Automatic Search for a Variant of Division Property Using Three Subsets

Kai Hu and Meiqin Wang(✉)

Key Laboratory of Cryptologic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China
hukai@mail.sdu.edu.cn, mqwang@sdu.edu.cn

Abstract. The division property proposed at Eurocrypt'15 is a novel technique to find integral distinguishers, which has been applied to most kinds of symmetric ciphers such as block ciphers, stream ciphers, and authenticated encryption, *etc.* The original division property is word-oriented, and later the bit-based one was proposed at FSE'16 to get better integral property, which is composed of conventional bit-based division property (two-subset division property) and bit-based division property using three subsets (three-subset division property). Three-subset division property has more potential to achieve better integral distinguishers compared with the two-subset division property. The bit-based division property could not be to apply to ciphers with large block sizes due to its unpractical complexity. At Asiacrypt'16, the two-subset division property was modeled using Mixed Integral Linear Programming (MILP) technique, and the limits of block sizes were eliminated. However, there is still no efficient method searching for three-subset division property. The propagation rule of the `XOR` operation for $\mathbb{L}$ (The definition of $\mathbb{L}$ and $\mathbb{K}$ is introduced in Sect. 2.), which is a set used in the three-subset division property but not in two-subset one, requires to remove some specific vectors, and new vectors generated from $\mathbb{L}$ should be appended to $\mathbb{K}$ when `Key-XOR` operation is applied, both of which are difficult for common automatic tools such as MILP, SMT or CP. In this paper, we overcome one of the two challenges, concretely, we address the problem to add new vectors into $\mathbb{K}$ from $\mathbb{L}$ in an automatic search model. Moreover, we present a new model automatically searching for a variant three-subset division property (VTDP) with STP solver. The variant is weaker than the original three-subset division property (OTDP) but it is still powerful in some ciphers. Most importantly, this model has no constraints on the block size of target ciphers, which can also be applied to ARX and S-box based ciphers. As illustrations, some improved integral distinguishers have been achieved for SIMON32, SIMON32/48/64(102), SPECK32 and KATAN/KTANTAN32/48/64 according to the number of rounds or number of even/odd-parity bits.

Keywords: Division property · Three-subset · STP · Automatic research

M. Matsui (Ed.): CT-RSA 2019, LNCS 11405, pp. 412–432, 2019.
https://doi.org/10.1007/978-3-030-12612-4_21

1 Introduction

Division property, a generalization of the integral property [6], was proposed by Todo at Eurocrypt'15 [12], which has been applied to most kinds of symmetric ciphers, such as block ciphers, stream ciphers and authenticated encryption [13, 14], *etc.* The most impressive application is that it was used to break, for the first time, the full MISTY1 at CRYPTO'15 [13]. Furthermore, the division property made significant progress in the cube attack because the limits of practical data complexity have been eliminated [14].

Since the division property was put forward, this cryptanalytic technique has been further investigated. The original division property [12] is word-oriented, and it can only describe the algebraic degree of `S-box` instead of the particular Boolean function. In order to further consider the Boolean function of `S-box`, Boura *et al.* gave more precise description for `S-box` in division property at CRYPTO'16 [3].

At FSE'16, Todo and Morii [15] introduced the bit-based division property which depicts the components of target primitive at bit level so that more information of the cipher structures can be utilized. Compared with the original word-level division property, the bit-based one is more likely to find better integral characteristics. Bit-based division property family proposed in [15] includes two-subset and three-subset division property. The two- and three-subset division property classify all vectors $\boldsymbol{u} \in \mathbb{F}_2^n$ into two and three subsets, respectively, according to the parity of a Boolean polynomial related to $\boldsymbol{u}$. In detail, the parity is even or unknown for two-subset division property while even, odd or unknown for three-subset division property. Because the odd-parity set is extracted from the unknown set in three-subset division property, it means that more information of Boolean function is traced. Therefore, three-subset division property has more potential to achieve better integral distinguishers. For example, the 14-round integral characteristic of SIMON32 has been found by two-subset division property while 15-round integral characteristic was found by three-subset division property [15].

Although the bit-based division property under Todo and Morri's framework is quite effective to find integral distinguishers, unfortunately, they can only work on ciphers with small block sizes because of the huge memory and time requirements. As pointed in [15], for a cipher with block size n, the time and memory complexities are upper bounded by 2^n. Xiang *et al.* have solved the problem of searching for two-subset division property by utilizing the MILP tools at Asiacrypt'16 [17]. They transformed the search problem into an MILP problem which can be used to find division property for ciphers with large block size. Automatic tools such as MILP solvers can describe the set with some constraints and conduct some inner optimization automatically, which do not need to go through all the vectors. Xiang *et al.*'s method has been extended and applied to improve the integral attacks on many ciphers [5, 9, 10, 16]. Especially, the MILP model to search division property was used to extend the cube attack, which has improved the attacks on Trivium, Grain128a, and Acorn [14].

Since the automatic search model for three-subset division property is still not constructed, it can be merely used on ciphers with small size until now. For two-subset division property, we only trace the set $\mathbb{K}$ but both the set $\mathbb{K}$ and $\mathbb{L}$ should be considered for three-subset division property. There are two challenges to face when we construct the automatic search model by MILP, SMT or CP. In one hand, the propagation rules for $\mathbb{L}$ are very different because some vectors which appear an even number of times should be removed from $\mathbb{L}$ and the propagation rule of XOR should remove the vectors occurring an even number of times, too. On the other hand, some new vectors generated from vectors in $\mathbb{L}$ will be added into $\mathbb{K}$.

In common MILP, SMT or CP models, the constraints are used only to narrow the range of the sets which the variables belong to. There are no direct methods which can solve the two following problems as far as we know,

1. decide the duplicated vectors which appear even times and remove them dynamically.
2. extend the range of a set which the specific variable belongs to.

In this paper, we introduce one new technique by an STP solver to overcome the second problem directly. We do not remove the duplicated vectors in $\mathbb{L}$ and then we get a variant of three-subset division property. Although VTDP is not more efficient than OTDP, we prove that the results of VTDP are valid and useful. Most importantly, we can automatically search for VTDP without the limits of block sizes. It can also be applied to S-box based and ARX ciphers.

1.1 Our Contributions

1.1.1 Automatic Search Algorithm for VTDP

In this paper, we introduce VTDP and construct a general model of automatic search for it. The details of our technical contributions are three-fold, which are listed as follows.

VTDP and Variant Three-Subset Division Trail. We describe the method to obtain VTDP from OTDP and prove the validity of this variant. Compared with OTDP, VTDP does not remove any duplicated vector in $\mathbb{L}$ and modify the propagation rule of XOR for $\mathbb{L}$. As a result, we can prove that the integral distinguishers found by VTDP are valid according to OTDP. To construct the automatic search model for VTDP, we introduce the definition of variant three-subset division trail. The definition of division trail to illustrate the propagation of two-subset division property is introduced in [17]. Similarly, we define the variant three-subset division trail in order to construct the automatic search model for VTDP. With this definition, the problem of searching for VDTP can be transformed to a problem of searching for a valid variant three-subset division trail.

Table 1. Results of VTDP for some ciphers

Cipher	Data	Round	Number of even/odd-parity bits	Time	Reference
SIMON32	2^{31}	14	32		[17]
		15	**3**	**27 s**	[15], Sect. 4.1
SIMON32(102)	2^{31}	20	1		[17]
		20	**3**	**25 s**	Sect. 4.1
SIMON48(102)	2^{47}	28	1		[17]
		28	**3**	**9.3 s**	Sect. 4.1
SIMON64(102)	2^{63}	36	1		[17]
		36	**3**	**1.1 h**	Sect. 4.1
KATAN/KTANTAN32	2^{31}	99	1		[9]
		101	**1**	**5.6 h**	Sect. 4.4
KATAN/KTANTAN48	2^{47}	63.5	1		[9]
		64	**1**	**16 h**	Sect. 4.4
KATAN/KTANTAN64	2^{63}	72.3	1		[9]
		72.3	**2**	**18 h**	Sect. 4.4
SPECK32	2^{31}	6	1		[11]
		6	**2**	**3.5 m**	Sect. 4.2

Models of Key-Independent Components for $\mathbb{L}$. To search for VTDP, we should build the models for propagation for $\mathbb{K}$ and $\mathbb{L}$. For $\mathbb{K}$, the models are the same as those in the two-subset division property [11,17], which can be referred directly. However, we should construct the models of all kinds of operations for $\mathbb{L}$. We first give a variant propagation rule of `XOR` for $\mathbb{L}$ and construct the automatic search models for common component such as `Copy`, `AND` and `XOR`. Then, to make our models more general, we consider `Modular Addition` and `S-box` also.

Model for `Key-XOR`. The difficult problem in constructing the models for VTDP is how to update the set $\mathbb{K}$ with the set $\mathbb{L}$ when a `Key-XOR` operation is applied to the state. By introducing the logical OR operation in STP, which is a simple but efficient solver for the theory of quantifier-free bit vectors, we succeed to solve this difficult problem. Thus, we can give a model for `Key-XOR` based on STP.

1.1.2 Applications

We apply our model to search for integral distinguishers of SIMON [1], SIMECK [18], SIMON(102) [7], SPECK [1], KATAN/KTANTAN [4]. The results are shown in Table 1. Note our model are also suitable to the ciphers with larger size and the `S-box` based ciphers but no better results can be obtained.

1.2 Organization of the Paper

We briefly recall some background knowledge about the bit-based division property in Sect. 2. In Sect. 3, we introduce VTDP and construct the whole automatic search model for it. We show some applications of our model in Sect. 4. At last, we conclude the paper in Sect. 5.

2 Preliminaries

2.1 Bit-Based Division Property

At Eurocrypt'15, the division property, a generalization of the integral property, was proposed [12], where better integral distinguishers for word-oriented cryptographic primitives have been detected. Later, Todo and Morii introduced the bit-based division property [15] where the propagation of integral characteristic can be described in a more dedicated manner for the concrete structures of the target primitives. As a result, more rounds of integral characteristics have been found with this new technique. For example, the integral distinguishers of SIMON32 have been improved from 10-round to 15-round.

Bit-based division property traces the propagation of vectors $\boldsymbol{u} \in \mathbb{F}_2^n$ according to the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ for all $\boldsymbol{x}$, where $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is a polynomial $\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \Pi_i x_i^{u_i}$ and x_i, u_i are the i-th bit of vector $\boldsymbol{u}$ and $\boldsymbol{v}$. For the traditional bit-based division property, only two cases are considered where $\boldsymbol{u}$ can be classified into two sets according to that the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is even or unknown. In this paper, we name it as *two-subset bit-based division property.*

Definition 1 (Two-Subset Bit-Based Division Property [15]**).** *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ be a set whose elements take an n-dimensional bit vector. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, it fulfils the following conditions:*

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

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

The two-subset bit-based division property uses the set $\mathbb{K}$ to represent the subset of $\boldsymbol{u}$ such that the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is unknown. According to [15], the two-subset bit-based division property is insufficient to find more accurate integral characteristic because it cannot exploit the fact that the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is definitely odd. Motivated by this fact, the three-subset bit-based division property is introduced in [15].

The three-subset bit-based division property classifies $\boldsymbol{u}$ into three sets on the basis of what the parity of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ is unknown, definitely even or odd. Therefore, the set $\mathbb{K}$ is used to represent the set of $\boldsymbol{u}$ with unknown $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$, and the set $\mathbb{L}$ is used to denote the set of $\boldsymbol{u}$ with $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$ equal to one.

Definition 2 (Three-Subset Bit-Based Division Property [15]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ and $\mathbb{L}$ be two sets whose elements take n-dimensional bit vectors. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^n}$, it fulfils the following conditions:*

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

According to [15], if there are $\boldsymbol{k}\in\mathbb{K}$ and $\boldsymbol{k}'\in\mathbb{K}$ satisfying $\boldsymbol{k}\succeq\boldsymbol{k}'$, then $\boldsymbol{k}$ is redundant. Moreover, if there are $\boldsymbol{l}\in\mathbb{L}$ and $\boldsymbol{k}\in\mathbb{K}$, the vector $\boldsymbol{l}$ is also redundant if $\boldsymbol{l}\succeq\boldsymbol{k}$. The redundant vectors in $\mathbb{K}$ and $\mathbb{L}$ will not affect the parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ for any $\boldsymbol{u}$.

Since we only focus on the bit-based division property in this paper, all notations of division property is for the bit level by default if we do not declare it.

Propagation Rules

Those for $\mathbb{K}$ are the same as those of two-subset one.

Rule 1 (Copy [15]). *Let F be a copy function, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^n$, and the output is calculated as $(x_1, x_1, x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multiset, respectively. Assume that $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^m}$, $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^{m+1}}$, where $\mathbb{K}'$ and $\mathbb{L}'$ are computed as*

$$\mathbb{K}' \leftarrow \begin{cases} (0,0,k_2,\ldots,k_m), & \textit{if } k_1 = 0 \\ (1,0,k_2,\ldots,k_m),(0,1,k_2,\ldots,k_m), & \textit{if } k_1 = 1 \end{cases},$$

$$\mathbb{L}' \leftarrow \begin{cases} (0,0,l_2,\ldots,l_m), & \textit{if } l_1 = 0 \\ (1,0,l_2,\ldots,l_m),(0,1,l_2,\ldots,l_m),(1,1,l_2,\ldots,l_m), & \textit{if } l_1 = 1 \end{cases}.$$

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

Rule 2 (AND [15]). *Let F be a function compressed by an AND, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1 \wedge x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multiset, respectively. Assume that $\mathbb{X}$ has $\mathcal{D}_{\mathbb{K},\mathbb{L}}^{1^m}$, $\mathbb{Y}$ has $\mathcal{D}_{\mathbb{K}',\mathbb{L}'}^{1^{m-1}}$, where $\mathbb{K}'$ is computed from $\boldsymbol{k}\in\mathbb{K}$ as*

$$\mathbb{K}' \leftarrow \left(\left\lceil \frac{k_1+k_2}{2} \right\rceil, k_3, k_4, \ldots, k_m \right).$$

Moreover, $\mathbb{L}'$ is computed from $\boldsymbol{l}\in\mathbb{L}$ s.t. $(l_1, l_2) = (0,0)$ or $(1,1)$ as

$$\mathbb{L}' \leftarrow \left(\left\lceil \frac{l_1+l_2}{2} \right\rceil, l_3, l_4, \ldots, l_m \right).$$

Rule 3 (XOR [15]). *Let F be a function compressed by an XOR, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1 \oplus x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multiset, respectively. Assume that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $\mathbb{Y}$ has $\mathcal{D}^{1^{m-1}}_{\mathbb{K}',\mathbb{L}'}$, where $\mathbb{K}'$ is computed from $\boldsymbol{k} \in \mathbb{K}$ s.t. $(k_1, k_2) = (0,0), (1,0),$ or $(0,1)$ as*

$$\mathbb{K}' \leftarrow (k_1 + k_2, k_3, k_4, \ldots, k_m).$$

Moreover, $\mathbb{L}'$ is computed from $\boldsymbol{l} \in \mathbb{L}$ s.t. $(l_1, l_2) = (0,0), (1,0),$ or $(0,1)$ as

$$\mathbb{L}' \overset{x}{\leftarrow} (l_1 + l_2, l_3, l_4, \ldots, l_m),$$

where $\mathbb{L} \overset{x}{\leftarrow} \boldsymbol{l}$ means

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

Boura *et al.* presented the propagation rules of S-box for $\mathbb{K}$ at bit-level in [3] for the first time. We summarize the technique in Rule 4.

Rule 4 (Bit-Based S-box for $\mathbb{K}$ [3]). *Let $F : \mathbb{F}_2^m \to \mathbb{F}_2^n$ be a function of substitution composed of $(f_1, f_2, \ldots, f_n)$, where the input $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ is calculated as*

$$\begin{aligned} y_1 &= f_1(x_1, x_2, \ldots, x_m), \\ y_2 &= f_2(x_1, x_2, \ldots, x_m), \\ &\vdots \\ y_n &= f_n(x_1, x_2, \ldots, x_m). \end{aligned}$$

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

Modular Addition is the nonlinear component of ARX ciphers. The Modular Addition operation can be decomposed into a series of basic operations such as Copy, AND and XOR. Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$, $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ and $\boldsymbol{z} = (z_0, z_1, \ldots, z_{n-1})$. If $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$, the Boolean function of z_i can be iteratively expressed as follows,

$$\begin{aligned} & z_{n-1} = x_{n-1} \oplus y_{n-1} \oplus c_{n-1}, c_{n-1} = 0, \\ & z_i = x_i \oplus y_i \oplus c_i, c_i = x_{i+1} \cdot y_{i+1} \oplus (x_{i+1} \oplus y_{i+1}) \cdot c_{i+1}, \\ & i = n-2, n-3, \ldots, 0. \end{aligned}$$

With some auxiliary variables, Sun *et al.* modeled Modular Addition at Aisacrypt'17 in [11] as follows.

Rule 5 (`ModularAddition` for $\mathbb{K}$ [11]). *Let $(a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{n-1}, d_0, d_1, \ldots, d_{n-1})$ be a division trail of n-bit `Modular Addition` operation, to describe the division property propagation, the `Copy`, `AND` and `XOR` models should be applied in a specific order.*

Rule 6 (`Key − XOR`). *Assuming F is a component of `Key-XOR`, $(\mathbb{K}, \mathbb{L})$ and $(\mathbb{K}', \mathbb{L}')$ are the input and output division property, respectively. According to [15], the propagation is as follows,*

$$\mathbb{L}' \leftarrow \boldsymbol{l}, \ \textit{for } \boldsymbol{l} \in \mathbb{L},$$
$$\mathbb{K}' \leftarrow \boldsymbol{k}, \ \textit{for } \boldsymbol{k} \in \mathbb{K},$$
$$\mathbb{K}' \leftarrow (l_1, l_2, \ldots l_i \vee 1, \ldots, l_m), \textit{ for } \boldsymbol{l} \in \mathbb{L} \textit{ satisfying } l_i = 0, 1 \leqslant i \leqslant m.$$

2.2 Automatic Search for Bit-Based Division Property

As pointed in [15], the time and memory complexities for bit-based division property are upper-bounded by 2^n, where n denotes the block length. Therefore, the bit-based division property was just applied to SIMON32 and SIMECK32 in [15].

Recently, the techniques of automatic search for distinguishers have developed a lot. Automatic search can trace the transitions of sets in an efficient way. The propagation of vectors can be modeled by a serial of constrained optimization or decision statements. The technique has been used to find better differential and linear characteristics. Especially, it is very efficient to search for the division property.

Xiang *et al.* transformed the problem of finding two-subset division property into an MILP problem for the first time [17]. With the help of MILP solver Gurobi, they can find division property for ciphers with large block sizes, e.g., SIMON128 or PRESENT. To search for two-subset bit-based division property, they introduced the definition of two-subset division trail.

Definition 3 (Two-Subset Division Trail [17]). *Let us consider the propagation of the division property $\{\boldsymbol{k}\} \overset{def}{=} \mathbb{K}_0 \rightarrow \mathbb{K}_1 \rightarrow \ldots \rightarrow \mathbb{K}_r$. Moreover, for any vector $\boldsymbol{k}^*_{i+1} \in \mathbb{K}_{i+1}$, there must exit a vector $\boldsymbol{k}^*_i \in \mathbb{K}_i$ such that $\boldsymbol{k}^*_i$ can propagate to $\boldsymbol{k}^*_{i+1}$ by the propagation rules of the division property. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \ldots \times \mathbb{K}_r$ if $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$ for all $i \in \{0, 1, \ldots, r-1\}$, we call $(\boldsymbol{k}_0 \rightarrow \boldsymbol{k}_1 \rightarrow \ldots \rightarrow \boldsymbol{k}_r)$ an r-round division trail.*

2.2.1 Models of Propagation with SMT/SAT

Since we will use STP solver to implement our model, we introduce the SMT/SAT models for $\mathbb{K}$ describing the basic components `Copy`, `AND`, `XOR` and complex components `Modular Addition` according to Rule 5.

Model 1 (Bit-Based `Copy` for $\mathbb{K}$ [11]). *Denote $(a) \xrightarrow{Copy} (b_0, b_1)$ a division trail of `Copy` operation, the following logical equations are sufficient to depict the propagation of bit-based division trail,*

$$\begin{cases} \bar{b}_0 \vee \bar{b}_1 = 1 \\ a \vee b_0 \vee \bar{b}_1 = 1 \\ a \vee \bar{b}_0 \vee b_1 = 1 \\ \bar{a} \vee b_0 \vee b_1 = 1 \end{cases}.$$

Model 2 (Bit-Based `XOR` for $\mathbb{K}$ [11]). *Denote $(a_0, a_1) \xrightarrow{XOR} (b)$ a division trail of XOR function, the following logical equations are sufficient to evaluate the bit-based division trail through XOR operation,*

$$\begin{cases} \bar{a}_0 \vee \bar{a}_1 = 1 \\ a_0 \vee a_1 \vee \bar{b} = 1 \\ a_0 \vee \bar{a}_1 \vee b = 1 \\ \bar{a}_0 \vee a_1 \vee b = 1 \end{cases}.$$

Model 3 (Bit-Based `AND` for $\mathbb{K}$ [11]). *Denote $(a_0, a_1) \xrightarrow{AND} (b)$ a division trail of AND function, the following logical equations are sufficient to evaluate the bit-based division trail through AND operation,*

$$\begin{cases} \bar{a}_1 \vee b = 1 \\ a_0 \vee a_1 \vee \bar{b} = 1 \\ \bar{a}_0 \vee b = 1 \end{cases}.$$

Model 4 (Bit-Based `ModularAddition` for $\mathbb{K}$ [11]). *According to Rule 5, we can use the models of basic operations Copy, AND and XOR and some auxiliary variables to implement the Modular Addition.*

2.2.2 Initial and Stopping Rules of Two-Subset Division Property

An MILP or SMT/SAT model to search for two-subset bit-based division property needs to set proper initial and stopping rules, i.e., assign values to the initial and output variables in the division trail.

Assume that $(a_0^0, a_1^0, \ldots, a_{n-1}^0) \rightarrow \ldots \rightarrow (a_0^r, a_1^r, \ldots, a_{n-1}^r)$ is an r-round division trail for an n-bit length cipher. Let $\mathcal{D}_{\boldsymbol{k}}^{1^n}$ denote the initial division property with $\boldsymbol{k} = (k_0, k_1, \ldots, k_{n-1})$, and then we append the following constraints to the search model,

$$a_i^0 = k_i, \;\; i = 0, 1, 2, \cdots, n-1.$$

To check whether the i_0-th $(0 \leqslant i_0 \leqslant n-1)$ output bit is balanced or not, we just add constraints on a_i^r $(i = 0, 1, \ldots, n-1)$ that

$$a_i^r = \begin{cases} 1, & if \; i = i_0, \\ 0, & else. \end{cases}$$

If there is a division trail, the i_0-th output bit is decided as unknown; otherwise, the i_0-th output bit is balanced.

3 Search for Variant Three-Subset Division Property

3.1 Variant of Three-Subset Division Property

Firstly, we introduce a compromising propagation rule of `XOR` for $\mathbb{L}$ for VTDP as follows,

Rule 7 (Variant `XOR`). *Let F be a function compressed by an `XOR`, where the input $(x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output is calculated as $(x_1 \oplus x_2, x_3, \ldots, x_m)$. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output multiset, respectively. Assuming that $\mathbb{X}$ has $\mathcal{D}^{1^m}_{\mathbb{K},\mathbb{L}}$, $\mathbb{Y}$ has $\mathcal{D}^{1^{m-1}}_{\mathbb{K}',\mathbb{L}'}$, where $\mathbb{K}'$ is computed from $\boldsymbol{k} \in \mathbb{K}$ s.t. $(k_1, k_2) = (0,0), (1,0),$ or $(0,1)$ as*

$$\mathbb{K}' \leftarrow (k_1 + k_2, k_3, k_4, \ldots, k_m).$$

Moreover, $\mathbb{L}'$ is computed from $\boldsymbol{l} \in \mathbb{L}$ s.t. $(l_1, l_2) = (0,0), (1,0),$ or $(0,1)$ as

$$\mathbb{L}' \leftarrow (l_1 + l_2, l_3, l_4, \ldots, l_m),$$

In VTDP, we do not remove the duplicated vectors which appear even number of times in $\mathbb{L}$, and there are no other differences between VTDP and OTDP.

In VTDP, some duplicated vectors which appear even times will further generate some unexpected vectors in $\mathbb{L}$ and $\mathbb{K}$ by `Key-XOR`. As a result, there are many unexpected division trails in $\mathbb{K}$ and $\mathbb{L}$. Note that these extra trails will not change the original division trails inherited from OTDP if we do not remove the redundant vectors. Let $\mathbb{K}_V$ and $\mathbb{L}_V$ be the set containing all the division trails from all the duplicated vectors which appears even times and $\mathbb{K}_O$ and $\mathbb{L}_O$ be the set containing all the division trails which are from the OTDP. The following proposition describes the relationships between VTDP and OTDP.

Proposition 1. *Regarding one fixed bit of ciphertext, the VTDP will determine the parity of this bit by checking the vectors in $\mathbb{K}$ and $\mathbb{L}$ after r-round encryption. Compared with the results from OTDP, those from VTDP satisfy the following three properties.*

1. *If VTDP indicates that the parity of the bit is not unknown (even or odd), the parity of this bit is not unknown, too, according to OTDP.*
2. *If VTDP indicates that the parity of the bit is even, the parity of this bit is really even.*
3. *If VTDP indicates that the parity of the bit is odd, the parity of this bit will be constant.*

The proof is provided in the full version of this paper. According to Proposition 1, we can illustrate the relationship between VTDP and OTDP by Fig. 1. In Fig. 1, the colors represent the results based on OTDP while the line patterns stand for those based on VTDP. If one bit is determined as an odd-parity bit, we can know the bit is definitely not unknown. Therefore, we can still obtain some useful information from these results. In practice, we can encrypt all possible plaintexts by traversing all active plaintext bits under a random key, and Xor all the corresponding considered ciphtext bits to determine the parity of the considered ciphertext bits. This parity result holds for any key, which can be applied to attack the target cipher with any key. In other words, with our searching result, the test for only one key can achieve the available integral distinguisher for any key. Thus, our searching result is significant for attack. It is reasonable to encrypt the plaintexts because we need all the details of the cipher structure except the key-schedule to construct the model to search for VTDP. Note that the requirement also lies in the algorithm to search for OTDP [15].

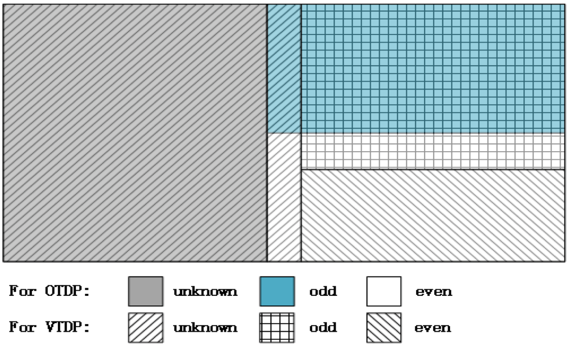

Fig. 1. Relationship between VTDP and OTDP.

3.2 Variant Three-Subset Division Trail

To model the automatic search for VTDP, we introduce the variant three-subset division trail.

Definition 4 (Variant Three-Subset Division Trial). *Let us consider the propagation of the division property* $\{(\boldsymbol{k},\boldsymbol{l})\} \overset{def}{=} \mathbb{K}_0 \times \mathbb{L}_0 \rightarrow \mathbb{K}_1 \times \mathbb{L}_1 \rightarrow \cdots \rightarrow \mathbb{K}_r \times \mathbb{L}_r$*. Moreover, for any vector tuple* $(\boldsymbol{k}^*_{i+1}, \boldsymbol{l}^*_{i+1})$*,* $\boldsymbol{k}^*_{i+1} \in \mathbb{K}_{i+1}$ *and* $\boldsymbol{l}^*_{i+1} \in \mathbb{L}_{i+1}$*, there must exit a vector tuple* $(\boldsymbol{k}^*_i, \boldsymbol{l}^*_i)$*,* $\boldsymbol{k}^*_i \in \mathbb{K}_i$ *and* $\boldsymbol{l}^*_i \in \mathbb{L}_i$*, such that* $(\boldsymbol{k}^*_i, \boldsymbol{l}^*_i)$ *can propagate to* $(\boldsymbol{k}^*_{i+1}, \boldsymbol{l}^*_{i+1})$ *by the propagation rules of the division property for* $i = 0, 1, \ldots, r-1$*. Furthermore, for* $((\boldsymbol{k}_0, \boldsymbol{l}_0), (\boldsymbol{k}_1, \boldsymbol{l}_1), \ldots, (\boldsymbol{k}_r, \boldsymbol{l}_r)) \in \mathbb{K}_0 \times \mathbb{L}_0 \times \mathbb{K}_1 \times \mathbb{L}_1 \times \cdots \times \mathbb{K}_r \times \mathbb{L}_r$*, if* $(\boldsymbol{k}_i, \boldsymbol{l}_i)$ *can propagate to* $(\boldsymbol{k}_{i+1}, \boldsymbol{l}_{k+1})$ *for all* $i \in \{0, 1, \ldots, r-1\}$*, we call* $(\boldsymbol{k}_0, \boldsymbol{l}_0) \rightarrow (\boldsymbol{k}_1, \boldsymbol{l}_1) \rightarrow \cdots \rightarrow (\boldsymbol{k}_r, \boldsymbol{l}_r)$ *an r-round variant three-subset division trail.*

Similar to methods in [17], we decide the parity of one output bit by checking whether certain division trails exist. Therefore, we need to transform the propagation rules of each component into constraints and solve the problem by an MILP or SMT/SAT tool. We divide all components into key-independent and key-dependent components according to whether there are secret keys involved.

For key-independent components, we construct the models of `Copy`, `AND`, `XOR` and `Modular Addition` operations for $\mathbb{L}$ according to Rule 1, 2, 5 and 7. Since there is no rule of `S-box` for $\mathbb{L}$, we give the rule and then model the `S-box` in Sect. 3.3 for the first time.

For key-dependent components, we concentrate only on the `Key-XOR` operation. We introduce a new technique that we can use the logical OR operation of STP solver to model the dependencies between $\mathbb{K}$ and $\mathbb{L}$ when a `Key-XOR` component is applied.

Note 1. Since redundant vectors do not affect the result of $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \pi_{\boldsymbol{u}}(\boldsymbol{x})$, our model will not remove them.

3.3 Models of VTDP for Key-Independent Components

Assuming that f is a key-independent component of a cipher, $(\mathbb{K}, \mathbb{L})$ and $(\mathbb{K}', \mathbb{L}')$ are the input and output division property of f, respectively. In our automatic search model, we allocate variables to represent the vectors in $\mathbb{K}$, $\mathbb{L}$, $\mathbb{K}'$ and $\mathbb{L}'$ at the bit level at first and then the constraints are added on these variables according to the propagation rule of f. Note that the propagations of $\mathbb{K} \xrightarrow{f} \mathbb{K}'$ and $\mathbb{L} \xrightarrow{f} \mathbb{L}'$ are conducted separately according to their own rules.

In this paper, we use STP solver to implement our model. STP is a simple but efficient solver for the theory of quantifier-free bit vectors. It is first introduced to find optimal differential characteristic by Mouha and Preneel [8]. At Asiacrypt'17, Sun et al. took it to search for division property [11]. We can describe the propagation rules in CNF formulas using the method proposed in [11]. The automatic search models for $\mathbb{K}$ has been listed in Sect. 2.1. We construct models for the basic operations `Copy`, `AND`, `XOR` and `Modular Addition` for $\mathbb{L}$ in a similar way.

For `Copy` operation, let a, b_0 and b_1 be three binary variables and $(a) \xrightarrow{\texttt{Copy}} (b_0, b_1)$ be the division trail. There are four possible division trails according to Rule 1, which are $(0) \rightarrow (0,0)$, $(1) \rightarrow (0,1)$, $(1) \rightarrow (1,0)$ and $(1) \rightarrow (1,1)$. To make (a, b_0, b_1) follow these four division trails only we put constraints on a, b_0 and b_1 as follows.

Model 5 (Bit-Based `Copy` for $\mathbb{L}$). *Denote (a, b_0, b_1) a division trail of `Copy` function, the following logical equations are sufficient to evaluate the bit-based division trail through `Copy` operation,*

$$\begin{cases} a \vee b_0 \vee \bar{b}_1 = 1 \\ \bar{a} \vee b_0 \vee b_1 = 1 \\ a \vee \bar{b}_0 = 1 \end{cases}.$$

For `AND` operation, let a_0, a_1 and b be three binary variables and $(a_0, a_1) \xrightarrow{\texttt{AND}} (b)$ be the division trail. There are two possible division trails according to Rule 2, which are $(0,0) \rightarrow (0)$ and $(1,1) \rightarrow (1)$, To make (a_0, a_1, b) follow these two division trails only we add constrains on a_0, a_1 and b as follows.

Model 6 (Bit-Based `AND` **for $\mathbb{L}$).** *Denote (a_0, a_1, b) a division trail of* ***AND*** *function, the following logical equations are sufficient to evaluate the bit-based division trail trough* ***AND*** *operation,*

$$\begin{cases} a_0 = b \\ a_1 = b \end{cases}.$$

For `XOR` operation, we follow the Rule 7 rather than Rule 3, let a_0, a_1 and b be three binary variables and $(a_0, a_1) \xrightarrow{\texttt{XOR}} (b)$ be the division trail. There are three possible division trails according to Rule 7, which are $(0,0) \to (0)$, $(0,1) \to (1)$ and $(1,0) \to (1)$. To make (a_0, a_1, b) follow these three division trails only we append constraints on a_0, a_1 and b as follows.

Model 7 (Bit-Based Variant `XOR` **for $\mathbb{L}$).** *Denote (a_0, a_1, b) a division trail of* ***XOR*** *function, the following logical equations are sufficient to evaluate the bit-based division trail trough* ***XOR*** *operation,*

$$\begin{cases} a_0 \vee a_1 \vee \bar{b} = 1 \\ \bar{a}_1 \vee b = 1 \\ \bar{a}_0 \vee a_1 \vee b = 1 \\ \bar{a}_0 \vee \bar{a}_1 \vee \bar{b} = 1 \end{cases}.$$

Model 8 (Bit-Based `ModularAddition` **for $\mathbb{L}$).** *The model of* ***Modular Addition*** *for $\mathbb{L}$ is totally same with that for $\mathbb{K}$ except that we use basic models of* ***Copy****,* ***AND*** *and variant* ***XOR*** *for $\mathbb{L}$ rather than $\mathbb{K}$.*

Modeling `S-box` **for $\mathbb{L}$**

The rule to calculate all the division trails of an `S-box` for $\mathbb{K}$ was presented in [3,17]. Here we introduce the rules to find all the division trails for $\mathbb{L}$.

Let $F : (\mathbb{F}_2)^m \to (\mathbb{F}_2)^n$ be a function of substitution composed of $(f_1, f_2, \ldots, f_n)$, where the input $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ takes values of $(\mathbb{F}_2)^m$, and the output $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ is calculated as

$$\begin{aligned} y_1 &= f_1(x_1, x_2, \ldots, x_m), \\ y_2 &= f_2(x_1, x_2, \ldots, x_m), \\ &\vdots \\ y_n &= f_n(x_1, x_2, \ldots, x_m). \end{aligned}$$

Similar to Rule 4, for each input vector $\boldsymbol{u} \in \mathbb{L}$, we consider each output vector $\boldsymbol{v} \in \mathbb{F}_2^n$ seperately to derive all the valid division trails. According to Definition 2, for each vector $\boldsymbol{v} \in \mathbb{F}_2^n$, $(\boldsymbol{u}, \boldsymbol{v})$ is a valid division trail if the polynomial $\pi_{\boldsymbol{v}}(\boldsymbol{y})$ contains the monomial $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ but does not contain the monomial $\pi_{\boldsymbol{u}'}(\boldsymbol{x})$ for any $\boldsymbol{u}'$ satisfying $\boldsymbol{u}' \succ \boldsymbol{u}$.

Algorithm 1. Calculating Division Trails of `S-box` for $\mathbb{L}$

```
Input: a vector $\boldsymbol{u}$ representing the input division property
Output: A set $\mathbb{L}$ of vectors representing the output division property
$\bar{\mathbb{S}} = \{\bar{\boldsymbol{u}} | \bar{\boldsymbol{u}} \succ \boldsymbol{u}\}$;
$F(X) = \{\pi_{\bar{u}}(\boldsymbol{x}) | \bar{\boldsymbol{u}} \in \bar{\mathbb{S}}\}$;
$Allocate \mathbb{L} = \varnothing$;
for each $\boldsymbol{v} \in \mathbb{F}_2^n$ do
    if $\pi_{\boldsymbol{v}}(\boldsymbol{y})$ does not contain any monomial in $F(X)$ and $\pi_{\boldsymbol{v}}(\boldsymbol{y})$ contains $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ then
        $\mathbb{L} \leftarrow \boldsymbol{v}$;
return $\mathbb{L}$;
```

We give Algorithm 1 to calculate all the valid division trails of `S-box` for $\mathbb{L}$.

To implement the model for `S-box`, firstly we use Algorithm 1 to compute all the division trails. Then we need to describe these trails in STP solver. We define an array variable to store all the trails and then use this array to add constraints on the variables representing the input and output division property[1].

3.4 Model of VTDP for Key-XOR

For `Key-XOR` operation f_k, the input and output division properties are $\{\mathbb{K}, \mathbb{L}\}$ and $\{\mathbb{K}', \mathbb{L}'\}$, respectively. In our model, we use four n-bit variables $\mathcal{K}, \mathcal{L}, \mathcal{K}'$ and $\mathcal{L}'$ to denote them, where n is the block size. Because the dependencies between $\mathbb{K}$ and $\mathbb{L}$ work on the block rather than a single bit, we use n-bit variables rather than binary variables.

According to Rule 6, f_k does not affect the propagation from $\mathbb{L}$ to $\mathbb{L}'$. Therefore, the constraint on $\mathcal{L}$ and $\mathcal{L}'$ is $\mathcal{L}' = \mathcal{L}$.

In many ciphers, round key is only XORed with a part of block. Without loss of generality, we assume that the round key is XORed with the left s $(1 \leqslant s \leqslant n)$ bits. This operation can be divided into two steps.

1. Allocate n-bit variables $\mathcal{V}_j$ $(j \in \{0, 1, 2, \ldots, s-1\})$. Check each bit of $\mathcal{L}$, i.e., $\mathcal{L}[0], \mathcal{L}[1], \ldots, \mathcal{L}[s-1]$, and assign $\mathcal{V}_j$ as follows,

$$\mathcal{V}_j = \begin{cases} \mathcal{L} \vee \boldsymbol{e}_j, & \text{if } \mathcal{L}[j] = 0, \\ \mathbf{1}, & otherwise, \end{cases}$$

   where $\boldsymbol{e}_j$ is an n-bit unit vector whose bit j is one and $\mathbf{1}$ is the vector with all components one. If $\mathcal{L}[j] \neq 0$, we set $\mathcal{V}_j$ as $\mathbf{1}$ because we use the STP statement IF-THEN-ELSE to implement it, which follows a strict grammar. Note that $\mathbf{1}$ has no effect on the search results.
2. Let $\{\mathcal{K}'\} = \{\mathcal{K}\} \cup \{\mathcal{V}_0\} \cup \{\mathcal{V}_1\} \cup \cdots \cup \{\mathcal{V}_{s-1}\}$.

[1] We can implement the model of `S-box` using the exclusion method as those of `Copy`, `AND` and `XOR`, also.

In STP solver, we can implement the first step with an IF-THEN-ELSE branch statement as follows,

$$\text{ASSERT } \mathcal{L}^j = \text{ IF } \mathcal{L}[j] = 0 \text{ THEN } \mathcal{L} \ \vee \ \boldsymbol{e}_j \text{ ELSE } \mathbf{1} \text{ ENDIF;}$$

For the second step, we use the following statement with the logical OR operation in STP to implement,

$$\text{ASSERT } \mathcal{K}' = \mathcal{K} \text{ OR } \mathcal{K}' = \mathcal{V}_0 \text{ OR } \mathcal{K}' = \mathcal{V}_1 \text{ OR } \dots \text{ OR } \mathcal{K}' = \mathcal{V}_{s-1};$$

Algorithm 2 concludes the model of the `Key-XOR` operation.

Algorithm 2. Generating Constraints of Propagation Rule of `Key-XOR`

```
Input: $n$-bit variables $\mathcal{K}, \mathcal{K}', \mathcal{L}, \mathcal{L}'$.
Output: A set $\mathbb{C}$ with constraints on $\mathcal{K}, \mathcal{K}', \mathcal{L}, \mathcal{L}'$.
Allocate $\mathbb{C}$ as $\varnothing$;
$\mathbb{C} \leftarrow \mathcal{L}' = \mathcal{L}$;
Allocate $n$-bit variables $\mathcal{V}_j$ $(j = 0, 1, \dots, s-1)$;
for $j = 0; j < s; j = j + 1$ do
    if $\mathcal{L}[j] == 0$ then
        $\mathbb{C} \leftarrow \mathcal{V}_j = \mathcal{L} \vee \boldsymbol{e}_j$;
    else
        $\mathbb{C} \leftarrow \mathcal{V}_j = \mathbf{1}$;
$\mathbb{C} \leftarrow \mathcal{K}' = \mathcal{K}$ OR $\mathcal{K}' = \mathcal{V}_0$ OR $\mathcal{K}' = \mathcal{V}_1$ OR $\cdots$ OR $\mathcal{K}' = \mathcal{V}_{s-1}$;
return $\mathbb{C}$;
```

Note 2. We just know that the STP solver supports the logical OR operation, so our model relies on it. However, any tool that can implement the two steps is suitable to our algorithm also.

3.5 Initial and Stopping Rules for VTDP

Initial Rule

In [15], to search for three-subset division property, Todo and Morii set the initial division property as $(\boldsymbol{k} = \mathbf{1}, \boldsymbol{l})$, where the active bits of $\boldsymbol{l}$ are set as one or zero for constant bits. It is the same for VTDP. For example, if we find integral characteristic for SIMON32 using 2^{31} chosen-plaintexts with first bit constant, the initial division property is then set as $(\boldsymbol{k} = \mathbf{1}, \boldsymbol{l} = \texttt{7fffffff})$. Let $((\mathcal{K}_0^0, \mathcal{K}_1^0, \dots, \mathcal{K}_{n-1}^0), (\mathcal{L}_0^0, \mathcal{L}_1^0, \dots, \mathcal{L}_{n-1}^0))$ denote the initial division property, where n is the block size. The constraints on $\mathcal{K}_i^0$ and $\mathcal{L}_i^0$ are

$$\mathcal{K}_i^0 = 1, \text{for } i = 0, 1, 2, \dots, n-1.$$

$$\mathcal{L}_i^0 = \begin{cases} 1, & \text{if the } i\text{-th bit is active,} \\ 0, & \text{otherwise.} \end{cases}$$

Stopping Rule

Our automatic search model only focuses on the parity of one output bit. Without loss of generality, we consider the i_0-th output bit. According to Definition 2, the first step is to examine whether there is a unit vector $\boldsymbol{e}_{i_0} \in \mathbb{K}$ for the r-th round, so we only need to set the constraints on $(\mathcal{K}_0^r, \mathcal{K}_1^r, \ldots, \mathcal{K}_{n-1}^r)$ as follows,

$$\mathcal{K}_i^r = \begin{cases} 1, & \text{if } i = i_0, \\ 0, & \text{otherwise.} \end{cases}$$

If the constraint problem has solutions, the i_0-th bit is unknown, and our algorithm stops. Otherwise, we need to remove the constraints on $\mathcal{K}_i^r$ $(0 \leqslant i \leqslant n-1)$ and add the following constraints on $(\mathcal{L}_0^r, \mathcal{L}_1^r, \ldots, \mathcal{L}_{n-1}^r)$,

$$\mathcal{L}_i^r = \begin{cases} 1, & \text{if } i = i_0, \\ 0, & \text{otherwise.} \end{cases}$$

If there is still no solution, the i_0-th bit is balanced, otherwise the parity of the i_0-th bit is even or odd.

3.6 Connection Between Key-Independent and Key-XOR Components

Note that we use bit-level variables to model the key-independent components in Sect. 3.3, but the implementations for key-XOR are based on n-bit variables. Therefore, in order to connect bit variables and n-bit variables, the concatenation operation "@" in STP is used to link them. Let the bit variables $(\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_{n-1})$ denote the output division property for $\mathbb{L}$ of a key-independent component, whose following operation is Key-XOR with input division property $\mathcal{L}' \in \mathbb{F}_2^n$. The link constraint on them is

$$\text{ASSERT } \mathcal{L}' = \mathcal{L}_0 @ \mathcal{L}_1 @ \ldots @ \mathcal{L}_{n-1};$$

Conversely, if $\mathcal{L}'$ is the output of Key-XOR while $(\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_{n-1})$ are the input of next key-independent component, we use the statement above, too.

4 Applications

In this section, we apply our model to SIMON, SIMECK, SIMON(102), SPECK, PRESENT and KATAN/KTANTAN. All our experiments are implemented on a server with 48 Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30 GHz and 96 GB memory. And some of the programs run in a parallel way as long as the memory is enough. In our illustrations, the character '?' represents unknown, '*' represents even or odd and '0' stands for even. All the programs for these algorithms are public in website https://github.com/VTDP/submission_for_ctrsa/.

4.1 VTDP of SIMON-Like Ciphers

SIMON [1] is a family of lightweight block ciphers published by the U.S. National Security Agency (NSA) in 2013. SIMON adopts Feistel structure and it has a very compact round function which only involves bit-wise `And`, `XOR` and `Circular Shift`. According to the block size, SIMON family is composed of SIMON32, SIMON48, SIMON64, SIMON96, SIMON128.

For SIMON32, we identify a 15-round integral characteristic which is as follows,

$$\texttt{(7fff,ffff)} \xrightarrow{14r} \texttt{(????,????,????,????,?*??,????,*???,???*)}.$$

Then we can encrypt the corresponding 2^{31} chosen-plaintexts and determine the three bits represented by '*' are all even, which is the same result as that in [15]. However, our automatic algorithm takes about 27 s which is much more efficient than that in [15]. Unfortunately, the results for SIMON48/64/96/128 with VTDP have no improvements compared with the previous distinguishers.

SIMECK is a family of lightweight block cipher proposed at CHES'15 [18]. The round function of SIMECK is very like SIMON except the rotation constants. We apply our automatic search algorithm to 15-, 18- and 21-round SIMECK32/48/64, respectively. All the integral characteristics from our algorithm are the same as those found by Xiang *et al.*

In [7], another variant of SIMON family named SIMON(102) is proposed with rotation constants (1, 0, 2).

For 20-round SIMON32(102), we find the following improved integral distinguisher

$$\texttt{(7fff,ffff)} \xrightarrow{19r} \texttt{(????,????,????,????,0*??,????,????,???*)},$$

which has two additional odd or even parity bits compared with the previous best results,

$$\texttt{(7fff,ffff)} \xrightarrow{19r} \texttt{(????,????,????,????,0???,????,????,????)}.$$

Similarly, for 28-round SIMON48(102) and 35-round SIMON64(102), a new distinguisher with two extra odd or even parity bits have been found, respectively.

$$\texttt{(7fff,ffff,ffff)} \xrightarrow{27r}$$
$$\texttt{(????,????,????,????,????,????,0*??,????,????,????,???*)}.$$

$$\texttt{(7fff,ffff,ffff,ffff)} \xrightarrow{35r}$$
$$\texttt{(????,????,????,????,????,????,????,????, 0*??,????,????,????,????,????,????,???*)}.$$

4.2 VTDP of ARX Cipher SPECK

SPECK [1] is a family of lightweight block ciphers published by NSA, too. Different from SIMON, SPECK takes the `Modular Addition` as its nonlinear operation. According to the block size, SPECK family has 5 members, SPECK32, SPECK48, SPECK64, SPECK96 and SPECK128. shift by i bits and $\boxplus$ represents the Modular Addition operation.

For SPECK32, there only exists one two-subset bit-based integral distinguisher for 6 rounds with 2^{31} chosen-plaintexts as follows,

$$(\texttt{ffff},\texttt{ffdf}) \xrightarrow{6r} (\texttt{????},\texttt{????},\texttt{????},\texttt{???0},\texttt{????},\texttt{????},\texttt{????},\texttt{????}).$$

However, based on VTDP, we can find one more distinguisher besides the above one,

$$(\texttt{ffff},\texttt{ffbf}) \xrightarrow{6r} (\texttt{????},\texttt{????},\texttt{????},\texttt{???*},\texttt{????},\texttt{????},\texttt{????},\texttt{????}).$$

4.3 VTDP of S-Box Based Cipher PRESENT

PRESENT [2] is an SP-network block cipher, of which the linear layers are bit permutations.

In [17], Xiang *et al.* found a 9-round integral distinguisher with 2^{60} chosen-plaintexts under the two-subset division property framework. Our algorithm achieves the same result. Furthermore, If we use more data complexity such as 2^{63} chosen-plaintexts with the leftmost 63 bits active, we find a new distinguisher with 28 balanced bits which is listed as follows,

$$(\texttt{ffff},\texttt{ffff},\texttt{ffff},\texttt{fffe}) \xrightarrow{9r}$$
$$(\texttt{???0},\texttt{???0},\texttt{???0},\texttt{0000},\texttt{???0},\texttt{???0},\texttt{???0},\texttt{0000},\texttt{???0},\texttt{???0},\texttt{???0},\texttt{0000},\texttt{???0},\texttt{???0},\texttt{???0},\texttt{0000}).$$

Note that this distinguisher can be found by Xiang *et al.*'s model.

4.4 VTDP of KATAN/KTANTAN Family

KATAN and KTANTAN [4] are two families of hardware oriented block ciphers and have three variants of 32-bit, 48-bit, 64-bit block. KATAN/KTANTAN takes a very simple structure composed of two LFSR's.

KATAN/KTANTAN32, 48, 64 conduct the round function once, twice and three times in one round with the same round key, respectively. The only difference between KATAN and KTANTAN is the key schedule.

Compared with the previous results [9], we obtained the longer integral distinguishers for KATAN/KTANTAN32 and 48 with our automatic algorithm for VTDP. Moreover, our identified integral characteristic for $72\frac{1}{3}$-round KATAN/KTANTAN64 has two more balanced bits.

For KATAN/KTANTAN32, Sun *et al.* found the following 99-round integral characteristic with the two-subset division property [9],

$$(\texttt{fffb},\texttt{ffff}) \xrightarrow{99r} (\texttt{????},\texttt{????},\texttt{????},\texttt{????},\texttt{????},\texttt{????},\texttt{????},\texttt{???0}).$$

However, our new distinguishers based on VTDP are listed as follows,

$$(\texttt{fffb},\texttt{ffff}) \xrightarrow{100r} (\texttt{????,????,????,????,????,????,????,??*0}),$$
$$(\texttt{fffb},\texttt{ffff}) \xrightarrow{101r} (\texttt{????,????,????,????,????,????,????,???*}).$$

For 64- and $72\frac{1}{2}$-round KATAN/KTANTAN48 and KATAN/KTANTAN64, respectively, the search program requires too much time to get VTDP. Therefore, we introduce a compromising strategy to simplify some propagation of vectors. For two-subset division property, we only trace $\mathbb{K}$, but for three-subset division property, $\mathbb{K}$ and $\mathbb{L}$ are considered. In general, the program of two-subset division property will take less time than that of the three-subset one. In our program, we can trace $\mathbb{K}$ and $\mathbb{L}$ for the first N rounds only; and append $\boldsymbol{u}$ to $\mathbb{K}$ for all $\boldsymbol{u} \in \mathbb{L}$ at the N-th round; then trace the modified $\mathbb{K}$ merely. Since after N rounds, the program becomes a two-subset division property, the stopping rules should follow that of the two-subset division property.

With the compromising strategy, we still find better integral distinguishers for KATAN/KTANTAN48 and 64 than those in [9].

For 64-round KATAN/KTANTAN48, the distinguisher we found is presented as follows ($N = 100$),

$$(\texttt{ffff},\texttt{efff},\texttt{ffff}) \xrightarrow{64r}$$
$$(\texttt{????,????,????,????,????,????,????,????,????,????,???0}),$$

which covers half more round than that in [9]. For KATAN/KTANTAN64, we find the same length of integral distinguisher with the previous best one [9] but ours has one more balanced bit as follows ($N = 50$),

$$(\texttt{ffff},\texttt{ffbf},\texttt{ffff},\texttt{ffff}) \xrightarrow{72.3r}$$
$$(\texttt{????,????,????,????,????,????,????,????,????,????,????,????,????,????,??00}).$$

5 Conclusions

In this paper, we proposed an automatic search model for a variant of three-subset division property and it can be applied to ciphers with large block sizes. Furthermore, we give the rules of `S-box` and `Modular Addition` for $\mathbb{L}$, which extend the usage of three-subset division property. With this model, the better integral distinguishers have been found compared with the previous results.

Acknowledgement. The authors would like to thank Yosuke Todo for his important comments and suggestions to this paper. This work is supported by National Cryptography Development Fund (MMJJ20170102), National Natural Science Foundation of China (Grant No. 61572293) and Major Scientific and Technological Innovation Projects of Shandong Province, China (2017CXGC0704).

References

1. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: PADAC 2015, pp. 175:1–175:6 (2015)
2. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
3. Boura, C., Canteaut, A.: Another view of the division property. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 654–682. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_24
4. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_20
5. Funabiki, Y., Todo, Y., Isobe, T., Morii, M.: Improved integral attack on HIGHT. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 363–383. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_19
6. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45661-9_9
7. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 161–185. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_8
8. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for ARX: application to salsa20. Cryptology ePrint Archive, Report 2013/328 (2013)
9. Sun, L., Wang, W., Liu, R., Wang, M.: MILP-aided bit-based division property for ARX-based block cipher. IACR Cryptology ePrint Archive 2016:1101 (2016)
10. Sun, L., Wang, W., Wang, M.: MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. IACR Cryptology ePrint Archive 2016:811 (2016)
11. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5
12. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12
13. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 413–432. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_20
14. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_9
15. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_18

16. Wang, Q., Grassi, L., Rechberger, C.: Zero-sum partitions of PHOTON permutations. IACR Cryptology ePrint Archive 2017:1211 (2017)
17. 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.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_24
18. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The Simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 307–329. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_16