb{L}_1$ are derived from [8]. Furthermore, the rules $\mathbb{K}$ and $\mathbb{L}_2$ are derived from [24].

There have been two previous methods to calculate all division trails for $\mathbb{L}$ of an S-box, as shown in Theorem 1. We briefly describe these methods as follows: firstly, by the algebraic normal form (ANF) of the S-box, each element y_i in the output $\boldsymbol{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_2^n$ can be represented as a boolean function by the input $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$, that is, $y_i = f_i(x_0, \ldots, x_{n-1})$, where $0 \leqslant i \leqslant n-1$. Secondly, suppose that for the input $\boldsymbol{\ell} = (\ell_0, \ldots, \ell_{n-1})$, for every $\boldsymbol{u} \in \mathbb{F}_2^n$, we calculate the $\pi_{\boldsymbol{u}}(\boldsymbol{y}) = \prod_{i=0}^{n-1} y[i]^{u[i]}$, where $y[i] = f_i(x_0, \ldots, x_{n-1})$. Then, we obtain $\pi_{\boldsymbol{u}}(\boldsymbol{y}) = \prod_{i=0}^{n-1} f_i(x_0, \ldots, x_{n-1})^{u[i]}$, which is a polynomial representation about $\boldsymbol{x}$. Finally, according to rule $\mathbb{L}_1$ or $\mathbb{L}_2$ of Theorem 1, for the input vector $\boldsymbol{\ell}$, check for each vector $\boldsymbol{u} \in \mathbb{F}_2^n$ whether the polynomial $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$ and $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ does not contain any monomial $\pi_{\bar{\boldsymbol{\ell}}}(\boldsymbol{x})$ satisfying $\bar{\boldsymbol{\ell}} \succ \boldsymbol{\ell}$(or polynomial $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$). If so, $(\boldsymbol{\ell}, \boldsymbol{u})$ is a valid division trail for $\mathbb{L}$ of the S-box.

We find that the rules $\mathbb{L}_1$ and $\mathbb{L}_2$ both have some limitations, which are illustrated with two specific examples as follows.

Example 1. (shows that the rule $\mathbb{L}_1$ only finds a part of the division trails for $\mathbb{L}$) Take the "S-box", which represents the core operation of the Simon family as an example. The "S-box" is a 4×4 S-box, and its input and output are shown in Fig. 1. Assume that the input multiset $\mathbb{X}$ to the "S-box" has BDPT $\mathcal{D}^{1^4}_{\boldsymbol{k},\boldsymbol{\ell}=(0,1,1,0)}$. The process of obtaining the valid division trail $(\boldsymbol{\ell}, \boldsymbol{u})$ is briefly described as follows:

Table 2: Propagation of the bit-based division property using three subsets for the Core Operation in Simon [20]

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

First, we get $y_i = f_i(x_0,x_1,x_2,x_3)$ where $0 \leqslant i \leqslant 3$, which is a Boolean function expression about the input $\boldsymbol{x}$, as shown in Equation (1). Second, for every $\boldsymbol{u} \in \mathbb{F}_2^4$, we calculate $\pi_{\boldsymbol{u}}(\boldsymbol{y}) = \prod_{i=0}^{3} f_i(x_0,x_1,x_2,x_3)^{u[i]}$. For ease of understanding, we take $\boldsymbol{u} = (0,1,0,1) \in \mathbb{F}_2^4$ and calculate $\pi_{(0,1,0,1)}(\boldsymbol{y}) = y_1y_3 = x_1(x_0x_1 \oplus x_2 \oplus x_3) = x_0x_1 \oplus x_1x_2 \oplus x_1x_3$. Finally, according to rule $\mathbb{L}_1$, we check whether $\pi_{(0,1,0,1)}(\boldsymbol{y}) = x_0x_1 \oplus x_1x_2 \oplus x_1x_3$ contains $\pi_{\boldsymbol{\ell}=(0,1,1,0)}(\boldsymbol{x}) = x_1x_2$ and does not contain any monomial $\pi_{\bar{\boldsymbol{\ell}}}(\boldsymbol{x})$ satisfying $\bar{\boldsymbol{\ell}} \succ \boldsymbol{\ell}$, where $\pi_{\bar{\boldsymbol{\ell}}}(\boldsymbol{x}) = \{x_0x_1x_2, x_1x_2x_3, x_0x_1x_2x_3\}$. Apparently, $\pi_{(0,1,0,1)}(\boldsymbol{y})$ satisfies the above conditions. So $(0,1,1,0) \to (0,1,0,1)$ is a valid division trail for $\mathbb{L}$ of the "S-box". To obtain all division trails for $\mathbb{L}$ where $\boldsymbol{\ell} = (0,1,1,0)$, we traverse $\boldsymbol{u} \in \mathbb{F}_2^4$ and get another valid division trail for $\mathbb{L}$:$(0,1,1,0) \to (0,1,1,0)$.

However, we find that a valid division trail for $\mathbb{L} : (0,1,1,0) \to (0,1,1,1)$ appears in Table 2 by [20] which cannot be found by the rule $\mathbb{L}_1$. The reason is that $\pi_{\boldsymbol{u}=(0,1,1,1)}(\boldsymbol{y}) = x_0x_1x_2 \oplus x_1x_2 \oplus x_1x_2x_3$ contains not only $\pi_{\boldsymbol{\ell}=(0,1,1,0)}(\boldsymbol{x}) = x_1x_2$ but also $x_0x_1x_2$ and $x_1x_2x_3 \in \pi_{\bar{\boldsymbol{\ell}}}(\boldsymbol{x})$. It is worth exploring the valid division trails for $\mathbb{L}$ which were missing by rule $\mathbb{L}_1$ compared to Table 2. Therefore, for each $\boldsymbol{\ell}, \boldsymbol{u} \in \mathbb{F}_2^4$, we calculate $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ and $\pi_{\bar{\boldsymbol{\ell}}}(\boldsymbol{x})$ where $\bar{\boldsymbol{\ell}} \succ \boldsymbol{\ell}$, and obtain all valid division trails for $\mathbb{L}$ of the "S-box" by rule $\mathbb{L}_1$. Note that the missing valid division trails for $\mathbb{L}$ compared to Table 2 are in bold.

Example 2. (shows that the rule $\mathbb{L}_2$ finds some extra invalid division trails for $\mathbb{L}$) Take the Present S-box as an example. Let the input to Present S-box be $\boldsymbol{x} = (x_0,x_1,x_2,x_3)$, and the corresponding output be $\boldsymbol{y} = (y_0,y_1,y_2,y_3)$, the algebraic normal form (ANF) of the Present S-box is shown in Equation (2).

Assume that the input multiset $\mathbb{X}$ to PRESENT S-box has BDPT $\mathcal{D}^{1^4}_{\boldsymbol{k},\boldsymbol{\ell}=(1,1,1,0)}$.

$$\begin{aligned} y_0 &= x_0x_1x_3 \oplus x_0x_2x_3 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_0 \oplus x_2 \oplus x_3 \oplus 1 \\ y_1 &= x_0x_1x_3 \oplus x_0x_2x_3 \oplus x_0x_2 \oplus x_0x_3 \oplus x_2x_3 \oplus x_0 \oplus x_1 \oplus 1 \\ y_2 &= x_0x_1x_3 \oplus x_0x_2x_3 \oplus x_1x_2x_3 \oplus x_0x_1 \oplus x_0x_2 \oplus x_0 \oplus x_2 \\ y_3 &= x_1x_2 \oplus x_0 \oplus x_1 \oplus x_3. \end{aligned} \tag{2}$$

Thus, for each $\boldsymbol{u} \in \mathbb{F}_2^4$, we calculate $\pi_{\boldsymbol{u}}(\boldsymbol{y}) = \prod_{i=0}^{3} f_i(x_0, x_1, x_2, x_3)^{u[i]}$. For ease of understanding, we take $\boldsymbol{u} = (0,1,0,1) \in \mathbb{F}_2^4$ and calculate $\pi_{(0,1,0,1)}(\boldsymbol{y}) = y_1y_3 = (x_0x_1x_3 \oplus x_0x_2x_3 \oplus x_0x_2 \oplus x_0x_3 \oplus x_2x_3 \oplus x_0 \oplus x_1 \oplus 1)(x_1x_2 \oplus x_0 \oplus x_1 \oplus x_3) = x_0x_1x_2 \oplus x_0x_2 \oplus x_0x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_3$. According to rule $\mathbb{L}_2$, we check whether $\pi_{(0,1,0,1)}(\boldsymbol{y}) = x_0x_1x_2 \oplus x_0x_2 \oplus x_0x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_3$ contains $\pi_{\boldsymbol{\ell}=(1,1,1,0)}(\boldsymbol{x}) = x_0x_1x_2$. Apparently, $\pi_{(0,1,0,1)}(\boldsymbol{y})$ satisfies this condition above, so $(1,1,1,0) \to (0,1,0,1)$ is a valid division trail for $\mathbb{L}$ of PRESENT S-box. To obtain all the division trails for $\mathbb{L}$ where $\boldsymbol{\ell} = (1,1,1,0)$, we traverse $\boldsymbol{u} \in \mathbb{F}_2^4$ and obtain the other division trails for $\mathbb{L}$:$(1,1,1,0) \to (0,1,1,1)$, $(1,1,1,0) \to (1,0,1,1)$, $(1,1,1,0) \to (1,1,0,1)$, $(1,1,1,0) \to (1,1,1,0)$ and $(1,1,1,0) \to (1,1,1,1)$.

However, we find that the division trail $(1,1,1,0) \to (1,1,1,1)$ discovered by rule $\mathbb{L}_2$ is an invalid division trail for $\mathbb{L}$, that is, $(1,1,1,0) \nrightarrow (1,1,1,1)$. The proof is described below:

$$\begin{aligned} &\bigoplus_{y\in\mathbb{Y}} \pi_{(1,1,1,1)}(\boldsymbol{y}) \\ =&\bigoplus_{y\in\mathbb{Y}} y_0y_1y_2y_3 \\ =&\bigoplus_{x\in\mathbb{X}} (x_0x_1x_2x_3 \oplus x_0x_1x_2 \oplus x_0x_2x_3 \oplus x_0x_2) \\ =&\bigoplus_{x\in\mathbb{X}} \pi_{(1,1,1,1)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \pi_{(1,1,1,0)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \pi_{(1,0,1,1)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \pi_{(1,0,1,0)}(\boldsymbol{x}) \\ =&\ \textit{unknown} \oplus 1 \oplus (0 \text{ or } \textit{unknown} \text{ depends on } \boldsymbol{k}) \oplus 0 \\ =&\ \textit{unknown}. \end{aligned}$$

According to Definition 2, the parity of $\bigoplus_{x\in\mathbb{X}} \pi_{(1,1,1,1)}(\boldsymbol{x})$ is *unknown* because $\boldsymbol{u} = (1,1,1,1) \succeq \boldsymbol{k}$ for any $\boldsymbol{k} \in \mathbb{F}_2^4$. So, the parity of $\bigoplus_{y\in\mathbb{Y}} \pi_{(1,1,1,1)}(\boldsymbol{y})$ is *unknown*. In other words, $(1,1,1,0) \to (1,1,1,1)$ is an invalid division trail for $\mathbb{L}$. Therefore, some extra invalid division trails for $\mathbb{L}$ may be obtained by the rule $\mathbb{L}_2$.

From Examples 1 and 2, we show that some valid division trails for $\mathbb{L}$ are missing by rule $\mathbb{L}_1$, and some extra invalid division trails for $\mathbb{L}$ are produced by rule $\mathbb{L}_2$. Thus, we have the following observation.

Observation 1 *The rule to calculate all the division trails for $\mathbb{L}$ of an S-box is between rules $\mathbb{L}_1$ and $\mathbb{L}_2$.*

From Theorem 1 and Observation 1, we find that rules $\mathbb{L}_1$ and $\mathbb{L}_2$ only consider $\pi_{\boldsymbol{u}}(\boldsymbol{y})$, not the specific parity of $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})$, which is *unknown* or 1 or 0. Based on this, we propose Theorem 2 and Observation 2.

Theorem 2. *Let $\boldsymbol{\ell}\in\mathbb{F}_2^n$ represent the input of an S-box. For any $\boldsymbol{u}\in\mathbb{F}_2^n$, $(\boldsymbol{\ell},\boldsymbol{u})$ is a valid division trail for $\mathbb{L}$ if and only if $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1$.*

Proof. Assume that the input multiset $\mathbb{X}$ in the S-box has BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$, where $\boldsymbol{k}=(k_0,\dots,k_{n-1}),\boldsymbol{\ell}=(\ell_0,\dots,\ell_{n-1})$, and the output multiset $\mathbb{Y}$ in the S-box has BDPT $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$. On the one hand, for any $\boldsymbol{u}\in\mathbb{F}_2^n$, if $(\boldsymbol{\ell},\boldsymbol{u})$ is a valid division trail for $\mathbb{L}$, we have $\boldsymbol{u}\in\mathbb{L}$. According to Definition 2, for any $\boldsymbol{\ell}'\in\mathbb{L}$, we have $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{\ell}'}(\boldsymbol{y})=1$. Thus, we get $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1$. On the other hand, if $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1$, we have $\boldsymbol{u}\in\mathbb{L}$ by Definition 2. For any $\boldsymbol{\ell}'\in\mathbb{L}$, the $(\boldsymbol{\ell},\boldsymbol{\ell}')$ is a valid division trail for $\mathbb{L}$. Then, we get a valid division trail for $\mathbb{L}$: $(\boldsymbol{\ell},\boldsymbol{u})$. Thus, Theorem 2 is proven.

Theorem 2 gives sufficient and necessary conditions for $(\boldsymbol{\ell},\boldsymbol{u})$ to be a valid division trail for $\mathbb{L}$, then we propose an observation as,

Observation 2 *The parity of $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})$ is related to the vector $\boldsymbol{k}$.*

Example 3. We take the "S-box" as an example. Assume that the input multiset $\mathbb{X}$ to the "S-box" has BDPT $\mathcal{D}^{1^4}_{\boldsymbol{k},\boldsymbol{\ell}=(0,1,1,0)}$. In Example 1, we find a division trail for $\mathbb{L}$:$(0,1,1,0)\to(0,1,1,1)$ that cannot be discovered by rule $\mathbb{L}_1$. Thus,

$$\begin{aligned}\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}=(0,1,1,1)}(\boldsymbol{y})&=\bigoplus_{y\in\mathbb{Y}}y_1y_2y_3\\&=\bigoplus_{x\in\mathbb{X}}(x_0x_1x_2\oplus x_1x_2\oplus x_1x_2x_3)\\&=\bigoplus_{x\in\mathbb{X}}\pi_{(1,1,1,0)}(\boldsymbol{x})\oplus\bigoplus_{x\in\mathbb{X}}\pi_{(0,1,1,0)}(\boldsymbol{x})\oplus\bigoplus_{x\in\mathbb{X}}\pi_{(0,1,1,1)}(\boldsymbol{x})\\&=(0\text{ or }\textit{unknown}\text{ depends on }\boldsymbol{k})\oplus1\oplus(0\text{ or }\textit{unknown}\text{ depends on }\boldsymbol{k}).\end{aligned}$$

To illustrate that the value of $\boldsymbol{k}$ affects the parity of $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}=(0,1,1,1)}(\boldsymbol{y})$, we take two specific values of the input vector $\boldsymbol{k}$. If the input vector $\boldsymbol{k}=(1,0,0,1)$,

$$\begin{aligned}\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}=(0,1,1,1)}(\boldsymbol{y})&=\bigoplus_{x\in\mathbb{X}}\pi_{(1,1,1,0)}(\boldsymbol{x})\oplus\bigoplus_{x\in\mathbb{X}}\pi_{(0,1,1,0)}(\boldsymbol{x})\oplus\bigoplus_{x\in\mathbb{X}}\pi_{(0,1,1,1)}(\boldsymbol{x})\\&=0\oplus1\oplus0=1.\end{aligned}$$

If the input vector $\boldsymbol{k}=(1,0,1,0)$,

$$\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}=(0,1,1,1)}(\boldsymbol{y})=\bigoplus_{x\in\mathbb{X}}\pi_{(1,1,1,0)}(\boldsymbol{x})\oplus\bigoplus_{x\in\mathbb{X}}\pi_{(0,1,1,0)}(\boldsymbol{x})\oplus\bigoplus_{x\in\mathbb{X}}\pi_{(0,1,1,1)}(\boldsymbol{x})$$

3.3 New Modeling Method for S-box

Let the set $\mathbb{U} = \{0,1\}^n$ represent all the elements on $\mathbb{F}_2^n$. For any $\boldsymbol{u} \in \mathbb{F}_2^n$, we assume that $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $i \leqslant 2^n$ monomials about the vector $\boldsymbol{x}$, which are $\pi_{\boldsymbol{\varphi}_0}(\boldsymbol{x})$,...,$\pi_{\boldsymbol{\varphi}_{i-1}}(\boldsymbol{x})$, respectively. If $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$, we assume that $\boldsymbol{\ell} = \boldsymbol{\varphi}_j$, where $0 \leqslant j \leqslant i-1$. According to Observation 2 and Theorem 2, we propose Theorem 3, which is a more accurate method to calculate all valid division trails for $\mathbb{L}$ of an S-box.

Theorem 3. *If the input multiset $\mathbb{X}$ to the S-box has BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$ where $\boldsymbol{k} = (k_0,\ldots,k_{n-1}), \boldsymbol{\ell} = (\ell_0,\ldots,\ell_{n-1})$. Let the output multiset $\mathbb{Y}$ of S-box have BDPT $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$, which is calculated by*

$$\mathbb{K} = \{\boldsymbol{u} \in \mathbb{F}_2^n \mid \pi_{\boldsymbol{u}}(\boldsymbol{y}) \textit{ contains any monomial } \pi_{\bar{\boldsymbol{k}}}(\boldsymbol{x}) \textit{ satisfying } \bar{\boldsymbol{k}} \succeq \boldsymbol{k}\}$$
$$\mathbb{L} = \{\boldsymbol{u} \in \mathbb{F}_2^n \mid \pi_{\boldsymbol{u}}(\boldsymbol{y}) \textit{ contains } \pi_{\boldsymbol{\ell}}(\boldsymbol{x}) \textit{ and the input vector } \boldsymbol{k} \in \mathbb{S}_\cap\}.$$

where $\mathbb{S}_\cap = \bigcap_{q=0}^{i-1} \mathbb{S}_q$ and $\mathbb{S}_q = \mathbb{U}\backslash\{\bar{\boldsymbol{\varphi}}_q \mid \boldsymbol{\varphi}_q \succeq \bar{\boldsymbol{\varphi}}_q\}$. If $q \neq j$, the $\mathbb{S}_q$ represents the possible value of the input vector $\boldsymbol{k}$ when $\bigoplus_{x\in\mathbb{X}} \pi_{\boldsymbol{\varphi}_q}(\boldsymbol{x}) = 0$. If $q = j$, the $\mathbb{S}_j$ represents the possible value of the input vector $\boldsymbol{k}$ when $\bigoplus_{x\in\mathbb{X}} \pi_{\boldsymbol{\varphi}_j}(\boldsymbol{x}) = 1$.

The proof is provided in Appendix A. Moreover, Appendix B shows a simple example for Theorem 3. According to Theorem 3, we present a generalized algorithm to calculate all valid division trails for $\mathbb{L}$ of an S-box.

We explain Algorithm 1 line by line:

Line 1 According to input BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$ and Definition 2, the parity of monomial $\pi_{\bar{\boldsymbol{k}}}(\boldsymbol{x})$ with $\bar{\boldsymbol{k}} \succeq \boldsymbol{k}$ over $\mathbb{X}$ is *unknown*, and we store these monomials in $F(\bar{X})$.
Line 2 Initialize $\bar{\mathbb{K}}$,$\bar{\mathbb{L}}$ as empty sets and let the set $\mathbb{U} = \{0,1\}^n$ represent all the elements on $\mathbb{F}_2^n$.
Line 3-6 For any possible $\boldsymbol{u}$, let the set $\mathbb{S}_\cap$ be an empty set, and the set $\mathbb{S}_\cap$ represent the intersection of the possible values of the input vector $\boldsymbol{k}$ when $\bigoplus_{y\in\mathbb{Y}} \pi_{\boldsymbol{u}}(\boldsymbol{y}) = 1$. For i monomials contained in $\pi_{\boldsymbol{u}}(\boldsymbol{y})$, initialize $\mathbb{S}_q$ as an empty set, $0 \leqslant q \leqslant i-1$.
Line 7-8 For any possible $\boldsymbol{u}$, if polynomial $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial in $F(\bar{X})$, the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ over $\mathbb{X}$ is *unknown*. We store all these vectors $\boldsymbol{u}$ in $\bar{\mathbb{K}}$.
Line 9-14 For any possible $\boldsymbol{u}$, if polynomial $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains the monomial $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$ and the input vector $\boldsymbol{k} \in \mathbb{S}_\cap = \bigcap_{q=0}^{i-1} \mathbb{S}_q$, where $\mathbb{S}_q = \mathbb{U}\backslash\{\bar{\boldsymbol{\varphi}}_q \mid \boldsymbol{\varphi}_q \succeq \bar{\boldsymbol{\varphi}}_q\}$ and $i \leqslant 2^n$ represent $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains i monomials, the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ over $\mathbb{X}$ is 1. We store all these vectors $\boldsymbol{u}$ in $\bar{\mathbb{L}}$.
Line 15 ***SizeReduce***$_{\boldsymbol{k}}$ function removes all redundant vectors in $\bar{\mathbb{K}}$. Namely, if there are $\boldsymbol{u}, \boldsymbol{u}' \in \bar{\mathbb{K}}$ satisfying $\boldsymbol{u} \succeq \boldsymbol{u}'$, the vector $\boldsymbol{u}$ can be removed from $\bar{\mathbb{K}}$. Moreover, ***SizeReduce***$_{\boldsymbol{l}}$ function removes all redundant vectors in $\bar{\mathbb{L}}$. If there are $\boldsymbol{\ell}' \in \bar{\mathbb{L}}$ and $\boldsymbol{u} \in \bar{\mathbb{K}}$ satisfying $\boldsymbol{\ell}' \succeq \boldsymbol{u}$, the vector $\boldsymbol{\ell}'$ can be removed from $\bar{\mathbb{L}}$.
Line 16 Return $\mathbb{K}, \mathbb{L}$ as output.

Given an n-bit S-box and its input BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$, Algorithm 1 returns the output BDPT $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$. Thus for any vector $\boldsymbol{k}' \in \mathbb{K}$, $(\boldsymbol{k}, \boldsymbol{k}')$ is a valid division trail

Algorithm 1: Calculating division trails of an S-box

```
Input: The input BDPT of an $n$-bit S-box $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$, where
       $\boldsymbol{k} = (k_0, \ldots, k_{n-1}), \boldsymbol{\ell} = (\ell_0, \ldots, \ell_{n-1})$
Output: The output BDPT $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$
$\bar{\mathbb{S}} = \{\bar{\boldsymbol{k}} \mid \bar{\boldsymbol{k}} \succeq \boldsymbol{k}\}, F(\bar{X}) = \{\pi_{\bar{\boldsymbol{k}}}(\boldsymbol{x}) \mid \bar{\boldsymbol{k}} \in \bar{\mathbb{S}}\}$
$\bar{\mathbb{K}} = \emptyset, \bar{\mathbb{L}} = \emptyset$ and $\mathbb{U} = \{0,1\}^n$
for $\boldsymbol{u} \in (\mathbb{F}_2)^n$ do
    $\mathbb{S}_\cap = \emptyset$
    for $0 \leqslant q \leqslant i-1$ do
        $\mathbb{S}_q = \emptyset$ /* $i \leqslant 2^n$ represents $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $i$ monomials */
    end
    if $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial in $F(\bar{X})$ then
        $\bar{\mathbb{K}} = \bar{\mathbb{K}} \cup \{\boldsymbol{u}\}$
    end
    if $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$ then
        for $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains every monomial $\pi_{\boldsymbol{\varphi}_q}(\boldsymbol{x})$ do
            $\mathbb{S}_q = \mathbb{U} \backslash \{\bar{\boldsymbol{\varphi}}_q \mid \boldsymbol{\varphi}_q \succeq \bar{\boldsymbol{\varphi}}_q\}$
        end
        $\mathbb{S}_\cap = \bigcap_{q=0}^{i-1} \mathbb{S}_q$
        if the input vector $\boldsymbol{k} \in \mathbb{S}_\cap$ then
            $\bar{\mathbb{L}} = \bar{\mathbb{L}} \cup \{\boldsymbol{u}\}$
        end
    end
end
$\mathbb{K} = \boldsymbol{SizeReduce_k}(\bar{\mathbb{K}})$ and $\mathbb{L} = \boldsymbol{SizeReduce_l}(\bar{\mathbb{L}})$
return $\mathbb{K}, \mathbb{L}$
```

for $\mathbb{K}$ of the S-box. Similarly, for any vector $\boldsymbol{\ell}' \in \mathbb{L}$, $(\boldsymbol{\ell}, \boldsymbol{\ell}')$ is a valid division trail for $\mathbb{L}$ of the S-box. Because vector $\boldsymbol{\ell}$ does not affect the propagation of vector $\boldsymbol{k}$ through the S-box, we will obtain a complete list of the division trail for $\mathbb{K}$ by traversing $\boldsymbol{k} \in \mathbb{F}_2^n$, and Xiang *et al.* [26] show the results of PRESENT S-box. Similarly, for a certain input vector $\boldsymbol{\ell} \in \mathbb{F}_2^n$, we will obtain a set of division trails for $\mathbb{L}$ by traversing $\boldsymbol{k} \in \mathbb{F}_2^n$. If we try all the 2^n possible input vector $\boldsymbol{\ell}$, we will obtain a complete list of division trails for $\mathbb{L}$. Table 4 in Appendix C presents a complete list of all the division trails for $\mathbb{L}$ of PRESENT S-box.

Representing the Division Trails of S-box as Linear Inequalities. For an n-bit S-box, each of its valid division trail can be viewed as a $2n$-dimensional vector in $\{0,1\}^{2n}$. Thus, all valid division trails form a subset A of $\{0,1\}^{2n}$. Similar to Model 1, we compute the H-Representation of the convex hull Conv(A) by using the inequality_generator function in SageMath [17]. It will return a set of linear inequalities $\mathcal{L}$ which characterize all valid division trails. However, $\mathcal{L}$ contains too many inequalities, which will make the size of the corresponding MILP problem too large to solve. Generally, the Greedy Algorithm [16] is used to reduce this set $\mathcal{L}$. However, Sasaki *et al.* [12] found that the number of inequalities selected by the greedy algorithm was not the optimal solution, and they proposed a new reduction algorithm. We apply it to reduce this set $\mathcal{L}$, which we showed in Algorithm 2.

Algorithm 2: MILP-Based Select a subset of linear inequalities from $\mathcal{L}$ of an S-box

```
Input: $A$: the set of all division trails of an S-box; $\mathcal{L}$: the set of all inequalities
       in the H-Represen-tation of Conv($A$) with $A$ a subset of $\{0,1\}^{2n}$
Output: $\mathcal{O}$: a set of inequalities seleted from $\mathcal{L}$ whose feasible solutions
        restricted in $\{0,1\}^{2n}$ are exactly $A$
1  $\mathcal{O} = \emptyset$ and $\mathcal{C} = \emptyset$
2  $B = \{0,1\}^{2n} \setminus A = \{b^{(0)}, b^{(1)} \dots, b^{(m-1)}\}$
3  $\mathcal{L} = \{l^{(0)}, l^{(1)} \dots, l^{(t-1)}\}$
4  for $b^{(i)} \in B$ do
       /* $0 \leqslant i \leqslant m-1$                                        */
5      $\mathcal{L}^* = \emptyset$ for $l^{(j)} \in \mathcal{L}$ do
           /* $0 \leqslant j \leqslant t-1$                                    */
6          if the inequality $l^{(j)}$ excludes impossible division trails $b^{(i)}$ then
7              $\mathcal{L}^* = \mathcal{L}^* \cup \{j\}$
8          end
9      end
10     $\mathcal{C}.AddConstraints(\mathcal{L}^*)$
11 end
12 $Obj$ = Minimize($\mathcal{L}$)
13 $\mathcal{M} = ConstructModel(\mathcal{C}, Obj)$
14 $\mathcal{O} = \mathcal{M}$.Optimize()
15 return $\mathcal{O}$
```

We explain Algorithm 2 line by line:

Line 1 Initialize $\mathcal{O}$ and $\mathcal{C}$ as empty sets.

Line 2-3 Let the set $B = \{0,1\}^{2n} \setminus A$ represent all impossible division trails, and the set $\mathcal{L}$ represent t inequalities obtained by using the **inequality_generator** function in SageMath.

Line 4-9 For each impossible division trail $b^{(i)}$, let the set $\mathcal{L}^*$ be an empty set. For any inequality $l^{(j)} \in \mathcal{L}$, if the inequality $l^{(j)}$ excludes impossible division trails $b^{(i)}$, we store the tags j of all these inequalities in $\mathcal{L}^*$. *AddConstraints*() function adds an inequality constraint $\sum_{j\in\mathcal{L}^*} z_j \geq 1$ to the constraint set $\mathcal{C}$ with the binary variables $z_0, z_1, \dots, z_{t-1}$, in which $z_j = 1$ represents that inequality $l^{(j)}$ is chosen and $z_j = 0$ represents that inequality $l^{(j)}$ will not be chosen. The constraint set $\mathcal{C}$ means that every impossible division trail is removed with at least one inequality. Thus, there are m constraints in the constraint set $\mathcal{C}$.

Line 10 Set the objective function *Obj*: Minimize $\sum_{\mathrm{j=0}}^{\mathrm{t-1}} \mathrm{z_j}$.

Line 11 *ConstructModel* function construct a MILP model $\mathcal{M}$ by using the constraint set $\mathcal{C}$ and the objective function *Obj*.

Line 12 The MILP model $\mathcal{M}$ is optimized by the openly available solver Gurobi. It will return a set of inequalities that is composed of all inequalities $l^{(j)} \in \mathcal{L}$ satisfying $z_j = 1$, where $0 \leqslant j \leqslant t-1$.
Line 13 Return $\mathcal{O}$ as output.

We applied Algorithm 2 to a set of linear inequalities generated with SageMath [17] against all valid division trails of various S-boxes. Compared to the previous reduction algorithm based on greedy algorithms, a smaller number of inequalities can be obtained by Algorithm 2. The results are shown in Table 3. To show the effectiveness of Algori-thm 2, the division trails for $\mathbb{K}$ and the the the division trails for $\mathbb{L}$ of the "S-box" is characterized by the 6 and 10 inequalities in Appendix D, respectively.

Table 3: Number of linear inequalities to characterize all valid division trails of an S-box

S-box	The number of division trails	#inequalities SageMath	Previous	Alg. 2
Simon "S-box"	$\lvert\mathbb{K}\rvert = 26$	12	—	6
	$\lvert\mathbb{L}\rvert = 30$	18	—	10
Present S-box	$\lvert\mathbb{K}\rvert = 47$	122	11	8
	$\lvert\mathbb{L}\rvert = 84$	257	23	20
Rectangle S-box	$\lvert\mathbb{K}\rvert = 49$	201	17	12
	$\lvert\mathbb{L}\rvert = 80$	246	19	17
GIFT S-box	$\lvert\mathbb{K}\rvert = 49$	218	15	12
	$\lvert\mathbb{L}\rvert = 64$	236	19	16

3.4 BDPT Modeling of Key-Xor Operation

To model the Key-Xor operation, we propose Propositon 1 according to Definition 4. The proof is provided in Appendix E.

Proposition 1. ***(Cross Propagation)*** *Assume that the input multiset $\mathbb{X}$ to an iterated block cipher has initial BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$, and let $\mathcal{D}^{1^n}_{\mathbb{K}_r,\mathbb{L}_r}$ denote the BDPT of the output multiset after r-round propagation through f_e and f_k, where*

$$\begin{aligned}\mathbb{K}_r &= \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{k}) \cup \underbrace{f_e \circ \cdots \circ f_e}_{r-1} \circ \boldsymbol{f_k}(\mathbb{L}_1) \cup \cdots \cup \boldsymbol{f_k}(\mathbb{L}_r)\\ &= \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{k}) \cup \underbrace{f_e \circ \cdots \circ f_e}_{r-1} \circ \boldsymbol{f_k} \circ f_e(\boldsymbol{\ell}) \cup \cdots \cup \boldsymbol{f_k} \circ \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{\ell}).\\ \mathbb{L}_r &= \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{\ell}).\end{aligned}$$

Then, the set of the last vectors of all r-round division trails for $\mathbb{K}$ that start with the vector $\boldsymbol{k}$, and the set of the last vectors of all r-round BDPT trails which start with the vector $\boldsymbol{\ell}$, is equal to $\mathbb{K}_r$. Furthermore, the set of the last vectors of all r-round division trails for $\mathbb{L}$, which start with the vector $\boldsymbol{\ell}$ is equal to $\mathbb{L}_r$.

According to Definition 4 and Propositon 1, the Key-Xor operations are independent of each other during r-round BDPT propagation. Without loss of generality, we consider the t-th Key-Xor operation, i.e.,

$$\underbrace{f_e \circ \cdots \circ f_e}_{r-t} \circ \boldsymbol{f_k} \circ \underbrace{f_e \circ \cdots \circ f_e}_{t}(\boldsymbol{\ell}) \in \mathbb{K}_r$$

We assume that the input vector $\boldsymbol{\ell}$ is propagated through t-round and get the set $\mathbb{L}_t$. For the t-th Key-Xor operation, the input and output BDPT are $\{\mathbb{L}_t\}$ and $\{\mathbb{K}_t^*, \mathbb{L}_t^*\}$, respectively. Our model uses three n-bit variables $\mathcal{L}_t$, $\mathcal{K}_t^*$, and $\mathcal{L}_t^*$ to denote them, where n is the block size. In many block ciphers, the round key is only XORed with a part of the block. Without loss of generality, we assume that the round key is XORed with the left s $(1 \leqslant s \leqslant n)$ bits. We consider the effect of the t-th Key-Xor operation on $\mathbb{K}_t^*$, $\mathbb{L}_t^*$, respectively.

For the output BDPT $\mathbb{L}_t^*$, according to Rule 1, f_k does not affect the propagation from $\mathbb{L}_t$ to $\mathbb{L}_t^*$. Therefore, the constraint on $\mathcal{L}_t$ and $\mathcal{L}_t^*$ is $\mathcal{L}_t^* = \mathcal{L}_t$.

For the output BDPT $\mathbb{K}_t^*$, according to Rule 1, for every vector $\boldsymbol{\ell}^* \in \mathbb{L}_t$ satisfying $\ell_i^* = 0, 0 \leqslant i \leqslant s-1$, we calculate $\ell_i^* \vee 1$ and add it to the set $\mathbb{K}_t^*$. Thus, the constraint on $\mathcal{L}_t$ and $\mathcal{K}_t^*$ is $\ell_0^t + \ell_1^t + \cdots + \ell_{s-1}^t \leqslant s-1$ and $\mathcal{K}_t^* \& \mathcal{L}_t = \mathcal{L}_t$.

We only considered the t-th Key-Xor operation f_k instead of considering a complete BDPT propagation chain, i.e., $\boldsymbol{\ell} \to \mathbb{K}_t^* \to \mathbb{K}_r$. We give Algorithm 3 to characterize the BDPT propagation chain based on MILP. In Algorithm 3, we construct a constraint set $\mathcal{C}_t$ using a linear inequality system, which accurately characterizes the BDPT trails of the set generated by the t-th Key-Xor operation.

We explain Algorithm 3 line by line:

Line 1 Initialize $\mathcal{P}$ as empty sets.

Line 2-3 In the MILP model, each n-bit variable represents the BDPT $\mathbb{K}$ or $\mathbb{L}$. Therefore, we allocate two sets of n-bit variables $\mathcal{K}_i^*$ and $\mathcal{L}_i$ to represent the sets $\mathbb{K}_i^*$ and $\mathbb{L}_i$, where n is the block size and $0 \leqslant i \leqslant r$.

Line 4 Set the objective function *Obj*: Minimize $\sum_{\mathrm{i}=0}^{\mathrm{n}-1} \mathrm{k}_{\mathrm{i}}^{\mathrm{r}^*}$, where $k_i^{r^*}$ represents the i-th bit of the n-bit variables $\mathcal{K}_r^*$.

Line 5-8 For the t-th Key-Xor operation $(1 \leqslant t \leqslant r-1)$, let the constraint set $\mathcal{C}_t$ be an empty set. The former t-round BDPT propagation is characterized by linear inequalities constraint set $\mathcal{O}_l(\mathbb{L})$, and the inequality constraints of each round are added to the constraint set $\mathcal{C}_t$.

Line 9-10 For the t-th Key-Xor operation, we add two new constraints $\ell_0^t + \ell_1^t + \cdots + \ell_{s-1}^t \leqslant s-1$ and $\mathcal{K}_t^* \& \mathcal{L}_t = \mathcal{L}_t$ to the constraint set $\mathcal{C}_t$. These two constraints can be used to obtain the vectors that can be added to the set $\mathcal{K}_t^*$.

Line 11-12 The remaining $(r-t)$-round BDPT propagation is characterized by linear inequalities constraint set $\mathcal{O}_k(\mathbb{K})$, and the inequality constraints of each

Algorithm 3: MILP-Based Characterize the Propagation Rule of Key-XOR Operations

```
Input: The initial input BDPT of an $n$-bit iterated block cipher
       $\mathcal{D}^{1^n}_{\mathbb{K}_0=\{\boldsymbol{k}\},\mathbb{L}_0=\{\boldsymbol{\ell}\}}$; $\mathcal{O}_k(\mathbb{K})$: a constraint set of linear inequalities whose
       feasible solutions are all valid division trails for $\mathbb{K}$ of the round
       function; $\mathcal{O}_l(\mathbb{L})$: a constraint set of linear inequalities whose feasible
       solutions are all valid division trails for $\mathbb{L}$ of the round function; An
       $n$-bit vector $\boldsymbol{m}$ representing the Key-XOR operation
Output: $\mathcal{P}$: A collection of MILP models with constraints for Key-XOR
        Operations
1  $\mathcal{P} = \emptyset$
2  Allocate $n$-bit variables $\mathcal{K}_i^*$ to denote $\mathbb{K}_i^*$, where $(i = 0, 1, \dots, r)$
3  Allocate $n$-bit variables $\mathcal{L}_i$ to denote $\mathbb{L}_i$, where $(i = 0, 1, \dots, r)$
4  $Obj$ = Minimize($\{\mathrm{k}_0^{\mathrm{r}^*} + \cdots + \mathrm{k}_{\mathrm{n}-1}^{\mathrm{r}^*}\}$)
5  for $(t = 1; t < r; t++)$ do
6      $\mathcal{C}_t = \emptyset$ for $(i = 0; i < t; i++)$ do
7          $\mathcal{C}_t \leftarrow \mathcal{O}_l(\mathcal{L}_i, \mathcal{L}_{i+1})$
8      end
9      $\mathcal{C}_t \leftarrow \{\ell_0^t + \ell_1^t + \cdots + \ell_{s-1}^t \leqslant s - 1\}$
10     $\mathcal{C}_t \leftarrow \mathcal{K}_t^* \& \mathcal{L}_t = \mathcal{L}_t$ for $(j = t; j < r; j++)$ do
11         $\mathcal{C}_t \leftarrow \mathcal{O}_k(\mathcal{K}_j^*, \mathcal{K}_{j+1}^*)$
12     end
13     $\mathcal{M}_t$ = ConstructModel($\mathcal{C}_t$, $Obj$)
14     $\mathcal{P}$ = addModel($\mathcal{M}_t$)
15 end
16 return $\mathcal{P}$
```

round are added to the constraint set $\mathcal{C}_t$.
Line 13 *ConstructModel* function constructs a MILP model $\mathcal{M}_t$ using the constraint set $\mathcal{C}_t$ and the objective function *Obj*.
Line 14 We add the MILP model $\mathcal{M}_t$, which characterizes the BDPT propagation of the t-th Key-Xor operation, to the model set $\mathcal{P}$.
Line 15 Return $\mathcal{P}$ as output.

Algorithm 3 constructs a MILP model for the former $r-1$ Key-Xor operations of an r-round block cipher, which characterizes all division trails of a complete BDPT propagation chain, i.e., $\boldsymbol{\ell} \to \mathbb{K}_t^* \to \mathbb{K}_r$, $1 \leqslant t \leqslant r-1$. By solving each model $\mathcal{M}_t$ in the model set $\mathcal{P}$ separately, we can obtain the set of the last vectors of all r-round BDPT trails, which start with the vector $\boldsymbol{\ell}$.

Remark 2. The r-th Key-Xor operation is ignored in our Algorithm 3, since it does not produce any unit vector for $\mathbb{K}_r$ in our model.

4 Initial, Stopping Rule and Search Algorithm

In this section, we first study the initial BDPT and stopping rule to use when searching for integral distinguishers based on BDPT. According to Definition 4 and Proposition 1, for each model which starts with the input vector $\boldsymbol{k}$ or $\boldsymbol{\ell}$, we convert the stopping rule into an objective function of the MILP model. At last, we propose an algorithm to search integral distinguishers based on BDPT given the initial BDPT $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$ for an n-bit block cipher.

4.1 Initial BDPT

In [20], Todo and Morii set the initial BDPT as $(\boldsymbol{k} = \mathbf{1}, \boldsymbol{\ell} = \texttt{7fffffff})$ to search the BDPT of Simon32, where the active bits of the vector $\boldsymbol{\ell}$ are set as 1, and the constant bit is set to 0. Similarly, we assume that $((k^0_0, k^0_1, \ldots, k^0_{n-1}), (\ell^0_0, \ell^0_1, \ldots, \ell^0_{n-1}))$ denotes the initial BDPT, where n is the block size. The constraints on k^0_i and ℓ^0_i are

$$k^0_i = 1, \text{ for } i = 0, 1, 2, \ldots, n-1$$
$$\ell^0_i = \begin{cases} 1, & \text{if the } i\text{-th bit is active,} \\ 0, & \text{otherwise.} \end{cases}$$

4.2 Stopping Rule

Stopping Rule 1 (for a single model)**.** We consider the stopping rule of the r-th round output sets $\mathbb{K}_r$ and $\mathbb{L}_r$, respectively.

According to Proposition 1, the set $\mathbb{K}_r$ is composed of all r-round division trails for $\mathbb{K}$, which start with the vector $\boldsymbol{k}$, and all r-round BDPT trails produced by the Key-XOR operations, which start with the vector $\boldsymbol{\ell}$. In the BDPT propagation, we note that only the vector $\mathbf{1}$ can propagate to vector $\mathbf{1}$. Thus, if the given initial BDPT is $\mathcal{D}^{1^n}_{\boldsymbol{k},\boldsymbol{\ell}}$ with $\boldsymbol{k} = \mathbf{1}$, the r-round division trails for $\mathbb{K}$ can be ignored because it does not produce any unit vector for $\mathbb{K}_r$, i.e.,

$$\mathbb{K}_r \setminus \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{k})$$

Therefore, the model set $\mathcal{P}$ constructed by Algorithm 3 can accurately describe the vectors in $\mathbb{K}_r$. For each model $\mathcal{M}_t$ in the model set $\mathcal{P}$, let $(\ell^0_0, \ell^0_1, \ldots, \ell^0_{n-1}) \xrightarrow{f_e} \cdots \xrightarrow{f_e} (\ell^t_0, \ell^t_1, \ldots, \ell^t_{n-1}) \xrightarrow{f_k} (k^{t^*}_0, k^{t^*}_1, \ldots, k^{t^*}_{n-1}) \xrightarrow{f_e} \cdots \xrightarrow{f_e} (k^{r^*}_0, k^{r^*}_1, \ldots, k^{r^*}_{n-1})$ denote an r-round BDPT trail for the t-th Key-Xor operation. The objective function can be set as follows:

$$Obj : \text{Minimize}\{\mathrm{k}^{\mathrm{r}^*}_0 + \mathrm{k}^{\mathrm{r}^*}_1 + \cdots + \mathrm{k}^{\mathrm{r}^*}_{\mathrm{n}-1}\}$$

According to Proposition 1, the set $\mathbb{L}_r$ is composed of all r-round division trails for $\mathbb{L}$, which start with the vector $\boldsymbol{\ell}$. Let $(\ell^0_0, \ell^0_1, \ldots, \ell^0_{n-1}) \xrightarrow{f_e} \cdots \xrightarrow{f_e}$

$(\ell_0^{\mathrm{r}}, \ell_1^{\mathrm{r}}, \ldots, \ell_{n-1}^{\mathrm{r}})$ denote an r-round division trail for $\mathbb{L}$. Thus, we can set the objective function as :

$$Obj : \text{Minimize}\{\ell_0^{\mathrm{r}} + \ell_1^{\mathrm{r}} + \cdots + \ell_{\mathrm{n-1}}^{\mathrm{r}}\}$$

Stopping Rule 2 (for the overall model)**.** Our overall model is divided into model sets $\mathcal{P}$ and model $\mathcal{M}_L$, which describe all vectors in sets $\mathbb{K}_r$ and $\mathbb{L}_r$, respectively. Our overall MILP model only focuses on the parity of one output bit. Without loss of generality, we consider the q-th output bit. For each model $\mathcal{M}_t$ in the model set $\mathcal{P}$, we can use the solver Gurobi to determine whether the MILP model $\mathcal{M}_t$ has feasible solution $\mathcal{K}_q = (k_0^{r^*}, \ldots, k_{n-1}^{r^*})$, where

$$k_i^{r^*} = \begin{cases} 1, & \text{if } i = q, \\ 0, & \text{otherwise.} \end{cases}$$

If any $\mathcal{M}_t$ in the model set $\mathcal{P}$ has feasible solution $\mathcal{K}_q$, there is a unit vector $\boldsymbol{e}_q \in \mathbb{K}_r$, and further the q-th output bit is *unknown*.

If there is not feasible solution $\mathcal{K}_q$ of model set $\mathcal{P}$ and the number of solutions $\mathcal{L}_q = (\ell_0^{r^*}, \ldots, \ell_{n-1}^{r^*})$ of model $\mathcal{M}_L$ is odd, where

$$\ell_i^{r^*} = \begin{cases} 1, & \text{if } i = q, \\ 0, & \text{otherwise.} \end{cases}$$

there is a unit vector $\boldsymbol{e}_q \in \mathbb{L}_r$, and further the parity of the q-th output bit is 1. Otherwise the q-th output bit is 0.

4.3 Search Algorithm

We present the automated search integral distinguishers algorithm, which decides the parity of the q-th output bit with the given initial BDPT $\mathcal{D}^{1^n}_{\mathbb{K}_0=\{\boldsymbol{k}\},\mathbb{L}_0=\{\boldsymbol{\ell}\}}$ for an n-bit block cipher. Firstly, we allocate all round variables and auxiliary variables. Secondly, we construct a MILP model $\mathcal{M}_L$ that describes all r-round division trails for $\mathbb{L}$ and calls Algorithm 3 to save the model set $\mathcal{P}$. Finally, according to the initial and stopping rules, we can obtain the parity of the q-th output bit based on BDPT. We illustrate the whole framework in Algorithm 4.

5 Applications

In this section, we apply our algorithm to Simon, Simeck, Present, Rectangle and GIFT-64 block ciphers. The results are listed in Table 1[3]. In addition, for the integral distinguishers, the label "a" represents the active bit, "c" represents the constant bit, "?" represents *unknown*, "b" represents the balanced bit whose sum is 0 or 1, "0" represents the balanced bit whose sum is 0, "1" represents the balanced bit whose sum is 1.

[3] All the experiments are conducted on the following platform: Xeon(R) CPU E5-2620 v3 @2.40 GHz, 128 G RAM, and the optimizer used to solve MILP models is Gurobi 9.0.3. The source code is available on GitHub https://github.com/CryptAnalystDesigner/MILP_Models_For_BDPT.git

Algorithm 4: Automated search r-round integral distinguishers

```
Input: The cipher $E$, the initial input BDPT of the $n$-bit block cipher
       $\mathcal{D}^{1^n}_{\mathbb{K}_0=\{\boldsymbol{k}\},\mathbb{L}_0=\{\boldsymbol{\ell}\}}$, and the number $q$
Output: The balanced information of the $q$-th output bit based on BDPT
1  Allocate all the variables denoting the input and output BDPT
2  $Obj$ = Minimize($\{\ell^r_0 + \cdots + \ell^r_{n-1}\}$)
3  $\mathcal{M}_L = ConstructModel(\{\mathcal{O}_l(\mathbb{L}) \times r\}, Obj)$
4  Call Algorithm 3 and save the model set $\mathcal{P}$
5  for every model $\mathcal{M}_t \in \mathcal{P}$ do
6      flag = 0
7      if the MILP model $\mathcal{M}_t$ has solutions $\mathcal{K}_q$ then
8          flag = flag + 1
9      end
10 end
11 if flag $\geqslant$ 1 then
12     return unknown
13 end
14 else
15     if the number of solutions $\mathcal{L}_q$ of model $\mathcal{M}_L$ is odd then
16         return 1
17     end
18     else
19         return 0
20     end
21 end
```

5.1 Applications to SIMON and SIMECK

SIMON [2] is a family of lightweight block ciphers published by the U.S. National Security Agency (NSA) in 2013. SIMON adopts the Feistel structure, and has a very compact round function that only involves bitwise And, Xor, and Circular shift operations. The structure of one round SIMON encryption is depicted in Fig. 2, where S^i denotes the left circular shift by i bits. The core operation of the round function and "S-box" are represented in Fig. 1. SIMECK [27] is a family of lightweight block ciphers proposed at CHES 2015, and its round function is very similar to that of SIMON except for the rotation constants. We only introduce the automatic search model for SIMON$2n$ based on BDPT.

In Algorithm 3, we introduce two constraint sets, $\mathcal{O}_k(\mathbb{K})$ and $\mathcal{O}_l(\mathbb{L})$, which describe division trails for $\mathbb{K}$ and division trails for $\mathbb{L}$ of the round function, respectively. For 1-round description of SIMON$2n$, they are similar except for the characterization of "S-box". Therefore, we only introduce 1-round description for $\mathcal{O}_l(\mathbb{L})$ of SIMON$2n$.

1-round Description for $\mathcal{O}_l(\mathbb{L})$ of SIMON$2n$. Denote one round division trail for $\mathbb{L}$ of SIMON$2n$ by $(a^{i,0}_0, \ldots, a^{i,0}_{n-1}, b^{i,0}_0, \ldots, b^{i,0}_{n-1}) \rightarrow (a^{i+1,0}_0, \ldots, a^{i+1,0}_{n-1}, b^{i+1,0}_0, \ldots, b^{i+1,0}_{n-1})$. In our model, we divide the round function of SIMON$2n$ into n "S-

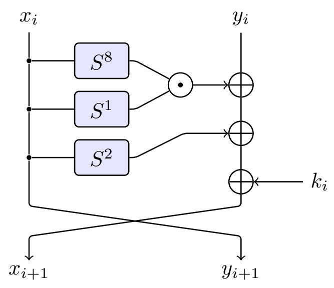

Fig. 2: Round function of SIMON

box" operations and a Key-Xor operation. We first consider the "S-box" operation and assume that the input and output of the j-th "S-box" are denoted as $(a_0^{i,j-1},\dots,a_{n-1}^{i,j-1},b_0^{i,j-1},\dots,b_{n-1}^{i,j-1})$ and $(a_0^{i,j},\dots,a_{n-1}^{i,j},b_0^{i,j},\dots,b_{n-1}^{i,j})$, respectively. According to Fig. 2, the input set that actually participates in the j-th "S-box" operation is $\{a_{(1-j) \bmod n}^{i,j-1}, a_{(8-j) \bmod n}^{i,j-1}, a_{(2-j) \bmod n}^{i,j-1}, b_{(n-j) \bmod n}^{i,j-1}\}$, and the corre-sponding output set is $\{a_{(1-j) \bmod n}^{i,j}, a_{(8-j) \bmod n}^{i,j}, a_{(2-j) \bmod n}^{i,j}, b_{(n-j) \bmod n}^{i,j}\}$. Appendix D shows the 10 inequalities for the division trails for $\mathbb{L}$ of the "S-box", and thus the 4-bit input and output can be modeled by the 10 inequalities, which are denoted by $\mathcal{L}_1$. For the rest $(2n-4)$ bits, which remains unchanged, we have

$$\mathcal{L}_2 : \begin{cases} a_m^{i,j} & = a_m^{i,j-1} \quad m \in \{0,1,\dots,n-1\} \setminus \{(1-j),(8-j),(2-j)\} \bmod n \\ b_m^{i,j} & = b_m^{i,j-1} \quad m \in \{0,1,\dots,n-1\} \setminus \{(n-j)\} \bmod n \end{cases}$$

Therefore, we get an accurate description $\{\mathcal{L}_1,\mathcal{L}_2\}$ of the division trails for $\mathbb{L}$ of the j-th "S-box". By repeating this procedure n times, we can get a set of linear inequalities for the n "S-box" operations.

At last, we consider the Key-Xor operation, and its input and output are denoted as $(a_0^{i,n},\dots,a_{n-1}^{i,n},b_0^{i,n},\dots,b_{n-1}^{i,n})$ and $(a_0^{i+1,0},\dots,a_{n-1}^{i+1,0},b_0^{i+1,0},\dots,b_{n-1}^{i+1,0})$, respectively. According to Rule 1, the Key-Xor operation does not affect the propagation of division trails for $\mathbb{L}$. Therefore, the Key-Xor operation in SIMON$2n$ can be modeled by the following inequalities:

$$\mathcal{L}_3 : \begin{cases} a_m^{i+1,0} = b_m^{i,n} \quad m \in \{0,1,\dots,n-1\} \\ b_m^{i+1,0} = a_m^{i,n} \quad m \in \{0,1,\dots,n-1\} \end{cases}$$

So far, we have modeled all operations used in the round function of SIMON$2n$ and get an accurate description $\{\{\mathcal{L}_1,\mathcal{L}_2\}\times n, \mathcal{L}_3\}$ of 1-round division trails for $\mathbb{L}$, i.e., $\mathcal{O}_l(\mathbb{L})$. Similarly, we can get the 1-round description for $\mathcal{O}_l(\mathbb{K})$ of SIMON$2n$.

Integral Distinguishers. We use Algorithm 3 and 4 to search the integral distinguishers of the SIMON and SIMECK family based on BDPT.

1. For SIMON64, we can find a 17-round integral distinguisher with 27 balanced bits, which has four more bits than the previous longest distinguisher [24].

For SIMON32, 48, 96, 128, the distinguishers we find are in accordance with the previous longest distinguishers found in [24].
2. For SIMECK32, 48, 64, the distinguishers we find are in accordance with the previous longest distinguishers found in [26,24].

The detailed integral distinguishers of SIMON and SIMECK are listed in Appendix F. To further prove the accuracy of our automatic search model, we apply Algorithm 3 and 4 to search integral distinguishers of SIMON(102) [10] based on BDPT, and show the detailed results in Appendix F.12.

5.2 Applications to PRESENT, RECTANGLE and GIFT-64

PRESENT [3] has an SPN structure and uses 80- and 128-bit keys with 64-bit blocks through 31 rounds, of which the linear layers are bit permutations. Fig. 3 illustrates the one-round structure of PRESENT. RECTANGLE [28] is a bit-slice lightweight block cipher proposed in 2015, and its structure is very similar to PRESENT. By revisiting the design strategy of PRESENT, Banik *et al.* propose a new lightweight block cipher GIFT [1], which corrects the well-known weakness of PRESENT with regards to linear hulls.

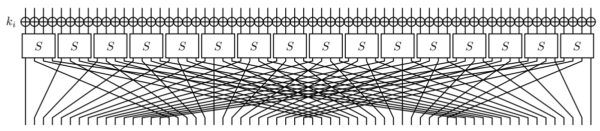

Fig. 3: One-round SPN Structure of PRESENT

Integral Distinguishers. We have shown how to model the $\mathcal{O}_l(\mathbb{K})$ and $\mathcal{O}_l(\mathbb{L})$ in SIMON. Thus, we omit the details for these three ciphers due to the limit of space. We apply Algorithm 3 and 4 to search for integral distinguishers of PRESENT, RECTANGLE and GIFT-64 based on BDPT.

1. For PRESENT, the distinguishers we find are in accordance with the previous longest distinguish-ers[24].
2. For RECTANGLE, we find a 10-round integral distinguisher with 9 balanced bits, which has eight more bits than the previous best integral distinguisher in [11].
3. For GIFT-64, we find a 11-round integral distinguisher, which is one more round than the previous best results [1].

The detailed integral distinguishers of PRESENT, RECTANGLE and GIFT-64 are listed in Appendix F.

6 Conclusions

In this paper, we proposed an automatic search model to search integral distinguishers for block ciphers based on BDPT. We first proposed an effective algorithm that can more accurately obtain the division trails for $\mathbb{K}$ and division trails for $\mathbb{L}$ of the S-box according to its ANF directly. Then we model each Key-Xor operation based on the MILP technique for the first time. By solving these MILP models, we could accurately characterize the Key-Xor operation. Finally, by selecting appropriate initial and stopping rules, we can construct an automatic search model that greatly improves the efficiency for block ciphers with little loss of accuracy, based on which we present an algorithm to estimate whether the q-th output bit is balanced.

We apply our automatic search model to search for integral distinguishers of some block ciphers. For SIMON64, RECTANGLE and GIFT-64, we obtained much better integral distinguishers than previous best results in the open literature. For other block ciphers, our results are in accordance with the previous longest distinguishers. Moreover, compared to the previous methods, our automatic search model reduces the time complexity of searching integral distinguishers for the block ciphers.

References

1. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: CHES. Lecture Notes in Computer Science, vol. 10529, pp. 321–345. Springer (2017)
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: DAC. pp. 175:1–175:6 (2015)
3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: CHES. Lecture Notes in Computer Science, vol. 4727, pp. 450–466 (2007)
4. Boura, C., Canteaut, A.: Another view of the division property. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 654–682 (2016)
5. Funabiki, Y., Todo, Y., Isobe, T., Morii, M.: Improved integral attack on HIGHT. In: ACISP (1). Lecture Notes in Computer Science, vol. 10342, pp. 363–383. Springer (2017)
6. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset - improved cube attacks against trivium and grain-128aead. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12105, pp. 466–495. Springer (2020)
7. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: Revisiting degree evaluations, cube attacks, and key-independent sums. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 12491, pp. 446–476. Springer (2020)

8. Hu, K., Wang, M.: Automatic search for a variant of division property using three subsets. In: CT-RSA. Lecture Notes in Computer Science, vol. 11405, pp. 412–432 (2019)
9. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2365, pp. 112–127 (2002)
10. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 9215, pp. 161–185. Springer (2015)
11. Lambin, B., Derbez, P., Fouque, P.: Linearly equivalent s-boxes and the division property. Des. Codes Cryptogr. **88**(10), 2207–2231 (2020)
12. Sasaki, Y., Todo, Y.: New algorithm for modeling s-box in MILP based differential and division trail search. In: SECITC. Lecture Notes in Computer Science, vol. 10543, pp. 150–165 (2017)
13. Sun, B., Hai, X., Zhang, W., Cheng, L., Yang, Z.: New observation on division property. Sci. China Inf. Sci. **60**(9), 98102 (2017)
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.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 128–157 (2017)
15. Sun, L., Wang, W., Wang, M.: Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Inf. Secur. **14**(1), 12–20 (2020)
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.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 158–178 (2014)
17. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.1.0) (2020), `https://www.sagemath.org`
18. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 413–432 (2015)
19. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 287–314 (2015)
20. Todo, Y., Morii, M.: Bit-based division property and application to simon family. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 357–377 (2016)
21. Wang, Q., Grassi, L., Rechberger, C.: Zero-sum partitions of PHOTON permutations. In: CT-RSA. Lecture Notes in Computer Science, vol. 10808, pp. 279–299 (2018)

22. Wang, Q., Liu, Z., Varici, K., Sasaki, Y., Rijmen, V., Todo, Y.: Cryptanalysis of reduced-round SIMON32 and SIMON48. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 8885, pp. 143–160 (2014)
23. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: MILP method of searching integral distinguishers based on division property using three subsets. IACR Cryptol. ePrint Arch. p. 1186 (2018)
24. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Milp-aided method of searching division property using three subsets and applications. In: ASIACRYPT (3). Lecture Notes in Computer Science, vol. 11923, pp. 398–427 (2019)
25. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: Exploring secret keys in searching integral distinguishers based on division property. IACR Trans. Symmetric Cryptol. **2020**(3), 288–304 (2020)
26. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 648–678 (2016)
27. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The simeck family of lightweight block ciphers. In: CHES. Lecture Notes in Computer Science, vol. 9293, pp. 307–329 (2015)
28. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)

A Proof of Theorem 3

According to Theorem 2, for the input vector $\boldsymbol{\ell}$, calculating $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})$ for all $\boldsymbol{u}\in\mathbb{F}_2^n$, if $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1$, we obtain a valid division trail for $\mathbb{L}$:$(\boldsymbol{\ell},\boldsymbol{u})$. In order to obtain the condition that $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1$ is established, we first study the $\pi_{\boldsymbol{u}}(\boldsymbol{y})$, which is a polynomial representation about $\boldsymbol{x}$.

$$\begin{aligned}\pi_{\boldsymbol{u}}(\boldsymbol{y}) &= \prod_{i=0}^{n-1} y[i]^{u[i]},\ \text{where } y[i]=f_i\left(x_0,\dots,x_{n-1}\right)\\ &= \prod_{i=0}^{n-1} f_i\left(x_0,\dots,x_{n-1}\right)^{u[i]}\\ &= \prod_{i=0}^{n-1}\left(\bigoplus_{\boldsymbol{v}\in\mathbb{F}_2^n} a_{\boldsymbol{v}}^{f_i}\pi_{\boldsymbol{v}}(\boldsymbol{x})\right)^{u[i]}\end{aligned}$$

Let

$$g(\boldsymbol{x})=\prod_{i=0}^{n-1}\left(\bigoplus_{\boldsymbol{v}\in\mathbb{F}_2^n} a_{\boldsymbol{v}}^{f_i}\pi_{\boldsymbol{v}}(\boldsymbol{x})\right)^{u[i]}=\bigoplus_{\boldsymbol{\varphi}\in\mathbb{F}_2^n} a_{\boldsymbol{\varphi}}^{g}\pi_{\boldsymbol{\varphi}}(\boldsymbol{x})$$

where $a_{\boldsymbol{v}}^{f_i}\in\mathbb{F}_2$ is a constant value depending on f_i and $\boldsymbol{v}$, $a_{\boldsymbol{\varphi}}^{g}\in\mathbb{F}_2$ also is a constant value depending on g and $\boldsymbol{\varphi}$. Thus,

$$\pi_{\boldsymbol{u}}(\boldsymbol{y})=\bigoplus_{\boldsymbol{\varphi}\in\mathbb{F}_2^n} a_{\boldsymbol{\varphi}}^{g}\pi_{\boldsymbol{\varphi}}(\boldsymbol{x})$$

we assume the set $\mathbb{O}=\{\boldsymbol{\varphi}\in\mathbb{F}_2^n\mid a_{\boldsymbol{\varphi}}^{g}=1\}$, we have

$$\begin{aligned}\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y}) &= \bigoplus_{x\in\mathbb{X}}\bigoplus_{\boldsymbol{\varphi}\in\mathbb{F}_2^n} a_{\boldsymbol{\varphi}}^{g}\pi_{\boldsymbol{\varphi}}(\boldsymbol{x})\\ &= \bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_0}(\boldsymbol{x})\oplus\cdots\oplus\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_{|\mathbb{O}|-1}}(\boldsymbol{x})\end{aligned}$$

where the $\boldsymbol{\varphi}_i\in\mathbb{O}$, $0\le i\le|\mathbb{O}|-1$. According to Definition 2, the parity of $\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_i}(\boldsymbol{x})$ is *unknown* or 1 or 0, where $0\le i\le|\mathbb{O}|-1$. If $\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1$, i.e.,

$$\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_0}(\boldsymbol{x})\oplus\cdots\oplus\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_{|\mathbb{O}|-1}}(\boldsymbol{x})=1$$

Obviously, the input vector $\boldsymbol{\ell}\in\mathbb{O}$ and $\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\ell}}(\boldsymbol{x})=1$. Let $\boldsymbol{\ell}=\boldsymbol{\varphi}_j$, where $j\in\{i\mid 0\le i\le|\mathbb{O}|-1\}$. For any $\boldsymbol{\varphi}_i\in\mathbb{O}\backslash\{\boldsymbol{\varphi}_j\}$, $\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_i}(\boldsymbol{x})=0$. Thus,

$$\begin{aligned}&\bigoplus_{y\in\mathbb{Y}}\pi_{\boldsymbol{u}}(\boldsymbol{y})=1\Leftrightarrow\ \text{for every } \boldsymbol{\varphi}_i\in\mathbb{O}\backslash\{\boldsymbol{\varphi}_j\},\\ &\bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\varphi}_i}(\boldsymbol{x})=0 \text{ and } \bigoplus_{x\in\mathbb{X}}\pi_{\boldsymbol{\ell}=\boldsymbol{\varphi}_j}(\boldsymbol{x})=1\end{aligned}$$

According to Definition 2, for every $\bigoplus_{x \in \mathbb{X}} \pi_{\boldsymbol{\varphi}_i}(\boldsymbol{x}) = 0$ and $\bigoplus_{x \in \mathbb{X}} \pi_{\boldsymbol{\varphi}_j}(\boldsymbol{x}) = 1$, the possible value of the input vector $\boldsymbol{k}$ is $\mathbb{S}_i = \mathbb{U} \backslash \{\bar{\varphi}_i \mid \varphi_i \succeq \bar{\varphi}_i\}$, where $0 \leq i \leq |\mathbb{O}| - 1$(Because $j \in \{i \mid 0 \leq i \leq |\mathbb{O}| - 1\}$, we have $0 \leq i \leq |\mathbb{O}| - 1$). Therefore, for all $0 \leq i \leq |\mathbb{O}| - 1$, we have

$$\mathbb{S}_{\cap} = \mathbb{S}_0 \cap \cdots \cap \mathbb{S}_{|\mathbb{O}|-1}$$

The set $\mathbb{S}_{\cap}$ represents the intersection of the possible values of $\boldsymbol{k}$ when the $\bigoplus_{x \in \mathbb{X}} \pi_{\boldsymbol{\varphi}_i}(\boldsymbol{x}) = 0$ and $\bigoplus_{x \in \mathbb{X}} \pi_{\boldsymbol{\varphi}_j}(\boldsymbol{x}) = 1$. If the input vector $\boldsymbol{k} \in \mathbb{S}_{\cap}$, it means that the input vector $\boldsymbol{k}$ such that (A) holds. So (A) is equivalent to $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$ and $\boldsymbol{k} \in \mathbb{S}_{\cap}$.

B Example for Theorem 3

To help readers understand the Theorem 3, we show a simple example of the core operation of Simon family. We first treat the core operation of Simon family as an "S-box", which is a 4 × 4 S-box, and its input and output are shown in Fig. 1. Let the input multiset $\mathbb{X}$ to the "S-box" have BDPT $\mathcal{D}^{1^4}_{\boldsymbol{k},\boldsymbol{\ell}=(0,0,1,0)}$. For every $\boldsymbol{u} \in \mathbb{F}_2^4$, we calculate the $\pi_{\boldsymbol{u}}(\boldsymbol{y}) = \prod_{i=0}^{3} f_i(x_0, x_1, x_2, x_3)^{u[i]}$ as shown in Table 4. According to Theorem 3 and Table 4, if the $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{\ell}}(\boldsymbol{x})$, the vector $\boldsymbol{u} \in \{(0,0,0,1),(0,0,1,0),(0,0,1,1)\}$. We consider the following three cases respectively.

1. When the vector $\boldsymbol{u} = (0,0,0,1)$, we calculate:

$$\begin{aligned}
&\bigoplus_{y \in \mathbb{Y}} \pi_{\boldsymbol{u}=(0,0,0,1)}(\boldsymbol{y}) \\
=&\bigoplus_{y \in \mathbb{Y}} y_3 \\
=&\bigoplus_{x \in \mathbb{X}} (x_0 x_1 \oplus x_2 \oplus x_3) \\
=&\bigoplus_{x \in \mathbb{X}} \pi_{(1,1,0,0)}(\boldsymbol{x}) \oplus \bigoplus_{x \in \mathbb{X}} \pi_{(0,0,1,0)}(\boldsymbol{x}) \oplus \bigoplus_{x \in \mathbb{X}} \pi_{(0,0,0,1)}(\boldsymbol{x}) \\
=&(0 \text{ or } \textit{unknown} \text{ Depends on } \boldsymbol{k}) \oplus 1 \oplus (0 \text{ or } \textit{unknown} \text{ Depends on } \boldsymbol{k})
\end{aligned}$$

   if $\bigoplus_{x \in \mathbb{X}} \pi_{(1,1,0,0)}(\boldsymbol{x}) = 0$, the vector $\boldsymbol{k} \in \mathbb{S}_0 = \mathbb{U} \backslash \{\bar{\varphi}_0 \mid \varphi_0 = (1,1,0,0) \succeq \bar{\varphi}_0\}$. Similarly, we can get $\boldsymbol{k} \in \mathbb{S}_1 = \mathbb{U} \backslash \{\bar{\varphi}_1 \mid \varphi_1 = (0,0,1,0) \succeq \bar{\varphi}_1\}$ and $\boldsymbol{k} \in \mathbb{S}_2 = \mathbb{U} \backslash \{\bar{\varphi}_2 \mid \varphi_2 = (0,0,0,1) \succeq \bar{\varphi}_2\}$. We calculate the intersection of the values of the vector $\boldsymbol{k}$, i.e.,

$$\begin{aligned}
&\mathbb{S}_{\cap} \\
=&\mathbb{S}_0 \cap \mathbb{S}_1 \cap \mathbb{S}_2 \\
=&\{(0,0,1,1),(0,1,0,1),(0,1,1,0),(0,1,1,1),(1,0,0,1), \\
&(1,0,1,0),(1,0,1,1),(1,1,0,1),(1,1,1,0),(1,1,1,1)\}
\end{aligned}$$

Thus, $(0,0,1,0) \to (0,0,0,1)$ is an valid division trail for $\mathbb{L}$ when the input vector $\boldsymbol{k} \in \mathbb{S}_\cap$.

2. When the vector $\boldsymbol{u} = (0,0,1,0)$, we calculate:

$$\begin{aligned}\bigoplus_{y\in\mathbb{Y}} \pi_{\boldsymbol{u}=(0,0,1,0)}(\boldsymbol{y}) &= \bigoplus_{y\in\mathbb{Y}} y_2 \\ &= \bigoplus_{x\in\mathbb{X}} x_2 \\ &= \bigoplus_{x\in\mathbb{X}} \pi_{(0,0,1,0)}(\boldsymbol{x}) \\ &= 1\end{aligned}$$

if $\bigoplus_{x\in\mathbb{X}} \pi_{(0,0,1,0)}(\boldsymbol{x}) = 1$, the vector $\boldsymbol{k} \in \mathbb{S}_0 = \mathbb{U}\backslash\{\bar{\boldsymbol{\varphi}}_0 \mid \boldsymbol{\varphi}_0 = (0,0,1,0) \succeq \bar{\boldsymbol{\varphi}}_0\}$. We calculate the intersection of the values of the vector $\boldsymbol{k}$, i.e.,

$$\begin{aligned}\mathbb{S}_\cap &= \mathbb{S}_0 \\ =&\{(0,0,0,1),(0,0,1,1),(0,1,0,0),(0,1,0,1),\\ &(0,1,1,0),(0,1,1,1),(1,0,0,0),(1,0,0,1),\\ &(1,0,1,0),(1,0,1,1),(1,1,0,0),(1,1,0,1),\\ &(1,1,1,0),(1,1,1,1)\}\end{aligned}$$

Thus, $(0,0,1,0) \to (0,0,0,1)$ is an valid division trail for $\mathbb{L}$ when the input vector $\boldsymbol{k} \in \mathbb{S}_\cap$.

3. When the vector $\boldsymbol{u} = (0,0,1,1)$, we calculate:

$$\begin{aligned}&\bigoplus_{y\in\mathbb{Y}} \pi_{\boldsymbol{u}=(0,0,1,1)}(\boldsymbol{y}) \\ =&\bigoplus_{y\in\mathbb{Y}} y_2y_3 \\ =&\bigoplus_{x\in\mathbb{X}} (x_0x_1x_2 \oplus x_2 \oplus x_2x_3) \\ =&\bigoplus_{x\in\mathbb{X}} \pi_{(1,1,1,0)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \pi_{(0,0,1,0)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \pi_{(0,0,1,1)}(\boldsymbol{x}) \\ =&(0 \text{ or } \textit{unknown} \text{ Depends on } \boldsymbol{k}) \oplus 1 \oplus (0 \text{ or } \textit{unknown} \text{ Depends on } \boldsymbol{k})\end{aligned}$$

if $\bigoplus_{x\in\mathbb{X}} \pi_{(1,1,1,0)}(\boldsymbol{x}) = 0$, the vector $\boldsymbol{k} \in \mathbb{S}_0 = \mathbb{U}\backslash\{\bar{\boldsymbol{\varphi}}_0 \mid \boldsymbol{\varphi}_0 = (1,1,1,0) \succeq \bar{\boldsymbol{\varphi}}_0\}$. Similarly, we can get $\boldsymbol{k} \in \mathbb{S}_1 = \mathbb{U}\backslash\{\bar{\boldsymbol{\varphi}}_1 \mid \boldsymbol{\varphi}_1 = (0,0,1,0) \succeq \bar{\boldsymbol{\varphi}}_1\}$ and $\boldsymbol{k} \in \mathbb{S}_2 = \mathbb{U}\backslash\{\bar{\boldsymbol{\varphi}}_2 \mid \boldsymbol{\varphi}_2 = (0,0,1,1) \succeq \bar{\boldsymbol{\varphi}}_2\}$. We calculate the intersection of the values of the vector $\boldsymbol{k}$, i.e.,

$$\begin{aligned}\mathbb{S}_\cap = \mathbb{S}_0 \cap \mathbb{S}_1 \cap \mathbb{S}_2 = \{&(0,1,0,1),(0,1,1,1),(1,0,0,1),\\ &(1,0,1,1),(1,1,0,1),(1,1,1,1)\}\end{aligned}$$

Thus, $(0,0,1,0) \to (0,0,1,1)$ is an valid division trail for $\mathbb{L}$ when the input vector $\boldsymbol{k} \in \mathbb{S}_\cap$.

Table 4: Correspondence between the vector $\boldsymbol{u}$ and $\pi_{\boldsymbol{u}}(\boldsymbol{y})$ for the "S-box"

Vector $\boldsymbol{u}$	$\pi_{\boldsymbol{u}}(\boldsymbol{y})$
$\boldsymbol{u} = [0,0,0,0]$	1
$\boldsymbol{u} = [0,0,0,1]$	$y_3 = x_0x_1 \oplus x_2 \oplus x_3$
$\boldsymbol{u} = [0,0,1,0]$	$y_2 = x_2$
$\boldsymbol{u} = [0,0,1,1]$	$y_2y_3 = x_0x_1x_2 \oplus x_2 \oplus x_2x_3$
$\boldsymbol{u} = [0,1,0,0]$	$y_1 = x_1$
$\boldsymbol{u} = [0,1,0,1]$	$y_1y_3 = x_0x_1 \oplus x_1x_2 \oplus x_1x_3$
$\boldsymbol{u} = [0,1,1,0]$	$y_1y_2 = x_1x_2$
$\boldsymbol{u} = [0,1,1,1]$	$y_1y_2y_3 = x_0x_1x_2 \oplus x_1x_2 \oplus x_1x_2x_3$
$\boldsymbol{u} = [1,0,0,0]$	$y_0 = x_0$
$\boldsymbol{u} = [1,0,0,1]$	$y_0y_3 = x_0x_1 \oplus x_0x_2 \oplus x_0x_3$
$\boldsymbol{u} = [1,0,1,0]$	$y_0y_2 = x_0x_2$
$\boldsymbol{u} = [1,0,1,1]$	$y_0y_2y_3 = x_0x_1x_2 \oplus x_0x_2 \oplus x_0x_2x_3$
$\boldsymbol{u} = [1,1,0,0]$	$y_0y_1 = x_0x_1$
$\boldsymbol{u} = [1,1,0,1]$	$y_0y_1y_3 = x_0x_1 \oplus x_0x_1x_2 \oplus x_0x_1x_3$
$\boldsymbol{u} = [1,1,1,0]$	$y_0y_1y_2 = x_0x_1x_2$
$\boldsymbol{u} = [1,1,1,1]$	$y_0y_1y_2y_3 = x_0x_1x_2x_3$

C Division trail for $\mathbb{L}$ of PRESENT S-box

Table 5 presents the division trails for $\mathbb{L}$ of PRESENT S-box.

D Linear inequalities description BDPT of the Extension-S-box

The following inequalities are the 6 inequalities used to describe the "S-box" whose feasible solutions are exactly the 26 division trails for $\mathbb{K}$ of the "S-box" where $(a_0, a_1, a_2, a_3) \rightarrow (b_0, b_1, b_2, b_3)$ denotes a division trail.

$$\mathcal{O} = \begin{cases} -a_0 - a_2 - a_3 + b_0 + b_2 + b_3 \geq 0 \\ -a_1 - a_2 - a_3 + b_1 + b_2 + b_3 \geq 0 \\ a_2 + a_3 - b_0 - b_2 - b_3 + 1 \geq 0 \\ a_2 + a_3 - b_1 - b_2 - b_3 + 1 \geq 0 \\ a_0 + a_1 + a_2 + a_3 - b_0 - b_1 - b_2 - b_3 \geq 0 \\ a_2 - b_2 \geq 0 \\ a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3 \text{ are binaries} \end{cases} \tag{3}$$

The following inequalities are the 10 inequalities used to describe the "S-box" whose feasible solutions are exactly the 30 division trails for $\mathbb{L}$ of the "S-box"

Table 5: Division trails for $\mathbb{L}$ of PRESENT S-box

Input $\boldsymbol{\ell}$	Output $\mathbb{L}$
$[0,0,0,0]$	$\{[0,0,0,0]\}$
$[0,0,0,1]$	$\{[0,0,0,1],[0,1,0,1],[1,0,0,0],[1,1,0,0]\}$
$[0,0,1,0]$	$\{[0,0,1,0],[0,1,1,0],[1,0,0,0],[1,1,0,0]\}$
$[0,0,1,1]$	$\{[0,0,1,1],[0,1,0,0],[0,1,0,1],[0,1,1,0],$ $[1,0,0,1],[1,0,1,0],[1,0,1,1],[1,1,0,0]\}$
$[0,1,0,0]$	$\{[0,0,0,1],[0,1,0,0],[1,0,0,1],[1,1,0,0]\}$
$[0,1,0,1]$	$\{[0,1,0,1],[1,0,0,1],[1,1,0,0]\}$
$[0,1,1,0]$	$\{[0,0,0,1],[0,1,1,0],[1,0,0,0],$ $[1,0,0,1],[1,0,1,0],[1,1,0,0]\}$
$[0,1,1,1]$	$\{[0,0,1,0],[0,0,1,1],[0,1,1,0],[1,0,0,0],$ $[1,0,0,1],[1,0,1,1],[1,1,0,1]\}$
$[1,0,0,0]$	$\{[0,0,0,1],[0,0,1,0],[0,0,1,1],$ $[0,1,0,0],[1,0,0,0],[1,1,0,0]\}$
$[1,0,0,1]$	$\{[0,0,1,1],[0,1,0,0],[0,1,0,1],$ $[0,1,1,0],[1,0,1,0],[1,1,1,0]\}$
$[1,0,1,0]$	$\{[0,0,1,0],[0,1,0,0],[0,1,0,1],$ $[0,1,1,1],[1,0,0,1][1,0,1,0],,$ $[1,0,1,1],[1,1,0,1],[1,1,1,0]\}$
$[1,0,1,1]$	$\{[0,0,1,0],[0,0,1,1],[0,1,0,0],$ $[0,1,1,0],[0,1,1,1],[1,0,0,0],$ $[1,0,1,0],[1,1,0,0],[1,1,0,1]\}$
$[1,1,0,0]$	$\{[0,0,1,0],[0,0,1,1],[1,0,0,1],[1,1,0,0]\}$
$[1,1,0,1]$	$\{[0,0,1,0],[0,1,0,0],[0,1,1,1],[1,0,0,0],$ $[1,0,0,1],[1,0,1,0],[1,1,1,0]\}$
$[1,1,1,0]$	$\{[0,1,0,1],[0,1,1,1],[1,0,1,1],[1,1,0,1],[1,1,1,0]\}$
$[1,1,1,1]$	$\{[1,1,1,1]\}$

where $(a_0,a_1,a_2,a_3)\rightarrow(b_0,b_1,b_2,b_3)$ denotes a division trail.

$$\mathcal{O}=\begin{cases}-a_1-a_2-a_3+b_1+b_2+b_3\geq 0\\ a_0-b_0\geq 0\\ a_1-b_1\geq 0\\ -a_0-a_2-a_3+b_0+b_2+b_3\geq 0\\ a_2-b_2\geq 0\\ a_0-a_1+b_1\geq 0\\ a_0+a_2+a_3-b_3\geq 0\\ -a_0+a_1+b_0\geq 0\\ a_1+a_2+a_3-b_3\geq 0\\ a_3-b_0-b_1-b_2-b_3+3\geq 0\\ a_0,a_1,a_2,a_3,b_0,b_1,b_2,b_3 \text{ are binaries}\end{cases} \tag{4}$$

E Proof of Propositon 1

According to Definition 4, we have the following iteration expression

$$\begin{aligned}\mathbb{K}_i &= f_e(\mathbb{K}_{i-1}) \cup f_k(\mathbb{L}_i) = f_e(\mathbb{K}_{i-1}) \cup f_k \circ f_e(\mathbb{L}_{i-1})\\ \mathbb{L}_i &= f_e(\mathbb{L}_{i-1})\end{aligned}$$

Thus,

$$\begin{aligned}\mathbb{K}_r &= f_e(\mathbb{K}_{r-1}) \cup f_k(\mathbb{L}_r)\\ &= f_e\left(f_e(\mathbb{K}_{r-2}) \cup f_k(\mathbb{L}_{r-1})\right) \cup f_k(\mathbb{L}_r)\\ &= f_e \circ f_e(\mathbb{K}_{r-2}) \cup f_e \circ f_k(\mathbb{L}_{r-1}) \cup f_k(\mathbb{L}_r)\\ &\ \ \vdots\\ &= \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{k}) \cup \underbrace{f_e \circ \cdots \circ f_e}_{r-1} \circ f_k \circ f_e(\boldsymbol{\ell}) \cup \cdots \cup f_k \circ \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{\ell})\\ \mathbb{L}_r &= \underbrace{f_e \circ \cdots \circ f_e}_{r}(\boldsymbol{\ell})\end{aligned}$$

F Integral Distinguishers listed in Table 1

For SIMON and SIMECK family block ciphers, all the integral distinguishers can be extended one more round by the technique in [22]. Moreover, since there is no whitening key at the beginning, we can trivially extend the integral distinguisher of GIFT-64 by one round [1].

F.1 SIMON32's 14-round Distinguisher

$$\begin{aligned}\text{Input:}&(\texttt{caaaaaaaaaaaaaaa,\ aaaaaaaaaaaaaaaa})\\ \text{Output:}&(\texttt{????????????????,\ ?0??????0??????0})\end{aligned}$$

F.2 SIMON48's 15-round Distinguisher

$$\begin{aligned}\text{Input:}&\ (\texttt{caaaaaaaaaaaaaaaaaaaaaaa,\ aaaaaaaaaaaaaaaaaaaaaaaa})\\ \text{Output:}&\ (\texttt{????????????????????????,\ 000000000000000000000000})\end{aligned}$$

F.3 SIMON64's 17-round Distinguisher

$$\begin{aligned}\text{Input:}&\ (\begin{matrix}\texttt{caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,}\\ \texttt{aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa}\end{matrix})\\ \text{Output:}&\ (\begin{matrix}\texttt{????????????????????????????????,}\\ \texttt{000000000000000?00???00000000000}\end{matrix})\end{aligned}$$

F.4 SIMON96's 21-round Distinguisher

Input: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output: (??,
0?0????0?????????????????????????????????0????0?)

F.5 SIMON128's 25-round Distinguisher

Input: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output: (????????????????????????????????
????????????????????????????????,
0?0?????????????????????????????
??????????????????????????????0?)

F.6 SIMECK32's 14-round Distinguisher

Input:(caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa)
Output:(????????????????, 00???00???00???0)

F.7 SIMECK48's 17-round Distinguisher

Input: (caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa)
Output: (????????????????????????, 0???00?????????????00???)

F.8 SIMECK64's 20-round Distinguisher

Input: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output: (????????????????????????????????,
00???0?????????????????????0???0)

F.9 PRESENT's 9-round Distinguisher

Input: (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac)
Output: (???0???0???00000???0???0???00000,
???0???0???00000???0???0???00000)

F.10 RECTANGLE's 10-round Distinguisher

$$\begin{pmatrix} \texttt{aaaaaaaaaaaaaaac} \\ \texttt{aaaaaaaaaaaaaaaa} \\ \texttt{aaaaaaaaaaaaaaaa} \\ \texttt{aaaaaaaaaaaaaaaa} \end{pmatrix} \rightarrow \begin{pmatrix} \texttt{?0??00000???00?0} \\ \texttt{????????????????} \\ \texttt{????????????????} \\ \texttt{????????????????} \end{pmatrix}$$

F.11 GIFT-64's 10-round Distinguisher

Input: (`aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,`
`aaaaaaaaaaaaaaaaaaaaaaaaaaaaacaa`)
Output: (`???b???b???b???b???b???b???b???b,`
`???b???b???b???b???b???b???b???b`)

F.12 Integral Distinguishers of SIMON(102)

In [10], another variant of SIMON family named SIMON(102) is proposed with rotation constants (1,0,2). Hu *et al.* [8] proposed a variant BDPT and applied it to improve the integral distinguishers of SIMON(102). The results are shown in F.13–F.15, where '*' represents that the output bit is '0' or '1'.

F.13 SIMON(102) 32's 14-round Distinguisher in [8]

Input:(`caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa`)
Output:(`????????????????, 0*?????????????*`)

F.14 SIMON(102) 48's 15-round Distinguisher in [8]

Input: (`caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa`)
Output: (`????????????????????????, 0*?????????????????????*`)

F.15 SIMON(102) 64's 17-round Distinguisher in [8]

Input: (`caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,`
`aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa`)
Output: (`????????????????????????????????,`
`0*?????????????????????????????*`)

Determining '*' is '0' or '1' can be helpful to integral attacks on SIMON(102). Therefore, we apply Algorithm 3 and 4 to search integral distinguishers of SIMON(102) based on BDPT, and obtain more accurate integral distinguishers compared with [8]. The results are shown in F.16–F.18. Besides, these integral distinguishers can be obtained by the method of exploring secret keys in [25]. Note that our automatic search model supposes that all secret keys are chosen randomly. If consider the secret keys, we may obtain better integral distinguishers.

F.16 SIMON(102) 32's 14-round Distinguisher

Input:(`caaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaa`)
Output:(`????????????????, 01?????????????1`)

F.17 SIMON(102) 48's 15-round Distinguisher

```
 Input: (caaaaaaaaaaaaaaaaaaaaaaa, aaaaaaaaaaaaaaaaaaaaaaaa)
Output: (????????????????????????, ????????????????????????)
```

F.18 SIMON(102) 64's 17-round Distinguisher

```
 Input: (caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
         aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output: (????????????????????????????????,
         01?????????????????????????????1)
```