Integral Attacks on Some Lightweight Block Ciphers

Shiqiang Zhu[1], Gaoli Wang[1,2*], Yu He[1] and Haifeng Qian[1]
[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[e-mail: glwang@sei.ecnu.edu.cn]
*Corresponding author: Gaoli Wang

Received April 13, 2020; revised October 8, 2020; accepted November 5, 2020; published November 30, 2020

Abstract

At EUROCRYPT 2015, Todo proposed a new technique named division property, and it is a powerful technique to find integral distinguishers. The original division property is also named word-based division property. Later, Todo and Morii once again proposed a new technique named the bit-based division property at FSE 2016 and find more rounds integral distinguisher for SIMON-32. There are two basic approaches currently being adopted in researches under the bit-based division property. One is conventional bit-based division property (CBDP), the other is bit-based division property using three-subset (BDPT). Particularly, BDPT is more powerful than CBDP. In this paper, we use Boolean Satisfiability Problem (SAT)-aided cryptanalysis to search integral distinguishers. We conduct experiments on SIMON-32/-48/-64/-96, SIMON (102)-32/-48/-64, SIMECK-32/-48/-64, LBlock, GIFT and Khudra to prove the efficiency of our method. For SIMON (102)-32/-48/-64, we can determine some bits are odd, while these bits can only be determined as constant in the previous result. For GIFT, more balanced (zero-sum) bits can be found. For LBlock, we can find some other new integral distinguishers. For Khudra, we obtain two 9-round integral distinguishers. For other ciphers, we can find the same integral distinguishers as before.

Keywords: Division property, three-subset, integral distinguisher, SAT, block cipher

http://doi.org/10.3837/tiis.2020.11.014 **ISSN : 1976-7277**

1. Introduction

Division property is a novel technique proposed by Todo [1] at EUROCRYPT 2015, which is a powerful technique to find integral distinguishers and has been applied to many ciphers. Division property could precisely depict the implicit features between traditional ALL and BALANCED properties [2]. With the division property, Todo achieved the first theoretical key recovery attack on full MISTY1 [3]. Furthermore, the integral distinguisher of generalized Feistel structures can be also found by division property and got more rounds integral distinguishers against LBlock and TWINE in [4].

Although the division property is more powerful to find intgral distinguishers than other methods, it search integral distinguishers at the word level. At FSE 2016, Todo and Morii [5] proposed a new technique that decomposes word-based division property, and they call that technique as a bit-based division property. The bit-based division property includes bit-based division property (CBDP) and bit-based division property using three-subset (BDPT). Specifically, the parity of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is 0 or unknown based on the CBDP, and the parity of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is 0, 1 and unknown based on the BDPT. In BDPT, the unknown set in CBDP is divided into the odd-parity set and the unknown set that means more information can be traced. Know then, BDPT is more powerful to find integral distinguisher than two-subset. For SIMON-32 [6], the 13-round integral distinguisher is found by CBDP while one more round can be found by BDPT than CBDP.

According to the result of Todo *et al.*, the bit-based division property is quite effective to find integral distinguishers. Unfortunately, as pointed out in [5], the bit-based division property is upper bounded by 2^n for an *n*-bit primitives, so it can not apply bit-based division property to some ciphers with large block sizes. At ASIACRYPT 2016, Xiang *et al*. [7] has solved that problem by utilizing MILP method. With this method, the primitives whose block sizes are larger than 32 can be analyzed. Later on, Sun *et al*. [8] convert the searching distinguisher problem into SAT/SMT problem [9]and use the automatic tool to solve it for ARX-based block ciphers.

It is noticed that there is no method to convert the BDPT into the automatic search model. Additionally, no research has been found that surveyed the method to add some new vectors into $\mathbb{K}$* and remove some redundant vectors in $\mathbb{L}$ based the automatic search tool when the cipher contains the Key-XOR operation. To overcome this problem, many further pieces of research have occurred. Hu *et al*. [10] propose a new method named the variant three-subset division property (VTDP) and they do not remove the vectors that appear even number of times. Thus, the VTDP is weaker than BDPT. At ASIACRYPT 2019, Wang *et al*.[11] introduced a new method named pruning techniques which can remove vectors that appear even number of times, and they modelled the BDPT based on MILP.

Our contributions. In this paper, we search the integral distinguisher of SIMON-32/-48/-64/-96, SIMON (102)-32/-48/-64, SIMECK-32/-48/-64, LBlock, GIFT and Khudra and get some more accurate results than before. We transfer the problem of security analysis to mathematical problem and use the automatic tool to work out mathematical problem. The contributions are listed as follows:

* The definition of $\mathbb{K}$ and $\mathbb{L}$ is introduced in Section 2.

Model for $\mathbb{L}$ and S-box. To search integral distinguishers by automatic tool, we should convert the propagation of BDPT to mathematical models. For $\mathbb{K}$, the models are the same as those in [8]. For $\mathbb{L}$, we build the models for COPY, XOR, AND respectively. We also use the same method as [7,11] to get all division trials of S-box. After getting all division trails of S-box, we use the Conjunctive Normal Form (CNF) that are the input of Cryptominisat solver to model the division trails.

We find the following new results.

1. Let SIMON-2*n*/SIMECK-2*n* be the SIMON/SIMECK block ciphers where 2*n* is block sizes, and *n* takes vaules form 16, 24, 32 and 48/16, 24 and 32. SIMON(102)-2*n* [12] is a variant of SIMON-2*n* family. SIMON(102)-2*n* alters the rotation from (1, 8, 2) to (1, 0, 2). For SIMON(102)-32/-48/-64, [10] points out that some bits are constant. In this paper, we can know that these bits are odd or balanced.
2. For GIFT, when the data complexity of the input is 2^{61} chosen plaintexts, we can find 4 more balanced bits than befoe in [13]. And when the data complexity of the input is 2^{63} chosen plaintexts, we can find 2 more balanced bits than befoe.
3. For LBlock, under the same number of rounds as before, we can obtain more integral distinguishers on 17-round.
4. For Khudra, we get the first two 9-round integral distinguishers.
5. For SIMON-2*n* and SIMECK-2*n*, the distinguishers that we find are the same as before.

Our results and comparisons are shown in **Table 1**.

Table 1. Summarization of integral distinguishers for some block ciphers

Cipher	Block Size (bit)	Data	Rounds	Time	Number of odd/balanced bits	Ref
SIMON-32	32	2^{31}	15	27s	3	[10]
			15	2m	3	[11]
			15	9m	3	Sec4.1
SIMON (102)-32[1]	32	2^{31}	20	25s	3	[10]
			20	5m	3	Sec4.1
SIMON (102)-48[1]	48	2^{47}	28	9.3s	3	[10]
			28	1h	3	Sec4.1
SIMON (102)-64[1]	64	2^{63}	36	1.1h	3	[10]
			36	11h	3	Sec4.1
SIMECK-32	32	2^{31}	15	51m	7	[11]
			15	11m	7	Sec4.1
GIFT-64	64	2^{63}	9		30	[13]
	64		9	2h6m	32	Sec4.2
	64	2^{61}	9		5	[13]
	64		9	2h13m	9	Sec4.2
LBlock-64[2]	64	2^{63}	17		4	[13]
	64		17	10h25	4	[11]
	64		17	1h21m	4	Sec4.3
Khudra-64	64	2^{63}	9	14h34	16	Sec4.4

[1] For SIMON(102), [10] points out that some bits are constant. In this paper,we can determine that these bits are odd or balanced.

[2] For LBlock, we can find some other integral distinguishers that are not the same as before.

[3] The platforms that experiments are implemented are listed as follows.

[10]:Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz and 96 GB memory.

[11]: Intel Celeron CPU 1007U I5 4590 @1.5GHz, 6.00 RAM, 64-bit Windows system.

This paper: Intel(R) Core(TM) i7-8700 CPU @3.19 GHz, 64-bit Windows system.

Organization of the paper. This overall structure is as follows: We introduce some notations and division property in Section 2. Section 3 introduces the models of some basic primitives based on CNF and gives a search algorithm. Section 4 shows some applications of our models. Section 5 Summarizes the results of the study.

2. PRELIMINARIES

2.1 Notation

All notations used in this paper will be introduced in this subsection. $\mathbb{F}_2$ denotes the smallest finite filed, and the two elements are zero and one. $\mathbb{F}_2^n$ denotes the n-bit string that only contains zero or one, and $a[i]$ denotes the i-th bit of a, where a is an n-bit vector in $\mathbb{F}_2^n$. The Hamming weight $w(a)$ of a is calculated as $\sum_0^{n-1} a[i]$. For any $\boldsymbol{a} = (a_0, a_1, \cdots, a_{m-1}) \in \mathbb{F}_2^{n_0} \times \cdots \times \mathbb{F}_2^{n_{m-1}}$, $\boldsymbol{W(a)} = (w(a_0), \cdots, w(a_{m-1}))$ denotes the vectorial Hamming weight of $\boldsymbol{a}$. Let $\boldsymbol{k} = (k_0, k_1, \cdots, k_{m-1})$ and $\boldsymbol{k^*} = (k_0^*, k_1^*, \cdots, k_{m-1}^*)$. If $k_i \geq k_i^*$ holds for all $i \in \{0, 1, \cdots, m-1\}$, we have $\boldsymbol{k} \succcurlyeq \boldsymbol{k^*}$; otherwise we have $\boldsymbol{k} \not\succcurlyeq \boldsymbol{k^*}$. $\mathbb{K}$ and $\mathbb{L}$ denote the set of $\boldsymbol{k}$ and $\boldsymbol{l}$, respectively, and $\boldsymbol{l} = (l_0, l_1, \cdots, l_{m-1})$.

Bit product function.[1] $\pi_u(x)$ denotes bit product function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. For any $u \in \mathbb{F}_2^n$ and $x \in \mathbb{F}_2^n$, $\pi_u(x)$ is defined as:

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

When the value of $u[i]$ equals zero, $x[i]^{u[i]}$ equals one. Otherwise the result of $x[i]^{u[i]}$ equals $x[i]$.

For any $\boldsymbol{u} = (u_0, \cdots, u_{m-1}) \in (\mathbb{F}_2^{n_0} \times \cdots \times \mathbb{F}_2^{n_{m-1}})$ and $\boldsymbol{x} = (x_0, \cdots, x_{m-1}) \in (\mathbb{F}_2^{n_0} \times \cdots \times \mathbb{F}_2^{n_{m-1}})$, $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is a function mapping from $\mathbb{F}_2^{n_0} \times \cdots \times \mathbb{F}_2^{n_{m-1}}$ to $\mathbb{F}_2$, and $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is defined as:

$$\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{m-1} \pi_{u_i}(x_i) \tag{2}$$

Subset $\mathbb{S}_k^n$. The $\mathbb{S}_k^n$ is a subset of $\mathbb{F}_2^n$ for any integer $k \in \{0, 1, 2, \cdots, n\}$. The subset $\mathbb{S}_k^n$ is a set of all $u \in \mathbb{F}_2^n$ satisfying $w(u) \geq k$, and it is defined as

$$\mathbb{S}_k^n = \{u \in \mathbb{F}_2^n | w(u) \geq k\}. \tag{3}$$

2.2 Division property

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

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \begin{cases} unknown, \text{if there exists } \boldsymbol{k}\in\mathbb{K}\ s.t.\ \boldsymbol{u}\succcurlyeq\boldsymbol{k}, \\ 0, \qquad\quad \text{otherwise.} \end{cases} \tag{4}$$

If there are $\boldsymbol{k}$ and $\boldsymbol{k}'$ belonging to $\mathbb{K}$ and satisfying $\boldsymbol{k}\succcurlyeq\boldsymbol{k}'$, the $\boldsymbol{k}$ will be considered as a redundancy, and it should be removed from $\mathbb{K}$.

It notes that **division property** is regarded as **bit-based division property** when *n* is restricted to 1.

Propagation rules

Rule 1 (COPY in CBDP). The COPY function is defined as: $(y_1, y_2) = (x, x)$, where x takes value from $\mathbb{F}_2^1$. $\mathbb{X}$ denotes input multiset and $\mathbb{Y}$ denotes output multiset. Assuming that the CBDP of the input multiset $\mathbb{X}$ is $D^1_{\{k\}}$, then the output multiset $\mathbb{Y}$ has CBDP $\mathcal{D}^{1,1}_{\mathbb{K}'}$ as follows:

$$\mathbb{K}' = \begin{cases} \{0,0\} & \text{if } k=0, \\ \{(1,0),(0,1)\} & \text{if } k=1. \end{cases} \tag{5}$$

Rule 2 (XOR in CBDP). The XOR function is defined as: $y = x_1 \oplus x_2$, where (x_1, x_2) takes value from $(\mathbb{F}_2^1 \times \mathbb{F}_2^1)$. $\mathbb{X}$ denotes input multiset and $\mathbb{Y}$ denotes output multiset. Assuming that the CBDP of the input multiset $\mathbb{X}$ is $\mathcal{D}^{1,1}_{\{\boldsymbol{k}\}}$, then the output multiset $\mathbb{Y}$ has CBDP $\mathcal{D}^1_{\mathbb{K}'}$ that is calculated as follows:

$$\mathbb{K}' = \begin{cases} \{(0)\} & \text{if } \boldsymbol{k}=(0,0), \\ \{(1)\} & \text{if } \boldsymbol{k}=(0,1) \text{ or} (1,0), \\ \emptyset & \text{if } \boldsymbol{k}=(1,1). \end{cases} \tag{6}$$

Rule 3 (AND in CBDP). The AND function is defined as: $y = x_1 \wedge x_2$, where (x_1, x_2) takes value from $(\mathbb{F}_2 \times \mathbb{F}_2)$. $\mathbb{X}$ denotes input multiset and $\mathbb{Y}$ denotes output multiset. Assuming that the CBDP of the input multiset $\mathbb{X}$ is $\mathcal{D}^{1,1}_{\{\boldsymbol{k}\}}$, then the output multiset $\mathbb{Y}$ has CBDP $\mathcal{D}^1_{\mathbb{K}'}$ that is calculated as follows:

$$\mathbb{K}' = \begin{cases} \{(0)\} & \text{if } \boldsymbol{k}=(0,0), \\ \{(1)\} & \text{otherwise.} \end{cases} \tag{7}$$

For more details of those rules, please refer to [1,5].

Definition 2. (Division trail [7]). *Let f denote the round function of block cipher. Assuming that input multiset $\mathbb{X}$ has CBDP $\mathcal{D}^{1,n}_{\{\boldsymbol{k}\}}$, and after i-round propagations, $\mathcal{D}^{1,n}_{\mathbb{K}'_i}$ denotes the CBDP of output multiset, thus, we have the trail of division property propagations: $\{\boldsymbol{k}\} \overset{def}{=} \mathbb{K}_0 \xrightarrow{f} \mathbb{K}_1 \xrightarrow{f} \mathbb{K}_2 \xrightarrow{f} \cdots \xrightarrow{f} \mathbb{K}_i \cdots \xrightarrow{f} \mathbb{K}_{r-1}$. Moreover, for any vector $\boldsymbol{k}_i \in \mathbb{K}_i$, there must exist a vector $\boldsymbol{k}_{i-1} \in \mathbb{K}_{i-1}$ which can propagate to $\boldsymbol{k}_i$ for any $i \in \{0,1,2,3,\cdots,r-1\}$, then we call $\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \cdots \to \boldsymbol{k}_{r-1}$ is an r- round division trail.*

2.3 SAT-aided CBDP

Division property needs to find all division trails, which leads to high computational and memory complexities. One of the most effective methods that solve above problem is the automatic search technique. Sun *et al* [8] transfer the problem of security analysis to SAT/SMT problem and use the automatic tool to work out that problem based on CBDP. In this subsection, we will review how to model some primitives, such as COPY, XOR, AND.

SAT Mode 1 (COPY). The COPY function copies a bit x to y_1 and y_2. According to the Rule 1, we can get all valid transitions (0,0,0), (1,0,1) and (1,1,0). The following CNFs depict the division trail of COPY:

$$\begin{cases} \neg y_1 \vee \neg y_2 = 1 \\ x \vee y_1 \vee \neg y_2 = 1 \\ x \vee \neg y_1 \vee y_2 = 1 \\ \neg x \vee y_1 \vee y_2 = 1 \end{cases} \tag{8}$$

SAT Mode 2 (XOR). The XOR function creates y=$x_1 \oplus x_2$. According to the Rule 2,we can get all valid transitions (0,0,0), (0,1,1), (1,0,1). The following CNFs depict the division trail of XOR:

$$\begin{cases} \neg x_1 \vee \neg x_2 = 1 \\ x_1 \vee x_2 \vee \neg y = 1 \\ x_1 \vee \neg x_2 \vee y = 1 \\ \neg x_1 \vee x_2 \vee y = 1 \end{cases} \tag{9}$$

SAT Mode 3 (AND). The AND function creates y=$x_1 \wedge x_2$. According to the Rule 3, we can get all valid transitions (0,0,0), (0,1,1), (1,0,1), (1,1,1). The following CNFs depict the division trail of AND:

$$\begin{cases} \neg x_2 \vee y = 1 \\ x_1 \vee x_2 \vee y = 1 \\ \neg x_1 \vee y = 1 \end{cases} \tag{10}$$

Initial division property and Stopping rules of CBDP. Using automatic tool to search integral distinguisher needs to set initial division property and proper stopping rules.

Initial division property. Assuming that $(a_0^0, a_1^0, a_2^0, \cdots, a_{n-1}^0) \rightarrow \cdots \rightarrow (a_0^r, a_1^r, a_2^r, \cdots, a_{n-1}^r)$ is an r-round division trail, where n is the length of cipher. Let the initial division property be denoted as $\mathcal{D}_{\boldsymbol{k}}^{1,n}$ and $\boldsymbol{k} = (k_0, k_1, \cdots, k_{n-1})$. Then we can set

$$a_i^0 = k_i, i = 0, 1, 2, 3, \cdots, n-1 \tag{11}$$

Stopping rules. To check the division property of the m-th $(0 \leq m \leq n-1)$ output bit, we need to add proper constraints on a_i^r $(i = 0, 1, 2, \cdots, n-1)$ that

$$a_i^r = \begin{cases} 1 & \text{if } i = m, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

After setting initial division property and stop rules, if automatic tool has a solution, the division property of the m-th output bit is unknown; otherwise, we regard that the division property of the m-th output bit is balanced.

2.4 SAT-aided method of CBDP

For improving the efficiency of searching distinguisher, we should convert the searching distinguisher problem into mathematical models. Fisrt, we need to get a CNF set $\mathcal{C}$ which describes the *r*-round division trails. Then, we set the initial division property and stop division property. The stop division property is $\mathcal{D}_{\boldsymbol{k}_r}^{1,n}$, where $\boldsymbol{k}_r = (a_0^r = 0, a_1^r = 0, \cdots, a_i^r = 1, \cdots, a_{n-1}^r = 0)$. If $\mathcal{C}$ has a solution, it indicates that the integral property of the *i*-th bit is unknown for the output of *r*-round cipher. Otherwise, the integral property of the *i*-th bit is balanced. The following Algorithm shows the detailed process.

Algorithm STwoDP$(E, \boldsymbol{k}, i)$

Input: The cipher E with n-bit,the initial division property $\mathcal{D}_{\boldsymbol{k}}^{1,n}$, and the number i.
Output: Whether the i-th bit of the output is balanced or not based on two-subset division property.
begin
    $\mathcal{C}$ is a CNF set which describe the two-subset division property division trails of E with given initial division property $\mathcal{D}_{\boldsymbol{k}}^{1,n}$.
    Let stop division property $\boldsymbol{k}_r = (a_0^r = 0, a_1^r = 0, \cdots, a_i^r = 1, \cdots, a_{n-1}^r = 0)$ and assign values to the $\mathcal{C}$.
    if $\mathcal{C}$ has feasible solution
        return unknown
    else
        return balanced
end

2.5 BDPT

Definition 3. (BDPT [5]). *Let $\mathbb{X}$ be a multiset with values in $(\mathbb{F}_2)^m$, $\boldsymbol{k}$ and $\boldsymbol{l}$ denote m-dimensional vectors whose i-th element take 0 or 1. When the division property of multiset $\mathbb{X}$ is $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1,m}$, it fulfils the following conditions:*

$$\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \begin{cases} unknown, & \text{if there exists } \boldsymbol{k}\in\mathbb{K}\ s.t.\ \boldsymbol{u}\succcurlyeq\boldsymbol{k}, \\ 1, & \text{else if there is } \boldsymbol{l}\in\mathbb{L}\ s.t.\ \boldsymbol{u}=\boldsymbol{l}, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

It notes that if there is a $\boldsymbol{k}\in\mathbb{K}$ satisfying $\boldsymbol{u}\succcurlyeq\boldsymbol{k}$, $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is unknown even if there is a $\boldsymbol{l}\in\mathbb{L}$ satisfying $\boldsymbol{l}=\boldsymbol{u}$.

Propagation rules

Rule 4 (COPY in BDPT [5]). The COPY function F is defined as: $\boldsymbol{y}=F(\boldsymbol{x})$, where $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ *and* $\boldsymbol{y}=(x_1,x_1,x_2,x_3,\cdots,x_n)$. Assuming that input multiset $\mathbb{X}$ has BDPT $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1,n}$, and output multiset $\mathbb{Y}$ has BDPT $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1,n+1}$, then $\mathbb{K}'$ and $\mathbb{L}'$ are computed as:

$$\begin{aligned} \mathbb{K}' &\leftarrow \begin{cases} (0,0,k_2,\cdots,k_n), & if\ k_1=0 \\ (1,0,k_2,\cdots,k_n),(0,1,k_2,\cdots,k_n), & if\ k_1=1 \end{cases} \\ \mathbb{L}' &\leftarrow \begin{cases} (0,0,l_2,\cdots,l_n), & \\ (1,0,l_2,\cdots,l_n),(0,1,l_2,\cdots,l_n), & \\ (1,1,l_2,\cdots,l_n). & \end{cases} if\ l_1=1 \end{aligned} \tag{14}$$

from all $k\in\mathbb{K}$ and all $l\in\mathbb{L}$, respectively.

Rule 5 (AND in BDPT [5]). The AND function F is defined as: $\boldsymbol{y}=F(\boldsymbol{x})$, where $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ and $\boldsymbol{y}=(x_1\wedge x_2,x_3,\cdots,x_n)$. Assuming that input multiset $\mathbb{X}$ has BDPT $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1,n}$, and output multiset $\mathbb{Y}$ has BDPT $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1,n-1}$, then $\mathbb{K}'$ is computed from all $k\in\mathbb{K}$ as:

$$\mathbb{K}' \leftarrow (\lceil\frac{k_1+k_2}{2}\rceil, k_3, k_4, \cdots, k_n) \tag{15}$$

Moreover, $\mathbb{L}'$ is computed from all $l\in\mathbb{L}$ satisfying $(l_1,l_2)=(0,0)$ or $(1,1)$ as:

$$\mathbb{L}' \leftarrow (\lceil\frac{l_1+l_2}{2}\rceil, l_3, l_4, \cdots, l_n) \tag{16}$$

Rule 6 (XOR in BDPT [5]). The XOR function F is defined as: $\boldsymbol{y}=F(\boldsymbol{x})$, where $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ and $\boldsymbol{y}=(x_1\oplus x_2,x_3,\cdots,x_n)$. Assuming that input multiset $\mathbb{X}$ has BDPT$\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1,n}$, and output multiset $\mathbb{Y}$ has BDPT $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1,n-1}$, then $\mathbb{K}'$ is computed from all $k\in\mathbb{K}$ which satisfies

$(k_1, k_2) = (0,0),\ (1,0)$ or $(0,1)$ as:

$$\mathbb{K}^{'} \leftarrow (k_1 + k_2, k_3, k_4, \cdots, k_n) \tag{17}$$

Moreover, $\mathbb{L}^{'}$ is computed from all $\boldsymbol{l} \in \mathbb{L}$ satisfying $(l_1, l_2) = (0,0),\ (1,0)$ or $(0,1)$ as:

$$\mathbb{L}^{'} \xleftarrow{x} (l_1 + l_2, l_3, l_4, \cdots, l_n) \tag{18}$$

where $\mathbb{L} \xleftarrow{x} \boldsymbol{l}$ means

$$\mathbb{L} \leftarrow \begin{cases} \mathbb{L} \cup \boldsymbol{l} & if\ the\ original\ \mathbb{L}\ does\ not\ include\ \boldsymbol{l}, \\ \mathbb{L} \setminus \boldsymbol{l} & if\ the\ original\ \mathbb{L}\ includes\ \boldsymbol{l}. \end{cases}$$

Rule 7 (XOR with secret round key [5]). Let the input and output multiset be $\mathbb{X}$ and $\mathbb{Y}$ satisfying $\mathcal{D}^{1^n}_{\mathbb{K},\mathbb{L}}$ and $\mathcal{D}^{1^n}_{\mathbb{K}^{'},\mathbb{L}^{'}}$ respectively. Assuming that the round key is XORed with the i-th bit $(0 \le i \le n)$, then $\mathbb{K}^{'}$ and $\mathbb{L}^{'}$ are calculated as follows:

$$\mathbb{K}^{'} \leftarrow (l_0, l_1, \cdots, l_i \vee 1, \cdots, l_{n-1}), \qquad \mathbb{L}^{'} = \mathbb{L}, \tag{19}$$

for all $\boldsymbol{l} \in \mathbb{L}$ satisfying $l_i = 0$.

Rule 8 (The S-box for $\mathbb{K}$ and $\mathbb{L}$). Let $F_2^m \rightarrow F_2^n$ be a substitution function, where the input $\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_m)$ and the output $\boldsymbol{y} = (y_1, y_2, y_3, \cdots, y_n)$. Then $\boldsymbol{y}$ is calculated as:

$$\begin{aligned} y_1 &= f_1(x_1, x_2, x_3, \cdots, x_m), \\ &\vdots \\ y_n &= f_n(x_1, x_2, x_3, \cdots, x_m). \end{aligned}$$

Assuming that the input multiset $\mathbb{X}$ has the BDPT $\mathcal{D}^{1,n}_{\mathbb{K},\mathbb{L}}$, the output multiset $\mathbb{Y}$ has the BDPT $\mathcal{D}^{1,n}_{\mathbb{K}^{'},\mathbb{L}^{'}}$. For each vector $\boldsymbol{k} \in \mathbb{K}$, decide for each vector $\boldsymbol{v} \in \mathbb{F}_2^n$ whether the polynomial $\pi_{\boldsymbol{v}}(\boldsymbol{y})$ contains any monomial $\pi_{\boldsymbol{k}^{'}}(\boldsymbol{x})$ where $\boldsymbol{k}^{'} \succcurlyeq \boldsymbol{k}$. If so, $(\boldsymbol{k}^{'}, \boldsymbol{v})$ is regard as a division trail of the S-box and $\boldsymbol{v}$ will be appended to $\mathbb{K}^{'}$. For each vector $\boldsymbol{l} \in \mathbb{L}$, decide for each vector $\boldsymbol{v} \in \mathbb{F}_2^n$ whether the polynomial $\pi_{\boldsymbol{v}}(\boldsymbol{y})$ contains monomial $\pi_{\boldsymbol{l}}(\boldsymbol{x})$. Then, $(\boldsymbol{l}, \boldsymbol{v})$ is regard as a division trail of the S-box and $\boldsymbol{v}$ will be appended to $\mathbb{L}^{'}$. For more details of S-box, please refer to [11,7].

2.6 Prunning techniques

The round function with many operations will generate many redundant division properties. With the increase of the round numbers or the size of the block cipher, it is infeasible to get all division trails. To overcome this problem, we introduce the prunning techniques which were proposed by Wang *et al*. [11]

Theorem 1 (Prunning techniques of $\mathbb{K}$ [11]). *For an r-round cipher E, let $\mathcal{D}^{1,n}_{\mathbb{K},\mathbb{L}}$ be the input division property of E. If $\boldsymbol{k} \in \mathbb{K}$ cannot output the unit vector $\boldsymbol{e}$ of E based on CBDP eventually, then $\mathcal{D}^{1,n}_{\mathbb{K}\rightarrow\boldsymbol{k},\mathbb{L}}$ will not generate the vector $\boldsymbol{e} \in \mathbb{K}_{r+1,0}$ based on BDPT, where $\mathbb{K} \rightarrow \boldsymbol{k}$ denotes removing $\boldsymbol{k}$ from $\mathbb{K}$.*

Theorem 2 (Prunning techniques of $\mathbb{L}$ [11]). *For an r-round cipher E, let $\mathcal{D}^{1,n}_{\mathbb{K},\mathbb{L}}$ be the input division property of E. If $\boldsymbol{l} \in \mathbb{L}$ cannot output the unit vector $\boldsymbol{e}$ of E based on CBDP eventually, then $\mathcal{D}^{1,n}_{\mathbb{K},\mathbb{L}\rightarrow\boldsymbol{l}}$ will not generate the vector $\boldsymbol{e} \in \mathbb{K}_{r+1,0}$ based on BDPT, where $\mathbb{L} \rightarrow \boldsymbol{l}$ denotes removing $\boldsymbol{l}$ from $\mathbb{L}$.*

The BDPT is more powerful to find integral property than CBDP. However, it is infeasible to search integral distinguisher of some ciphers with large block sizes. By the "fast propagation", Wang [11] can resolve this problem against BDPT.

Definition 4 (Fast propagation [11]). Let $\mathcal{D}^{1,n}_{\mathbb{K},\mathbb{L}}$ *be input division property of cipher E, and we translate the BDPT into CBDP $\mathcal{D}^{1,n}_{\overline{\mathbb{K}}}$ where $\overline{\mathbb{K}} = \mathbb{K} \cup \mathbb{L}$. The output division property of E is computed from all vectors of $\overline{\mathbb{K}} = \mathbb{K} \cup \mathbb{L}$ based on CBDP.*

3. Modeling for the BDPT

Due to much time complexity, it is infeasible to search integral property of some ciphers that have large block sizes. To overcome the limit of the huge complexity, the researchers begin to transfer the problem of security analysis to mathematical problem and use the automatic tool to work out mathematical problem, such as SAT, MILP, CP *etc*.

3.1 SAT-aided BDPT

In order to use automatic tools to improve efficiency, we should model the primitives of cipher by CNFs.

Assuming that f is the round function of the block cipher, let $\mathcal{D}^{1,n}_{\mathbb{K},\mathbb{L}}$ be the BDPT of input and $\mathcal{D}^{1,n}_{\mathbb{K}',\mathbb{L}'}$ be the BDPT of output multiset. We use CNF to constrain division trails of $\mathbb{K} \xrightarrow{f} \mathbb{K}'$ and $\mathbb{L} \xrightarrow{f} \mathbb{L}'$ based on the propagation rules of BDPT, respectively. Those models for $\mathbb{K}$ are the same as before [8], therefore, we do not introduce how to model for $\mathbb{K}$. We only give models for $\mathbb{L}$ in this subsection.

SAT Model 4 (COPY for $\mathbb{L}$). Let f be a COPY function. $x \xrightarrow{f} (y_1, y_2)$ denotes a COPY operation where $y_1 = x$ and $y_2 = x$. Assuming that input multiset has $\mathcal{D}^1_l$, then the output multiset has $\mathcal{D}^1_{(l,l),(0,l),(1,l-1),\cdots,(l,0)}$ from Rule 4. When $l = 1$, the output multiset has $\mathcal{D}^1_{(0,1),(1,0),(1,1)}$, otherwise, if $l = 0$, the output multiset has $\mathcal{D}^1_{(0,0)}$. Therefore, the following CNFs are depicted for $\mathbb{L}$:

$$\begin{cases} x \vee \neg y_1 = 1 \\ x \vee y_1 \vee \neg y_2 = 1 \\ \neg x \vee y_1 \vee y_2 = 1 \end{cases} \tag{20}$$

Apparently, all solutions of the above CNFs corresponding to (x, y_1, y_2) are $(0,0,0)$, $(1,0,1)$ and $(1,1,0)$, $(1,1,1)$.

SAT Model 5 (AND for $\mathbb{L}$). Let f be an AND function. $(x_1, x_2) \xrightarrow{f} y$ denotes the AND operation where $y = x_1 \& x_2$. Assuming that input multiset has $\mathcal{D}^{1,2}_{\boldsymbol{l}}$ where $\boldsymbol{l} = (l_0, l_1)$, then the output multiset has $\mathcal{D}^1_{\lceil \frac{l_1+l_2}{2} \rceil}$ from Rule 5. When $l_1 = 1$ and $l_2 = 1$, the output multiset has $\mathcal{D}^1_1$, otherwise if $l_1 = 0$ and $l_2 = 0$, the output multiset has $\mathcal{D}^1_0$. As result, the following CNFs are depicted for $\mathbb{L}$:

$$\begin{cases} \neg x_2 \vee y = 1 \\ \neg x_1 \vee x_2 = 1 \\ x_1 \vee \neg y = 1 \end{cases} \tag{21}$$

Apparently, all solutions of the above CNFs corresponding to (x_1, x_2, y) are $(0,0,0)$ and $(1,1,1)$.

SAT Model 6 (XOR for $\mathbb{L}$). Let f be an XOR function. $(x_1, x_2) \xrightarrow{f} y$ denotes the XOR operation where $y = x_1 \wedge x_2$. Assuming that input multiset has $\mathcal{D}^{1,2}_{\boldsymbol{l}}$ where $\boldsymbol{l} = (l_0, l_1)$, then the output multiset has $\mathcal{D}^1_{l_1+l_2}$ from Rule 6. When $l_1 = 0, l_2 = 1$ or $l_1 = 1, l_2 = 0$, the output multiset has $\mathcal{D}^1_1$. Otherwise if $l_1 = 0$ and $l_2 = 0$, the output multiset has $\mathcal{D}^1_0$. As a result, the following CNFs are depicted for $\mathbb{L}$:

$$\begin{cases} \neg x_1 \vee \neg x_2 = 1 \\ x_1 \vee x_2 \vee \neg y = 1 \\ x_1 \vee \neg x_2 \vee y = 1 \\ \neg x_1 \vee x_2 \vee y = 1 \end{cases} \tag{22}$$

Apparently, all solutions of the above CNFs corresponding to (x_1, x_2, y) are $(0,0,0)$, $(0,1,1)$ and $(1,0,1)$. Although Model 6 can get all division trails of XOR, vectors appearing even number of times are not eliminated, so it needs to eliminate these vectors after getting all division trails.

Representing the division trails of S-box as CNFs. According to Rule 8, it's easy to get all division trails through S-box. Assuming that $\boldsymbol{s}$ is a substitution function that consists of an S-box with n -bit input and m -bit output. Let the input and output multiset of $\boldsymbol{s}$ be $\mathbb{X}$ and $\mathbb{Y}$ satisfying $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1,n}$ and $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1,m}$, respectively. We allocate variables to represent the input and output of $\boldsymbol{s}$ at the bit level. Then, we create a truth table for the variables. The truth table has a total of 2^{n+m} elements. If an element of the truth table satisfies one of the division trails $\boldsymbol{k} \longrightarrow \boldsymbol{k}'$ where $\boldsymbol{k} \in \mathbb{K}, \boldsymbol{k}' \in \mathbb{K}'$, then this element of the truth table is set to be true. Finally, it is easy to use CNFs to depict the division trails of $\mathbb{K} \longrightarrow \mathbb{K}'$. We can also use the same method to depict the division trails of $\mathbb{L} \longrightarrow \mathbb{L}'$.

So far, we have modelled the propagation rules of $\mathbb{L}$ and the division trails of S-box. Thus, for some block ciphers based on simple primitives, we can construct CNFs to simulate a round of division property propagation.

3.2 Initial division property

When we convert searching the distinguisher problem to SAT problem, we need to use automatic tool to solve it.

Initial division property. In order to search the integral distinguisher with maximum number of rounds, initial division property should contain more active bits and less constant bits. Assuming that a cipher whose block size is *n*, we set the initial division property as follows:

$$\begin{cases} \mathbb{K} = \{1\} \text{ for } i \in \{0,1,2,\cdots,n-1\}, \\ \mathbb{L} = \{l_i = 1 \text{ and } l_j = 0\ (j \neq i)\} \text{ for each } j \in \{0,\cdots,n-1\}. \end{cases} \tag{23}$$

3.3 Stopping rules

Theorem 3 (Set without integral property [7]). *Let $\mathcal{D}_{\mathbb{K}}^{n_0,n_1,\cdots,n_{m-1}}$ denote the CBDP of multiset $\mathbb{X}$. If $\mathbb{K}$ contains all n unit vectors, then the multiset $\mathbb{X}$ doesn't have integral property.*

According to Theorem 3, if $\mathbb{K}$ contains all *n* unit vectors, the multiset $\mathbb{X}$ does not have any useful integral property. The stop rule 1 is applied to $\mathbb{K}$.

Stopping Rule 1 [11]. *For the cipher E, if $\boldsymbol{k} \in \mathbb{K}$ can generate the output unit vector $\boldsymbol{e}_m$ based on the CBDP. Then according to the definition of the CBDP, we know that the integral property of m-th output bit is unknown for E.*

Stopping Rule 2 [11]. *For the cipher E, if $\boldsymbol{l} \in \mathbb{L}$ can generate the output unit vector $\boldsymbol{e}_m$ based on the CBDP, then $\boldsymbol{l}$ should be the input division property of the next part. If all vectors of $\mathbb{L}$ can not generate $\boldsymbol{e}_m$, we conclude that the integral property of m-th bit is balanced for E.*

Stopping Rule 3 [11]. *If $\mathbb{K}_{r+1,0} = \emptyset$ and $\mathbb{L}_{r+1,0} \neq \emptyset$, then there is a new integral distinguisher whose xor-sum is odd.*

Stopping rule 1 can help us find integral property of some bits is unknown, Stopping rule 2 can help us find integral property of some bits is balanced, Stopping rule 3 can find a new integral distinguisher whose xor-sum is odd.

3.4 Optimized cipher structure

For many ciphers, the size of round function is more than 32. If we get all division trails of one round based on BDPT, the computational complexity may be over 2^{32}. We should divide the cipher into small parts to decrease the computational complexity. Let Q_i denote the *i*-th round function of a cipher$E = Q_r \cdot Q_{r-1} \cdot \cdots \cdot Q_1$, then we can divide Q_i into l_i parts $Q_i = Q_{i,l_i-1} \cdot Q_{i,l_i-2} \cdot \cdots \cdot Q_{i,0}$. Therefore, we have

$$E = \prod_{i=1}^{r} \prod_{j=0}^{l_{i-1}} Q_{i,j} \tag{24}$$

Let $E_{i,j} = (Q_{i,j-1} \cdot Q_{i,j-2} \cdot \cdots \cdot Q_{i,0}) \cdot (Q_{i-1} \cdot Q_{i-2} \cdot \cdots \cdot Q_1)$ and $\overline{E_{i,j}} = (Q_r \cdot Q_{r-1} \cdot \cdots \cdot Q_{i+1}) \cdot (Q_{i,l_i-1} \cdot Q_{i,l_i-2} \cdot \cdots \cdot Q_{i,j})$. Then, the cipher E is denoted as $E = \overline{E_{i,j}} \cdot E_{i,j}$. For a cipher E, we only search all the division trails of $Q_{i,j}$. In [11], the authors give a more detailed explanation.

Example: For SIMON-32, the round function is:

$$(L_i, R_i) = ((L_{i-1}^{\lll 1} \wedge L_{i-1}^{\lll 8}) \oplus L_{i-1}^{\lll 2} \oplus R_{i-1} \oplus k_i, L_{i-1}) \tag{25}$$

Assume that the input is denoted as $(L_1 = x_0, x_1, \cdots x_{15}, R_1 = x_{16}, x_{17} \cdots x_{31})$. If we use an optimized cipher structure on the SIMON-32, then we get the following equations:

$$\begin{gathered} Q_{1,0} = (x_1 \wedge x_8 \oplus x_2 \oplus x_{16}), \\ Q_{1,1} = (x_2 \wedge x_9 \oplus x_3 \oplus x_{17}), \\ \vdots \\ Q_{1,15} = (x_0 \wedge x_7 \oplus x_1 \oplus x_{31}), \\ Q_{1,16} = (R_1 \oplus k_1, L_1). \end{gathered}$$

Then, we can reduce the computational complexity from 2^{32} to 2^4.

3.5 The method of automatic search

According to the definition of "fast propagation", we can model for the cipher. For an r-round cipher $E = \overline{E_{i,j}} \cdot E_{i,j}$, the first round of $\overline{E_{i,j}}$ may be a partial round $(Q_{i,l_i-1} \cdot Q_{i,l_i-2} \cdot \cdots \cdot Q_{i,j})$. Let input multiset $\mathbb{X}$ of $\overline{E_{i,j}}$ have the BDPT $\mathcal{D}^{1,n}_{\mathbb{K}_{i,j},\mathbb{L}_{i,j}}$, then $\mathbb{X}$ must have the division property $\mathcal{D}^{1,n}_{\mathbb{K}_{i,j} \cup \mathbb{L}_{i,j}}$. So we can use the propagation rules of CBDP to judge whether the vectors of $\mathbb{L}_{i,j}$ can generate a unit vector. If the vectors of $\mathbb{L}_{i,j}$ cannot generate a unit vector, those vectors should be eliminated according to Prunning techniques. If the vectors of $\mathbb{L}_{i,j}$ can generate a unit vector, we get all division property $\mathbb{L}_{i,j+1}$ of $Q_{i,j}$ by those vectors of $\mathbb{L}_{i,j}$ based on the propagation rules of BDPT. According to the rule of Key-XOR operation, some vectors will be added into $\mathbb{K}_{i,j+1}$ from the vectors of $\mathbb{L}_{i,j+1}$. Then $\mathcal{D}^{1,n}_{\mathbb{K}_{i,j+1},\mathbb{L}_{i,j+1}}$ is used as an input division property of $\overline{E_{i,j+1}}$. Repeating the previous steps until the results are obtained. **Fig. 1** can help the reader comprehend the "fast propagation".

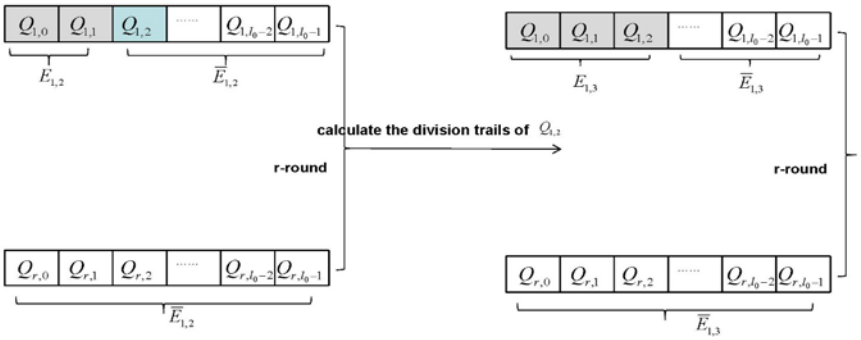

Fig. 1. The structure of "fast propagation"

3.6 Automatic search algorithm

We use the c++ interface of Cryptominisat to solve the SAT problem. The Cryptominisat is an SAT solver and can take "assumptions" as a parameter, so that we can set the initial division property and output division property. The Algorithm 2 is presented with pseudo code.

Algorithm Return all balanced bits of cipher

1: **Input:** The cipher E with size n, rounds r and initial division $\mathcal{D}^{1,n}_{\mathbb{K}_{1,0},\mathbb{L}_{1,0}}$ where $\mathbb{K}_{1,0} = \{(1,1,1,\cdots,1)\}$ and $\mathbb{L}_{1,0} = \{l | l_i = 0, l_j = 1 \text{ if} j \neq i\}$.
2: **Output:** Balanced bits.
3: **for** $bit = 0; bit \leq n-1$ **do**
4: **for** $i = 0; i \leq r-1$ **do**
5: **for** $j = 0; j \leq l_i - 1$ **do**
6: **for** $\boldsymbol{k} \in \mathbb{K}_{i,j}$ **do**
7: **if** STwoDP($\overline{E_{i,j}}$,$\boldsymbol{k}$,bit) is balanced **then**
8: $\mathbb{K}_{i,j} \rightarrow \boldsymbol{k}$
9: **end if**
10: **end for**
11: $\mathbb{L}^{'}_{i,j} = \emptyset$
12: **for** $\boldsymbol{l} \in \mathbb{L}_{i,j}$ **do**
13: **if** STwoDP($\overline{E_{i,j}}$,$\boldsymbol{l}$,bit) is balanced **then**
14: $\mathbb{L}^{'}_{i,j} = \mathbb{L}^{'}_{i,j} \cup \boldsymbol{l}$
15: **end if**
16: **end for**
17: **if** $\mathbb{L}^{'}_{i,j} = \emptyset$ **then**
18: $BalanceBit \leftarrow \boldsymbol{bit}$
19: **end if**
20: $\mathcal{D}_{\mathbb{K}_{i',j'},\mathbb{L}_{i',j'}}$ = the output division property of $Q_{i,j}$ based on three-subset division property.
21: **end for**
22: **end for**
23: **if** $\mathbb{L}^{'}_{i,j} \neq \emptyset$ **then**
24: print $\mathbb{L}^{'}_{i,j}$
25: **end if**
26: **end for**
27: return Balance bits

Algorithm 2 is explained as follows:

Line 3-5 By an optimized cipher structure, we divide the cipher E into some smaller parts, so that the division property will not propagate too many division trails.

Line 6-10 For any $\boldsymbol{k} \in \mathbb{K}_{i,j}$, we set $\boldsymbol{k}$ to be the initial division property, and judge if *bit*-th bit is balanced. If STwoDP($\overline{E_{i,j}}$,$\boldsymbol{k}$,bit) is balanced, according to Prunning techniques of $\mathbb{K}$, we

remove it from $\mathbb{K}_{i,j}$.

Line 11 We initialize $\mathbb{L}'_{i,j}$ to an empty set.

Line 12-16 For any $l \in \mathbb{L}_{i,j}$, we set $\boldsymbol{l}$ to be the initial division property, and judge if *bit*-th bit is balanced. If STwoDP($\overline{E_{i,j}}$,$\boldsymbol{l}$,bit) is balanced, according to Prunning techniques of $\mathbb{L}$, we add the $\boldsymbol{l}$ into $\mathbb{L}'_{i,j}$.

Line 17-19 If $\mathbb{L}'_{i,j}$ is an empty set, according to Stop rule 2, the bit-th bit is balanced.

Line 20-22 If we do not know the xor-sum of bit-th, we search all division trails of $Q_{i,j}$ in line with the propagation rules of BDPT. And we set $\mathcal{D}_{\mathbb{K}_{i',j'},\mathbb{L}_{i',j'}}$ to be the input division property of $\overline{E_{i,j+1}}$.

Line 23-26 After r-round, if $\mathbb{L}'_{i,j}$ is not an empty set, according to Stop rule 3, we get a new distinguisher which xor-sum is odd.

Line 27 Return all balanced bits.

4. Applications

We apply our method to some light-weight block ciphers SIMON, SIMECK, LBlock, GIFT and Khudra. All our experiments are implemented on a server with Intel(R) Core(TM) i7-8700 CPU@3.19 GHz, 64-bit windows system. When the initial division properties are different, we can run the programs in a parallel way. In our description of results, the character '?' indicates unknown, 'b' represents balanced bit, 'o' stands for odd bit. The tool developed for this paper can be available at: https://github.com/zsq123/IntegralDistinguishersSolver。

4.1 Applications to SIMON and SIMECK

SIMON [6] employs Feistel structure and it is also a lightweight block cipher. The round function of SIMON only involves simple primitives. Let SIMON-2*n* denote the SIMON family with 2*n*-bit block sizes and $n \in \{16, 24, 32, 48, 64\}$. The **Fig. 2** shows the structure of SIMON-2*n*, where (X_l, X_r) denotes the input of the round function, (X_{l+1}, X_{r+1}) denotes the output of the round function. The core operation of the round function is represented in **Fig. 3**.

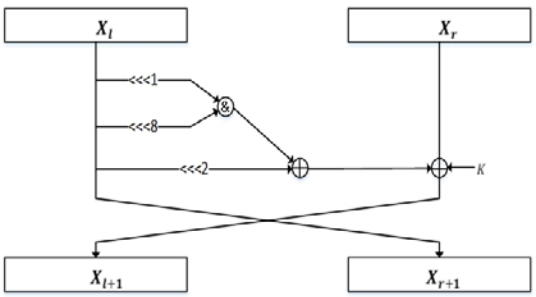

Fig. 2. The structure of SIMON-2*n*

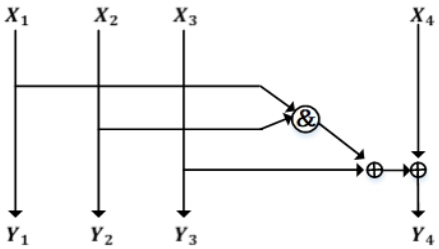

Fig. 3. The core operation of the round function

In SIMON cipher, we divide the round function of SIMON into $n+1$ parts

$$Q_i = Q_{i,n} \cdot Q_{i,n-1} \cdot \cdots \cdot Q_{i,j} \cdot \cdots \cdot Q_{i,0} \tag{26}$$

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

$$Q_{i,j} = (x^{i,j}_{(j-1) \bmod n} \& x^{i,j}_{(j-8) \bmod n}) \oplus x^{i,j}_{(j-2) \bmod n} \oplus x^{i,j}_{j} \tag{27}$$

Moreover,

$$Q_{i,n} = (X_r^{i,n} \oplus k_i, X_l^{i,n}) \tag{28}$$

where k_i is the key. When we get the output division property based on the BDPT, we only iterate through 4 bits and the other $(2n-4)$ bits remain unchanged. This can reduce the computational complexity.

Example: For Simon-32, if the input division property of $Q_{1,15}$ is$\mathcal{D}_{\mathbb{K}_{1,15}=\emptyset,\mathbb{L}_{1,15}=\{l_1\}}$, where $\boldsymbol{l}_1$ =(1, **1**, **0**, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1 , 1, 1, **0**, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1), the 15-th core operation division property of Simon-32 has 4 bits. And the division property is (1,1,0,0). Then according to the propagation rules of BDPT, the division property of the output multiset is (1,1,0,0), (0,0,0,1), (1,0,0,1), (0,1,0,1), (1,1,0,1), so the propagation from $\boldsymbol{l}_1$ generates five vectors as:

$\boldsymbol{l}_2$=(1, **1**, **0**, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, **0**, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_3$=(1, **0**, **0**, 1, 1, 1, 1, 1, **0**, 1, 1, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_4$=(1, **1**, **0**, 1, 1, 1, 1, 1, **0**, 1, 1, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_5$=(1, **0**, **0**, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_6$=(1, **1**, **0**, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, **1**, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1, 1). The output division property of $Q_{1,15}$ is $\mathcal{D}_{\mathbb{K}_{1,16}=\emptyset,\mathbb{L}_{1,16}=\{\boldsymbol{l}_2,\boldsymbol{l}_3,\boldsymbol{l}_4,\boldsymbol{l}_5,\boldsymbol{l}_6\}}$.

Due to $Q_{1,16}$ has Key-XOR operation, new vectors are generated from $\mathbb{L}_{1,16}$ according to Rule 4, and some vectors in $\mathbb{L}_{1,16}$ will become redundant because of the new vectors of $\mathbb{K}_{1,16}$, so $\mathcal{D}_{\mathbb{K}_{1,16}=\{\boldsymbol{k}_1\},\mathbb{L}_{1,16}=\{\boldsymbol{l}_7,\boldsymbol{l}_8,\boldsymbol{l}_9,\boldsymbol{l}_{10}\}}$, where

$\boldsymbol{k}_1$=(1,1,1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_7$=(0, 1,1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_8$=(1, 1,1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_9$=(1, 1,1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1),
$\boldsymbol{l}_{10}$=(1, 1,1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1).

We use $\mathcal{D}_{\mathbb{K}_{1,16}=\{\boldsymbol{k}_1\},\mathbb{L}_{1,16}=\{\boldsymbol{l}_7,\boldsymbol{l}_8,\boldsymbol{l}_9,\boldsymbol{l}_{10}\}}$ as input division property of $\overline{E_{2,0}}$ based on CBDP. For $\mathbb{K}_{1,16}$, if there exists $\boldsymbol{k} \in \mathbb{K}_{1,16}$ that generates the unit vector $\boldsymbol{e}_m$, then the integral property of the *m*-th bit is unknown. Otherwise, we empty the elements of $\mathbb{K}_{1,16}$ in the light of Prunning techniques of $\mathbb{K}$. For $\mathbb{L}_{1,16}$, if there is no element of $\mathbb{L}_{1,16}$ that can generate the unit vector $\boldsymbol{e}_m$ based on CBDP, then the integral property of *m*-th bit is balanced and the program terminates. If the element of $\mathbb{L}_{1,16}$ that can generate the unit vector $\boldsymbol{e}_m$ based on CBDP, then we use the vectors of $\mathbb{L}_{1,16}$ that can generate a unit vector as input division property of $Q_{2,0}$ and find all division trails based on the BDPT. Next repeat the previous steps until the program terminates.

For SIMON-32, when the data complexity is 2^{31} chosen plaintexts, we obtain a 14-round integral distinguisher as follows.

SIMON-32: $(7fff, ffff) \xrightarrow{14r}$(????, ????, ????, ????, ?b??, ????, b???, ???b).

Although we only get a 14-round integral distinguisher, we can get a 15-round integral distinguisher using the technique in [14] . When we apply the same method to SIMON-48/-64/-96, the results for SIMON-48/-64/-96 based on BDPT are the same as before.

SIMECK [15] is another lightweight block cipher and it is very similar to the SIMON except the rotation constants, the rotation constants of SIMECK is (0,5,1). The round function of SIMECK is denotes as:

$$(L_i, R_i) = (L_{i-1} \wedge L_{i-1}^{\lll 5} \oplus L_{i-1}^{\lll 1} \oplus R_{i-1} \oplus k_i, L_{i-1}) \tag{29}$$

Let SIMECK-2n denote the SIMECK block ciphers, where 2n is the length of cipher and *n* is chosen from 16, 24 and 32. We apply our automatic search algorithm to SIMECK-2n and get 15-/18-/21-round distinguishers for SIMECK-32/-48/-64, respectively. Unfortunately, our results are the same as those found by Xiang *et al*. [7].

In [12], the authors propose a variant of SIMON family and they name it as SIMON(102). SIMON(102) alters the rotation from (1, 8, 2) to (1, 0, 2). For SIMON(102)-32, we can find xor-sum of some bits are odd, it is more accurate than [10] that finds some bits are constant. In our algorithm, in the output of the 19-round the $\mathbb{L}$ set is not empty, the vectors of $\mathbb{L}$ is: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1) and (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). It shows that the parity of the 1-st bit and the 15-th bit are odd.

$$\text{SIMON(102)-32}: (7fff, ffff) \xrightarrow{19r} (????, ????, ????, ????, bo??, ????, ????, ???o).$$

Similarly, for SIMON(102)-48 and SIMON(102)-64, we also find more accurate integral distinguishers. We list them as follows:

$$\text{SIMON(102)-48}: (7ffffff, ffffff) \xrightarrow{27r}$$
$$(????, ????, ????, ????, ????, ????, bo??, ????, ????, ????, ????, ???o).$$
$$\text{SIMON(102)-64}: (7fffffff, ffffffff) \xrightarrow{35r}$$
$$(????, ????, ????, ????, ????, ????, ????, ????, bo??, ????, ????, ????, ????, ????, ????, ???o).$$

To test the validity of our model, we do some experiment on SIMON(102)-32. We randomly generate 20 secret keys and exhaustively search odd or balanced bits. We find that all odd/balanced bits found by model are indeed odd/balanced bits. As a result, our new model is valid.

4.2 Application to GIFT

GIFT [16] is an SPN structure block cipher. Its linear layer consists of bit permutation. It is very similar to PRESENT. GIFT has two versions GIFT-64 which has 28-round with a block size of 64 bits and GIFT-128 which has 40-round with a block size of 128 bits. One round of GIFT-64 is shown in **Fig. 4**.

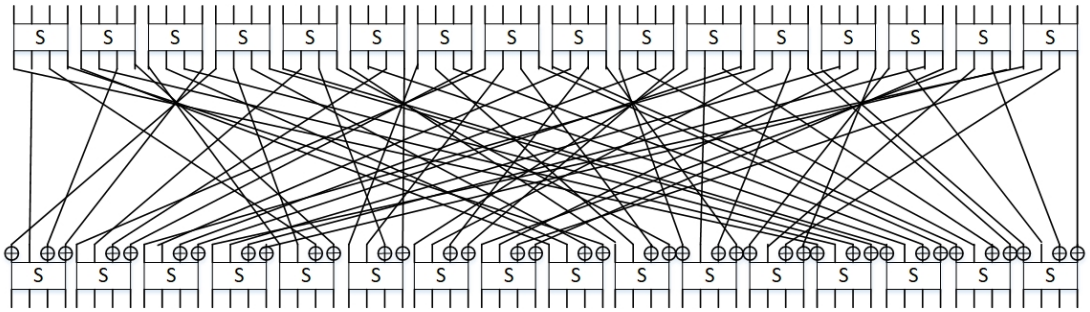

Fig. 4. One round function of GIFT-64

For GIFT-64, we divide the round function into 17 parts:

$$Q_i = Q_{i,16} \cdot Q_{i,15} \cdot \cdots \cdot Q_{i,j} \cdot \cdots \cdot Q_{i,0} \tag{30}$$

When $0 \leq j \leq 15$, we have

$$Q_{i,j} = S(X_j) \tag{31}$$

where X is the input of the round function and $X = X_{15}||\cdots||X_0$.

When $j - 16$, we have

$$Q_{i,16} = P(Sbox(X)) \oplus k_i \tag{32}$$

where P is the linear permutation function of GIFT, $Sbox(X)$ is the output of all S-boxes and k_i is the i-th round key.

We apply our algorithm to GIFT, the round of integral distinguisher that we find is the same as before. However, our result can find more balanced bits. When we input 2^{63} chosen plaintexts, we find a new integral distinguisher which has 2 more balanced bits than before. We list the result as follows:

$\text{GIFT-64} : (ffffffff, fffffffd) \xrightarrow{9r}$
$(bb??, bb??, bb??, bb??, bb??, bb?, bb??, bb??, bb??, bb??, bb??, bb??, bb??, bb??, bb??, bb??).$

When we input 2^{61} chosen plaintexts, we find a new integral distinguisher which has 4 more balanced bits than before. We list the result as follows:

$\text{GIFT-64} : (ffffffff, fffffff4) \xrightarrow{9r}$
$(b???, b???, ????, ????, ????, b???, b???, b???, b???, b???, ????, ????, ????, ????, b???, b???).$

4.3 Application to LBlock

LBlock [17] is a Feistel block cipher. The sizes of block and key are 64 and 80 respectively. And it uses 8 different S-boxes in the round function. The round function of LBlock is illustrated in **Fig. 5**.

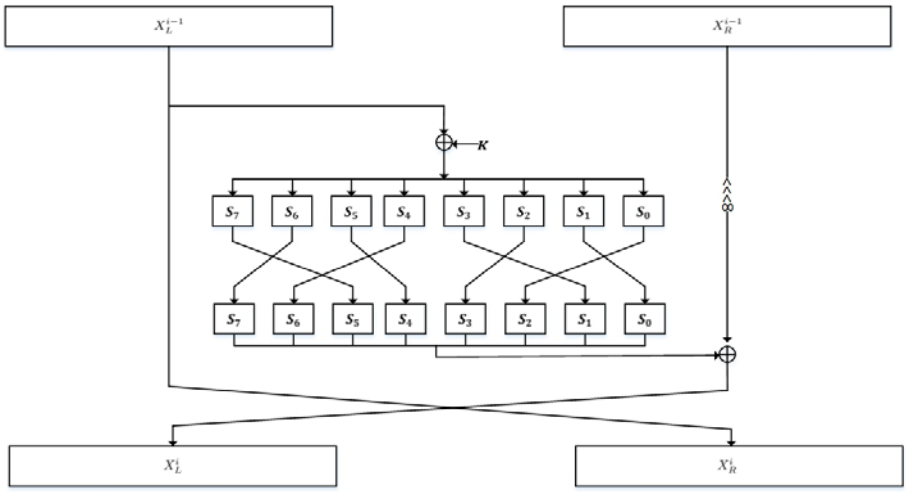

Fig. 5. One round function of LBlock

We divide the round function Q_i into 9 parts

$$Q_i = Q_{i,8} \cdot Q_{i,7} \cdot \cdots \cdot Q_{i,0} \tag{33}$$

When $0 \leq j \leq 7$, we have

$$Q_{i,j} = S_j(x_j^i \oplus k^i) \oplus y_j^i \tag{34}$$

where $S_j(x_j^i \oplus k^i)$ is the output of the j-th S-box.

When $j = 8$, we have

$$Q_{i,8} = P(X_i) \tag{35}$$

where P is the linear permutation function of the LBlock and $X_i = (S_0(x_0^i \oplus k^i) \oplus y_0^i)||(S_1(x_1^i \oplus k^i) \oplus y_1^i)|| \cdots ||(S_7(x_7^i \oplus k^i) \oplus y_7^i)$.

We apply our attack algorithm to LBlock. When we input 2^{63} chosen plaintexts, four integral distinguishers are discovered, which are listed as follows:

$\text{LBlock-64} : (ffffffffb, fffffffff) \xrightarrow{17r}$

$(????, ????, ????, ????, ????, ????, ????, ????, bb??, ????, ????, ????, ????, ????, ????, bb??)$.

LBlock-64 : $(fffffffbf, fffffffff) \xrightarrow{17r}$
$(????, ????, ????, ????, ????, ????, ????, ????, ????, ????, ????, ????, ????, bb??, b?b?, ????)$.

LBlock-64 : $(ffffbfff, fffffffff) \xrightarrow{17r}$
$(????, ????, ????, ????, ????, ????, ????, ????, ????, ????, ????, b?b?, ?b?b, ????, ????, ????)$.

LBlock-64 : $(fbffffff, fffffffff) \xrightarrow{17r}$
$(????, ????, ????, ????, ????, ????, ????, ????, ????, b??b, bb??, ????, ????, ????, ????, ????)$.

The first one is the same as the 17-round distinguisher in [13], which is obtained under CBDP.

4.4 Application to Khudra

Khudra [18] is a lightweight block cipher using “Generalized type-2 transformations" of Feistel Structure (GFS) [19] with 64-bit block size. It has 18-round and 80-bit keys.

For Khudra, the input of the first round is calculated as $X_0 = P_3||(P_2 \oplus wk_2)||P_1||(P_0 \oplus wk_1)$, where the plaintext $P = P_3||P_2||P_1||P_0$ and wk is the whitening-key. Next, X_0 is encrypted by the round function. The input of the i-th round is denoted as X_{i-1}. Last, the ciphertext C is calculated as $C = X_{18,0}||(X_{18,1} \oplus wk_3)||X_{18,2}||(X_{18,3} \oplus wk_4)$, where X_{18} is the output of the 18 -th round function and $X_{18} = X_{18,3}||X_{18,2}||X_{18,1}||X_{18,0}$. The structure of Khudra is demonstrated in **Fig. 6**.

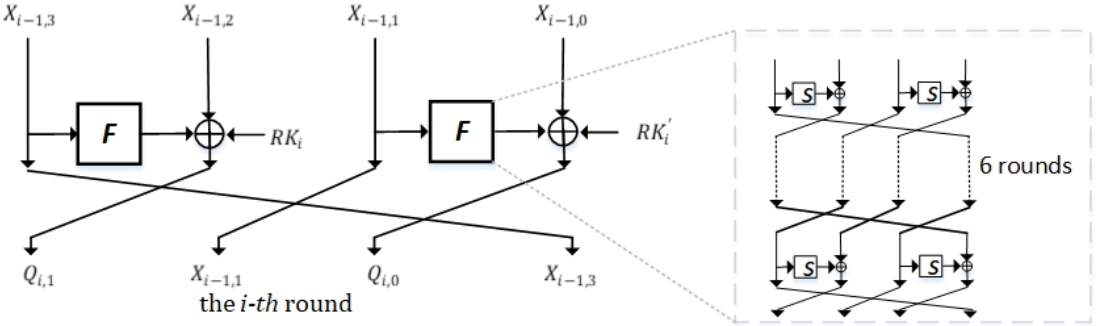

Fig. 6. The structure of Khudra

We divide the round function Q_i into 3 parts

$$Q_i = Q_{i,2} \cdot Q_{i,1} \cdot Q_{i,0} \tag{36}$$

Here

$$Q_{i,0} = F(X_{i-1,1}) \oplus RK_i' \oplus X_{i-1,0},$$
$$Q_{i,1} = F(X_{i-1,3}) \oplus RK_i \oplus X_{i-1,2},$$
$$Q_{i,2} = X_i = Per(Q_{i,1}||X_{i-1,1}||Q_{i,0}||X_{i-1,3}),$$

where F is the round function which has an iterated structure based on a 6-round GFS, Per is a linear permutation. If there exist $\boldsymbol{k} \in \mathbb{K}$ that can generate the unit vector $\boldsymbol{e}_m$, then the m-th bit must be unknown in line with the definition of division property. So if there exist $\boldsymbol{k}$ that can decide m-th is unknown based on CBDP, we do not need to judge all vectors of $\mathbb{K}$.

Khudra includes the initial and the final transformations which contain the whitening-key. It is note that the distinguishers we found in this paper for Khudra does not take account of the above two transformations. When the data complexity is 2^{63} chosen plaintexts, we find two integral distinguishers which are shown as follows:

Khudra-64 : $(effffffff, fffffffff) \xrightarrow{9r}$

$(????, ????, ????, ????, ????, ????, ????, ????, ????, ????, ????, ????, bbbb, bbbb, bbbb, bbbb).$
$\text{Khudra-64}: (ffffffff, effffffff) \xrightarrow{9r}$
$(????, ????, ????, ????, bbbb, bbbb, bbbb, bbbb, ????, ????, ????, ????, ????, ????, ????, ????).$

5. Conclusions

In this paper, we construct an automatic model to search the integral characteristics and solve the complexity problem of searching integral distinguishers by converting the searching integral distinguisher problem into an SAT problem. First, we model the propagations of BDPT in CNF-formulas. Then we give the construction of SAT models of the S-box. Furthermore, we construct an algorithm which can detect the integral distinguishers efficiently and can help us get all balanced bits based on the BDPT.

We apply our automatic model to some block ciphers, such as SIMON-32/-48/-64/-96, SIMECK-32/-48/-64, SIMON(102)-32/-48/-64, GIFT, LBlock and Khudra. For SIMON-2n and SIMECK-2n, our results are the same as before. For SIMON(102)-32/-48/-64, we can find 20-/28-/36-round integral distinguisher. Although the rounds of integral distinguisher are the same as before for SIMON(102)-32/-48/-64, we can determine some bits are odd, while these bits can only be determined as constant in the previous result. For GIFT, our 9-round distinguisher has more balanced bits than previous longest integral distinguisher. With the same number of rounds as before, we can obtain more integral distinguishers on 17-round LBlock. For Khudra, we get the first 9-round integral distinguisher. As a result, our result show that our model is powerful in finding integral distinguishers for block ciphers.

Acknowledgements

This research is supported by the National Cryptography Development Fund (No. MMJJ20180201), the National Natural Science Foundation of China (No. 62072181), International Science and Technology Cooperation Projects (No. 61961146004) and the Fundamental Research Funds for the Central Universities.

References

[1] Y. Todo, “Structural evaluation by generalized integral property,” in *Proc. of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, vol. 9056, pp. 287-314, April, 2015. Article (CrossRef Link).

[2] L. R. Knudsen and D. Wagner, “Integral cryptanalysis,” in *Proc. of International Workshop on Fast Software Encryption* 2002, vol. 2365, pp. 112-127, July, 2002. Article (CrossRef Link).

[3] Y. Todo, “Integral cryptanalysis on full MISTY1,” *Journal of Cryptology*, vol.30, pp.920–959, 2017. Article (CrossRef Link).

[4] H. Zhang and W. Wu, “Structural evaluation for generalized Feistel structures and applications to LBlock and TWINE,” in *Proc. of Cryptology -- INDOCRYPT 2015*, vol. 9462, pp. 218-237, November, 2015. Article (CrossRef Link).

[5] Y. Todo and M. Morii, “Bit-based division property and application to Simon family,” in *Proc. of the 23rd International Conference, FSE 2016*, vol. 9783, pp. 357-377, July, 2016. Article (CrossRef Link).

[6] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” in *Proc. of the 52nd Annual Design Automation Conference*, pp. 1-6, June, 2015. Article (CrossRef Link).

[7] Z. Xiang, W. Zhang, Z. Bao, and D. Lin, "Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers," in *Proc. of ASIACRYPT 2016*, vol. 10031, pp. 648-678, November, 2016. Article (CrossRef Link).
[8] L. Sun, W. Wang, and M. Wang, "Automatic search of bit-based division property for ARX ciphers and word-based division property," in *Proc. of ASIACRYPT 2017*, vol. 10624, pp.128-157, November, 2017. Article (CrossRef Link).
[9] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. of the third annual ACM symposium on Theory of computing*, pp. 151-158, May, 1971. Article (CrossRef Link).
[10] K. Hu and M. Wang, "Automatic search for a variant of division property using three subsets," in *Proc. of CT-RSA 2019*, vol. 11405, 2019, pp. 412-432, February, 2019. Article (CrossRef Link).
[11] S. Wang, B. Hu, J. Guan, K. Zhang, and T. Shi, "Milp aided method of searching division property using three subsets and applications," in *Proc. of ASIACRYPT 2019*, vol. 11923, pp. 398-427, November, 2019. Article (CrossRef Link).
[12] S. K*ö*lbl, G. Leander, and T. Tiessen, "Observations on the SIMON block cipher family," in *Proc. of CRYPTO 2015*, vol. 9215, pp. 161-185, August 2015. Article (CrossRef Link).
[13] Z. Eskandari, A. B. Kidmose, S. K*ö*lbl, and T. Tiessen, "Finding integral distinguishers with ease," in *Proc. of Selected Areas in Cryptography – SAC 2018*, vol. 11349, pp. 115-138, January, 2019. Article (CrossRef Link).
[14] Q. Wang, Z. Liu, K. Varici, Y. Sasaki, V. Rijmen, and Y. Todo, "Cryptanalysis of reduced-round SIMON32 and SIMON48," in *Proc. of INDOCRYPT 2014*, vol. 8885, pp. 143-160, October, 2014. Article (CrossRef Link).
[15] G. Yang, B. Zhu, V. Suder, M. Aagaard, and G. Gong, "The simeck family of lightweight block ciphers," in *Proc. of Cryptographic Hardware and Embedded Systems 2015*, vol. 9293, pp. 307-329, September, 2015. Article (CrossRef Link).
[16] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: a small PRESENT - towards reaching the limit of lightweight encryption," in *Proc. of Cryptographic Hardware and Embedded Systems* 2017, vol. 10529, pp. 321-345, January, 2017. Article (CrossRef Link).
[17] W. Wu and L. Zhang, "LBlock: a lightweight block cipher," in *Proc. of Applied Cryptography and Network Security 2011*, vol. 6715, pp. 327-344, June, 2011. Article (CrossRef Link).
[18] S. Kolay and D. Mukhopadhyay, "Khudra: A new lightweight block cipher for FPGAs," in *Proc. of Security, Privacy, and Applied Cryptography Engineering*, vol. 8804, pp. 126-145, October, 2014. Article (CrossRef Link).
[19] V. T. Hoang and P. Rogaway, "On generalized Feistel Networks," in *Proc. of CRYPTO 2010*, vol. 6223, pp. 613-630, August, 2010. Article (CrossRef Link).

Shiqiang Zhu received the B.S. degree in software engineering from Shanghai Ocean University in 2017 and the M.S. degree in information security from East China Normal University in 2020. His research interests include cryptanalysis of symmetric-key ciphers.

Gaoli Wang received the B.S. degree in fundamental mathematics in 2003, and the Ph.D. degree in information security in 2008, both from Shandong University. She is currently a Professor in Software Engineering Institute, East China Normal University. Her research interests include cryptography, computer and network security.

Yu He received the B.S. degree in computer science and technology from Henan Agricultural University in 2018. She is currently pursuing the M.S. degree with East China Normal University. Her research interests include cryptanalysis of symmetric-key ciphers.

Haifeng Qian received the B.S. degree in mathematics in 2000, and the M.S. degree in mathematics in 2003, both in East China Normal University. He received the Ph.D. degree in information security in 2006 from Shanghai Jiaotong University. He is currently a Professor in Software Engineering Institute, East China Normal University. His research interests include cryptography, computer and network security.