Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers

Zejun Xiang[1,2], Wentao Zhang[1,2], Zhenzhen Bao[1,2], and Dongdai Lin[1,2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{xiangzejun, zhangwentao, baozhenzhen, ddlin}@iie.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. Division property is a generalized integral property proposed by Todo at EUROCRYPT 2015, and very recently, Todo *et al.* proposed bit-based division property and applied to SIMON32 at FSE 2016. However, this technique can only be applied to block ciphers with block size no larger than 32 due to its high time and memory complexity. In this paper, we extend Mixed Integer Linear Programming (MILP) method, which is used to search differential characteristics and linear trails of block ciphers, to search integral distinguishers of block ciphers based on division property with block size larger than 32.

Firstly, we study how to model division property propagations of three basic operations (copy, bitwise AND, XOR) and an Sbox operation by linear inequalities, based on which we are able to construct a linear inequality system which can accurately describe the division property propagations of a block cipher given an initial division property. Secondly, by choosing an appropriate objective function, we convert a search algorithm under Todo's framework into an MILP problem, and we use this MILP problem appropriately to search integral distinguishers. As an application of our technique, we have searched integral distinguishers for SIMON, SIMECK, PRESENT, RECTANGLE, LBlock and TWINE. Our results show that we can find 14-, 16-, 18-, 22- and 26-round integral distinguishers for SIMON32, 48, 64, 96 and 128 respectively. Moreover, for two SP-network lightweight block ciphers PRESENT and RECTANGLE, we found 9-round integral distinguishers for both ciphers which are two more rounds than the best integral distinguishers in the literature [22][29]. For LBlock and TWINE, our results are consistent with the best known ones with respect to the longest distinguishers.

Key words: MILP, division property, integral cryptanalysis, SIMON, SIMECK, PRESENT, RECTANGLE, LBlock, TWINE

1 Introduction

Programming problem is a mathematical optimization which aims to achieve the minimal or maximal value of an objective function under certain constraints, and

it has a wide range of applications from industry to academic community. Mixed Integer Linear Programming (MILP) is a kind of programming problem whose objective function and constraints are linear, and all or some of the variables involved in the problem are restricted to be integers. In recent years, MILP has found its applications in cryptographic community. Mouha *et al.* [11] and Wu *et al.* [21] applied MILP method to automatically count differential and linear active Sboxes for word-based block ciphers, which can be used to evaluate the resistance of block ciphers against differential and linear attacks. Later Sun *et al.* [13] extended this technique to count active Sboxes of SP-network block ciphers whose linear layer is a bit permutation.

Recently, this technique was improved [15] to search differential characteristics and linear trails with a minimal number of active Sboxes. They constructed the MILP model by a small number of linear inequalities chosen from the H-Representation of the convex hull of a set of points which are derived from the difference distribution (resp. linear approximation) table of Sbox. However, this method may result in invalid differential characteristics (resp. linear trails). Moreover, differential characteristic (resp. linear trail) with a minimal number of active Sboxes does not alway result in differential characteristic (resp. linear trail) with highest probability. To solve these problems, Sun *et al.* [14] encoded the probability of differentials (resp. linear approximations) of Sbox into the MILP model and they proved that it is always feasible to choose a set $\mathcal{L}$ of linear inequalities from the H-Representation of the convex hull of a set of points A, such that the feasible solutions of $\mathcal{L}$ are exactly the points in A. Thus, by adding $\mathcal{L}$ into the model and setting the probability as objective function, the MILP optimizer will always return (if the MILP problem can be solved in limited time) a valid differential characteristic (resp. linear trail) with highest probability.

Division property is a generalized integral property introduced by Todo [18] at EUROCRYPT 2015 to search integral distinguishers of block cipher structures which is the core part of integral cryptanalysis [4,7,8,10]. Todo studied propagation rules of division property through different block cipher operations and presented generalized algorithms to search integral distinguishers which only exploits the algebraic degree of nonlinear components of the block cipher. By using division property, Todo presented 10-, 12-, 12-, 14- and 14-round[3] integral distinguishers for SIMON32, 48, 64, 96 and 128 respectively. For PRESENT cipher a 6-round integral distinguisher was found. Later at CRYPTO 2015 Todo [17] proposed a full-round integral attack of MISTY1 based on a 6-round integral distinguisher. Sun *et al.* [12] revisited division property, and they studied the property of a set (multiset) satisfying certain division property. At CRYPTO 2016, Boura and Canteaut [6] proposed a new notion which they called parity set to study division property, based on which they found better integral distinguisher for PRESENT cipher.

[3] Since the round key is Xored into the state after the round function, we can easily extend one more round before the distinguisher by using the technique proposed in [20].

Very recently, Todo *et al.* [19] introduced bit-based division property at FSE 2016 which treats each bit independently in order to find better integral distinguishers. They applied this technique to SIMON32, and as a result a 14-round integral distinguisher for SIMON32 was found. However, as pointed out in [19], searching integral distinguisher by bit-based division property required much more time and memory. For a block cipher with block size n, the time and memory complexity is upper bounded by 2^n. Thus, bit-based division property can only apply to block ciphers with block size at most 32. For block ciphers with a much larger block size, searching integral distinguisher by bit-based division property under Todo *et al.*'s framework would be computationally infeasible. Thus, Xiang *et al.* [24] proposed a state partition to get a tradeoff between the time-memory complexity and the accuracy of the integral distinguisher, and they improved distinguishers of SIMON48 and SIMON64 by one round for both variants.

1.1 Our Contributions

In this paper, we present a novel technique to search integral distinguishers based on bit-based division property by using MILP method. First we propose a new notion that we call *division trail* to illustrate division property propagation. We show that each division property propagation can be represented by division trails, furthermore, we have proved that it is sufficient to check the last vectors of all division trails in order to estimate whether a useful distinguisher exists. Based on this observation we construct a linear inequality system for a given block cipher such that all feasible solutions of this linear inequality system are exactly all the division trails. Thus, the constructed linear inequality system is sufficient to describe the division property propagations. Then, we study the stopping rule in division property propagation. The stopping rule determines whether the resulting division property can be propagated further to find a longer integral distinguisher. It is observed that for a division property propagation, if the resulting vectors for the first time contain all the vectors of Hamming weight one after propagating $r+1$ rounds, the propagation procedure should terminate and an r-round distinguisher can be derived. Hence, we set the sum of the coordinates of the last vector of r-round division trail as objective function. By combining this objective function and the linear inequality system derived from the division trails, we construct an MILP problem and present an algorithm to estimate whether r-round distinguisher exists given some initial division property. To illustrate our new technique, we run experiments (all the MILP problems in our experiments are solved by the openly available software Gurobi [1]) on SIMON, SIMECK, PRESENT, RECTANGLE, TWINE, LBlock:

1. For SIMON [3] family block ciphers, we first model division property propagations through *Copy*, *And* and *Xor* operations by linear inequalities, since those operations are the basic operations in SIMON family. By using these inequalities we construct an MILP problem and serve it in our search algorithm. As a result we found 14-, 16-, 18-, 22- and 26-round integral distinguishers for SIMON32, 48, 64, 96 and 128 respectively. For SIMON48, 64, 96

and 128, our results are 2, 1, 1 and 1 more rounds than the previous results in [27]. SIMECK [25] is a family of lightweight block ciphers whose round function is very similar to SIMON except the rotation constants. We applied our search technique to SIMECK and we found 15-, 18- and 21-round distinguishers for SIMECK32, 48 and 64 respectively.

2. PRESENT [5] and RECTANGLE [28] are two SP-network lightweight block ciphers whose linear layers are bit permutations. Unlike SIMON, these two ciphers are Sbox-based block ciphers. In [17,18], Sbox is treated as a whole, that is for an n-bit Sbox the input value to the Sbox is viewed as a value of $\mathbb{F}_2^n$. In this paper we study bit-based division property propagation of Sbox, and we present an algorithm to compute division trails of Sbox. We observed that, considering bit-based division property could preserve more integral property along with division property propagation through Sbox. By converting division trails of Sbox layer into a set of linear inequalities we construct MILP models for PRESENT and RECTANGLE, as a result, we found 9-round distinguishers for both ciphers which are two more rounds than the best integral distinguishers in the literature.
3. TWINE [16] and LBlock [23] are two generalized Feistel structure block ciphers. By modeling *Sbox*, *Copy* and *Xor* with linear inequalities, we apply our technique to these two ciphers and we found 16-round distinguishers which are in accordance with the results in [26].

Our results are listed in Table 1. All the ciphers explored above except SIMON32 have a block size larger than 32, and searching integral distinguishers by bit-based division property under Todo's framework is computationally infeasible for those ciphers. Note that all our experiments are conducted on a desktop and the consuming time varies from seconds to minutes which is very efficient, the details are listed in Table 1. Moreover, by converting the search algorithm into MILP problems, we can find better integral distinguishers for SIMON48/64/96/128, SIMECK48/64, PRESENT and RECTANGLE.

The rest of the paper is organized as follows: In Section 2 we introduce some basic background which will be used later. Section 3 studies how to model some basic operations and components used in block cipher, and to construct a linear inequality system to accurately describe the division property propagations. Section 4 studies the stopping rule and a search algorithm will be presented in this section. Section 5 shows some applications of the technique, and we conclude in Section 6.

2 Preliminaries

2.1 Notations

Let $\mathbb{F}_2$ denote the finite field with only two elements and $\mathbb{F}_2^n$ denote the n-bit string over $\mathbb{F}_2$. let $\mathbb{Z}$ and $\mathbb{Z}^n$ denote the integer ring and the set of all vectors whose coordinates are integers respectively. For any $a \in \mathbb{F}_2^n$, let $a[i]$ denote the i-th bit of a, and the Hamming weight of a is calculated as $\sum_{i=0}^{n-1} a[i]$. For any

Table 1. Results on Some Block Ciphers.

Cipher	Block size	Round (Previous)	Round (Sect. 5)	Data	Balanced bits	time
SIMON32	32	15 [19]	14	31	16	4.1s
SIMON48	48	14 [27]	16	47	24	48.2s
SIMON64	64	17 [27]	18	63	22	6.7m
SIMON96	96	21 [27]	22	95	5	17.4m
SIMON128	128	25 [27]	26	127	3	58.4m
SIMECK32	32	15 [19]	15	31	7	6.5s
SIMECK48	48	12 [18]	18	47	5	56.6s
SIMECK64	64	12 [18]	21	63	5	3.0m
PRESENT	64	7 [22]	9	60	1	3.4m
RECTANGLE	64	7 [28]	9	60	16	4.1m
LBlock	64	16 [26]	16	63	32	4.9m
TWINE	64	16 [26]	16	63	32	2.6m

For SIMON and SIMECK family block ciphers, since the round key is Xored into the state after the round function, we can add one more round before the distinguishers using the technique in [20]. The results presented in the third and fourth columns have been added by one round.

$\boldsymbol{a} = (a_0, \cdots, a_{m-1}) \in \mathbb{F}_2^{n_0} \times \cdots \times \mathbb{F}_2^{n_{m-1}}$, the vectorial Hamming weight of $\boldsymbol{a}$ is defined as $W(\boldsymbol{a}) = (w(a_0), \cdots, w(a_{m-1}))$ where $w(a_i)$ is the Hamming weight of a_i. Let $\boldsymbol{k} = (k_0, k_1, \cdots, k_{m-1})$ and $\boldsymbol{k^*} = (k_0^*, k_1^*, \cdots, k_{m-1}^*)$ be two vectors in $\mathbb{Z}^m$. Define $\boldsymbol{k} \succeq \boldsymbol{k^*}$ if $k_i \geq k_i^*$ holds for all $i = 0, 1, \cdots, m-1$. Otherwise we write $\boldsymbol{k} \not\succeq \boldsymbol{k^*}$.

Bit Product Function $\pi_u(x)$ and $\boldsymbol{\pi_u}(\boldsymbol{x})$: For any $u \in \mathbb{F}_2^n$, let $\pi_u(x)$ be a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. For any $x \in \mathbb{F}_2^n$, define $\pi_u(x)$ as follows:

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

Let $\boldsymbol{\pi_u}(\boldsymbol{x})$ be a function from $(\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times \cdots \times \mathbb{F}_2^{n_{m-1}})$ to $\mathbb{F}_2$ for all $\boldsymbol{u} \in (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times, \cdots, \times \mathbb{F}_2^{n_{m-1}})$. For any $\boldsymbol{u} = (u_0, u_1, \cdots, u_{m-1}), \boldsymbol{x} = (x_0, x_1, \cdots, x_{m-1}) \in (\mathbb{F}_2^{n_0} \times \mathbb{F}_2^{n_1} \times, \cdots, \times \mathbb{F}_2^{n_{m-1}})$, define $\boldsymbol{\pi_u}(\boldsymbol{x})$ as follows:

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

2.2 Division Property

Division property [18] is a generalized integral property which can exploit the properties hidden between traditional integral properties $\mathcal{A}$ and $\mathcal{B}$. Thus, by

propagating division property we desire to get some better distinguishers. In the following we will introduce division property and present some propagation rules.

Definition 1 (Division Property [17]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $(\mathbb{F}_2^n)^m$, and $\boldsymbol{k}$ be an m-dimensional vector whose coordinates take values between 0 and n. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}^{n,m}_{\boldsymbol{k}^{(0)},\boldsymbol{k}^{(1)},\cdots,\boldsymbol{k}^{(q-1)}}$, it fulfills the following conditions: The parity of $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ over all $\boldsymbol{x} \in \mathbb{X}$ is always even when*

$$\boldsymbol{u} \in \left\{(u_0, u_1, \cdots, u_{m-1}) \in (\mathbb{F}_2^n)^m | W(\boldsymbol{u}) \not\succeq \boldsymbol{k^{(0)}}, \cdots, W(\boldsymbol{u}) \not\succeq \boldsymbol{k^{(q-1)}}\right\}$$

Proposition 1 (Copy [17]). *Denote $\mathbb{X}$ an input multiset whose elements belong to $\mathbb{F}_2^n$, and let $x \in \mathbb{X}$. The copy function creates (y_0, y_1) from x where $y_0 = x, y_1 = x$. Assuming the input multiset has division property $\mathcal{D}^n_k$, let $\mathbb{Y}$ be the corresponding output multiset, then $\mathbb{Y}$ has division property $\mathcal{D}^{n,2}_{(0,k),(1,k-1),\cdots,(k,0)}$.*

Proposition 2 (Compression by And [24]). *Denote $\mathbb{X}$ an input multiset whose elements belong to $\mathbb{F}_2^n \times \mathbb{F}_2^n$, let $(x_0, x_1) \in \mathbb{X}$ be an input to the compression function and denote the ouput value by y where $y = x_0 \& x_1$. Let $\mathbb{Y}$ be the corresponding output multiset. If input multiset $\mathbb{X}$ has division property $\mathcal{D}^{n,2}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_0, k_1)$, then the division property of $\mathbb{Y}$ is $\mathcal{D}^n_k$ where $k = \max\{k_0, k_1\}$.*

Proposition 3 (Compression by Xor [17]). *Denote $\mathbb{X}$ an input multiset whose elements belong to $\mathbb{F}_2^n \times \mathbb{F}_2^n$, let $(x_0, x_1) \in \mathbb{X}$ be an input to the compression function and denote the ouput value by y where $y = x_0 \oplus x_1$. Let $\mathbb{Y}$ be the corresponding output multiset. If input multiset $\mathbb{X}$ has division property $\mathcal{D}^{n,2}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_0, k_1)$, then the division property of $\mathbb{Y}$ is $\mathcal{D}^n_{k_0+k_1}$.*

Proposition 4 (Substitution [17]). *Denote $\mathbb{X}$ an input multiset whose elements belong to $\mathbb{F}_2^{n_1}$, let F be a substitution function (Sbox) with algebraic degree d and F maps an element in $\mathbb{F}_2^{n_1}$ to an element in $\mathbb{F}_2^{n_2}$, denote $\mathbb{Y}$ the corresponding output multiset $F(\mathbb{X})$. Assuming the input multiset has division property $\mathcal{D}^{n_1}_k$, then the output multiset has division property $\mathcal{D}^{n_2}_{\lceil \frac{k}{d} \rceil}$. Moreover, if $n_1 = n_2$ and the substitution function is bijective, assuming the input multiset has division property $\mathcal{D}^{n_1}_{n_1}$, then the output multiset has division property $\mathcal{D}^{n_1}_{n_1}$*

For more details regarding division property we refer the readers to [17,18,19].

2.3 Modeling a Subset in $\{0,1\}^n$ by Linear Inequalities

Convex Hull and **H-Representation**: The convex hull of a set A of points is the smallest convex set that contains A, and the H-Representation of a convex set is a set of linear inequalities $\mathcal{L}$ corresponding to the intersection of some halfspaces such that the feasible solutions of $\mathcal{L}$ are exactly the convex set.

In [14,15] Sun *et al.* treat a differential $(x_{u-1}, \cdots, x_0) \to (y_{v-1}, \cdots, y_0)$ of an $u \times v$ Sbox as an $(u+v)$-dimensional vector $(x_{u-1}, \cdots, x_0, y_{v-1}, \cdots, y_0)$.

By computing the H-Representation of the convex hull of all possible input-output differential pairs of an Sbox, a set of linear inequalities will be returned to characterize the differential propagation. Moreover, they proved that for a given subset A of $\{0,1\}^n$, it is always feasible to choose a set of linear inequalities $\mathcal{L}$ from the H-Representation of the convex hull of A, such that A represents all feasible solutions of $\mathcal{L}$ restricted in $\{0,1\}^n$.

Theorem 1 ([14]). *Let A be a subset of $\{0,1\}^n$, and denote $\mathrm{Conv}(A)$ the convex hull of A. For any $x \in \{0,1\}^n$, $x \in \mathrm{Conv}(A)$ if and only if $x \in A$.*

Thus, they first computed a set of vectors A which is composed of all differential pairs of a given Sbox, and then calculated the H-Representation of the convex hull of A by using the inequality_generator() function in the Sage [2] software, and this will return a set of linear inequalities $\mathcal{L}$ which are the H-presentation of $\mathrm{Conv}(A)$. According to Theorem 1, $\mathcal{L}$ is an accurate description of the difference propagations of the given Sbox, that is, all feasible solutions of $\mathcal{L}$ restricted in $\{0,1\}^n$ are exactly A. Since $\mathcal{L}$ is the H-Representation of $\mathrm{Conv}(A)$, each possible differential characteristic corresponds to a point in A, thus, each possible differential characteristic satisfies the linear inequalities in $\mathcal{L}$. On the other hand, for any impossible differential characteristic id, there always exists at least one linear inequality in $\mathcal{L}$ such that id does not satisfy this inequality. Otherwise, if id satisfies all the inequalities in $\mathcal{L}$ which indicates id belongs to $\mathrm{Conv}(A)$, and this is equivalent to $id \in A$.

Since $\mathcal{L}$ is an accurate description of A, adding all the linear inequalities in $\mathcal{L}$ into the MILP problem when searching differential characteristics of a block cipher, it will always return valid differential characteristics. However, the number of linear inequalities in the H-Representation of $\mathrm{Conv}(A)$ is often very large such that adding all the inequalities into the MILP model will make the problem computationally infeasible. Thus, Sun *et al.* [14] proposed a greedy algorithm (See Algorithm 1) to select a subset of $\mathcal{L}$ whose feasible solutions restricted in $\{0,1\}^n$ are exactly A. This algorithm can greatly reduce the number of inequalities required to accurately describe A.

In order to illustrate the procedure of this section, we present a toy example in Appendix A.

3 Modeling Division Property Propagations of Basic Operations and Sbox by Linear Inequalities

In [18] Todo introduced division property by using some vectors in $\mathbb{Z}^m$, and the propagation of division property through a round function of the block cipher is actually a transition of the vectors. Given an initial division property $\mathcal{D}_{\boldsymbol{k}}^{n,m}$, let f_r denote the round function of a block cipher, the division property of the state after one round f_r can be computed from $\mathcal{D}_{\boldsymbol{k}}^{n,m}$ by the rules introduced in [17,18], and denote the division property after one round f_r by $\mathcal{D}_{\mathbb{K}}^{n,m}$ where $\mathbb{K}$ is a set of vectors in $\mathbb{Z}^m$. Thus, the division property propagation through f_r is actually the transition from $\boldsymbol{k}$ to the vectors in $\mathbb{K}$. Traditionally, if two

Algorithm 1: Select a subset of linear inequalities from $\mathcal{L}$

Input : $\mathcal{L}$: the set of all inequalities in the H-Representation of Conv(A) with A a subset of $\{0,1\}^n$

Output: A subset $\mathcal{L}^*$ of $\mathcal{L}$ whose feasible solutions restricted in $\{0,1\}^n$ are A

```
begin
    L* = ∅
    B = {0,1}^n \ A
    L̄ = L
    while B ≠ ∅ do
        l ← The inequality in L̄ which maximizes the number of points in B
            that do not satisfy this inequality (choose the first one if there are
            multiple such inequalities).
        B* ← The points in B that do not satisfy l.
        L* = L* ∪ {l}
        L̄ = L̄ \ {l}
        B = B \ B*
    end
    return L*
end
```

vectors $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ in $\mathbb{K}$ satisfying that $\boldsymbol{k}_1 \succeq \boldsymbol{k}_2$, then $\boldsymbol{k}_1$ is redundant and will be removed from $\mathbb{K}$. However, since the redundant vectors do not influence the division property, in this paper we do not remove redundant vectors in $\mathbb{K}$, that is for any vector derived from $\boldsymbol{k}$ by using the propagation rules we add this vector into $\mathbb{K}$. Moreover, for any vector $\bar{\boldsymbol{k}}$ in $\mathbb{K}$, we call that $\boldsymbol{k}$ can propagate to $\bar{\boldsymbol{k}}$ through f_r.

Definition 2 (Division Trail). *Let f_r denote the round function of an iterated block cipher. Assume the input multiset to the block cipher has initial division property $\mathcal{D}_{\boldsymbol{k}}^{n,m}$, and denote the division property after i-round propagation through f_r by $\mathcal{D}_{\mathbb{K}_i}^{n,m}$. Thus, we have the following chain of division property propagations:*

$$\{\boldsymbol{k}\} \stackrel{def}{=} \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots$$

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

Proposition 5. *Denote the division property of input multiset to an iterated block cipher by $\mathcal{D}_{\boldsymbol{k}}^{n,m}$, let f_r be the round function. Denote*

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

the r-round division property propagation. Thus, the set of the last vectors of all r-round division trails which start with $\boldsymbol{k}$ is equal to $\mathbb{K}_r$.

Generally, given an initial division property $\mathcal{D}_{\boldsymbol{k}}^{n,m}$, and if one would like to check whether there exists useful integral property after r-round encryption, we have to propagate the initial division property for r rounds to get $\mathcal{D}_{\mathbb{K}_r}^{n,m}$ and check all the vectors in $\mathbb{K}_r$. According to Proposition 5, it is equivalent to find all r-round division trails which start with $\boldsymbol{k}$, and check the last vectors in the division trails to judge if any exploitable distinguisher can be extracted. Based on this observation, in the following we focus on how to accurately describe all division trails.

A linear inequality system will be adopted to describe division property propagations, that is we will construct a linear inequality system such that the feasible solutions represent all division trails. Since division property propagation is a deterministic procedure, the constructed linear inequality system must satisfy:

- For each division trail, it must satisfy all linear inequalities in the linear inequality system. That is each division trail corresponds to a feasible solution of the linear inequality system.
- Each feasible solution of the linear inequality system corresponds to a division trail. That is all feasible solutions of the linear inequality system do not contain any impossible division trail.

A linear inequality system satisfying the above two conditions is an accurate description of division property propagation. In the rest of the paper, we only consider bit-based division property. We start by modeling bit-based division property propagation of some basic operations and Sbox in block ciphers.

3.1 Modeling Copy, And and Xor

In this subsection, we show how to model bit-wise *Copy, And* and *Xor* operations by linear inequalities.

Modeling Copy. Copy operation is the basic operation used in Feistel block cipher. The left half of the input is copied into two equal parts, one of which is fed to the round function. Since we consider bit-based division property, the division property propagation of each bit is independent of each other. Thus, we consider only a single bit.

Let $\mathbb{X}$ be an input multiset whose elements take a value of $\mathbb{F}_2$. The copy function creates $y = (y_0, y_1)$ from $x \in \mathbb{X}$ where $y_0 = x$ and $y_1 = x$. Assuming the input multiset has division property $\mathcal{D}_k^1$, then the corresponding output multiset has division property $\mathcal{D}_{(0,k),\cdots,(k,0)}^1$ from Proposition 1. Since we consider bit-based division property, the input multiset division property $\mathcal{D}_k^1$ must satisfy $k \leq 1$. If $k = 0$, the output multiset has division property $\mathcal{D}_{(0,0)}^1$, otherwise if $k = 1$, the output multiset has division property $\mathcal{D}_{(0,1),(1,0)}^1$. Thus, $(0) \xrightarrow{copy} (0,0)$ is the only division trail given the initial division property $\mathcal{D}_0^1$, and $(1) \xrightarrow{copy} (0,1)$, $(1) \xrightarrow{copy} (1,0)$ are the two division trails given the initial division property $\mathcal{D}_1^1$.

Now we are ready to give a linear inequality description of these division trails. Denote $(a) \xrightarrow{copy} (b_0, b_1)$ a division trail of Copy function, the following inequality[4] is sufficient to describe the division propagation of Copy.

$$\begin{cases} a - b_0 - b_1 = 0 \\ a, b_0, b_1 \text{ are binaries} \end{cases} \tag{1}$$

Apparently, all feasible solutions of the inequalities in (1) corresponding to (a, b_0, b_1) are $(0, 0, 0)$, $(1, 0, 1)$ and $(1, 1, 0)$, which are exactly the three division trails of Copy function described above.

Modeling And. Bit-wise And operation is a basic nonlinear function, it is the only nonlinear operation for SIMON family. Similar to the modeling procedure of Copy function, we can express its division property propagation as a set of linear inequalities.

Let $\mathbb{X}$ be an input multiset whose elements take a value of $\mathbb{F}_2 \times \mathbb{F}_2$. The And function creates $y = x_0 \& x_1$ from $x = (x_0, x_1) \in \mathbb{X}$. Assuming the input multiset has division property $\mathcal{D}^{1,2}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_0, k_1)$, the division property of the corresponding output multiset is $\mathcal{D}^1_k$ where $k = \max\{k_0, k_1\}$ according to Proposition 2. Since we consider bit-based division property here, $\boldsymbol{k} = (k_0, k_1)$ must satisfy $0 \leq k_0, k_1 \leq 1$. Thus, there are four division trails for And function which are $(0,0) \xrightarrow{Xor} (0), (0,1) \xrightarrow{Xor} (1), (1,0) \xrightarrow{Xor} (1)$ and $(1,1) \xrightarrow{Xor} (1)$. Denote $(a_0, a_1) \xrightarrow{and} (b)$ a division trail of And function, the following linear inequalities are sufficient to describe this propagation features.

$$\begin{cases} b - a_0 \geq 0 \\ b - a_1 \geq 0 \\ b - a_0 - a_1 \leq 0 \\ a_0, a_1, b \text{ are binaries} \end{cases} \tag{2}$$

It is easy to check that all feasible solutions of the inequalities in (2) corresponding to (a_0, a_1, b) are $(0, 0, 0)$, $(0, 1, 1)$, $(1, 0, 1)$ and $(1, 1, 1)$, which are exactly the four division trails of And function described above.

Modeling Xor. Bit-wise Xor is another basic operation used in block ciphers. Similarly, a linear inequality system can be constructed to describe the division property propagation through Xor function.

Let $\mathbb{X}$ denote an input multiset whose elements take a value of $\mathbb{F}_2 \times \mathbb{F}_2$. The Xor function creates $y = x_0 \oplus x_1$ from $x = (x_0, x_1) \in \mathbb{X}$. Assuming the input multiset $\mathbb{X}$ has division property $\mathcal{D}^{1,2}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_0, k_1)$, thus, the corresponding output multiset $\mathbb{Y}$ has division property $\mathcal{D}^1_{k_0+k_1}$. Since we consider bit-based

[4] In this paper we do not make a distinction between equality and inequality, since the MILP problem use both equalities and inequalities as constraints.

division property here, $\boldsymbol{k} = (k_0, k_1)$ must satisfy $0 \leq k_0, k_1 \leq 1$. Moreover, the element of $\mathbb{Y}$ takes a value in $\mathbb{F}_2$, the division property $\mathcal{D}^1_{k_0+k_1}$ of $\mathbb{Y}$ must satisfy $k_0 + k_1 \leq 1$. That is, if $(k_0, k_1) = (1, 1)$, the division property propagation will abort. Thus, there are three valid division trails: $(0, 0) \xrightarrow{Xor} (0)$, $(0, 1) \xrightarrow{Xor} (1)$ and $(1, 0) \xrightarrow{Xor} (1)$. Let $(a_0, a_1) \xrightarrow{Xor} (b)$ denote a division trail through Xor function, the following inequality can describe the division trail through Xor function.

$$\begin{cases} a_0 + a_1 - b = 0 \\ a_0, a_1, b \text{ are binaries} \end{cases} \tag{3}$$

We can check that all the feasible solutions of inequality (3) corresponding to (a_0, a_1, b) are $(0, 0, 0)$, $(0, 1, 1)$ and $(1, 0, 1)$, which are exactly the division trails described above.

3.2 Modeling Sbox

Sbox is an important component of block ciphers, for a lot of block ciphers it is the only non-linear part. In [17,18], the Sbox is treated as a whole and the division property is considered while the element in the input multiset taking a value in $\mathbb{F}_2^n$ for an n-bit Sbox. In [19] Todo *et al.* introduced bit-based division property, but they only applied their technique to non-Sbox based ciphers SIMON and SIMECK. In this section, we study bit-based division property propagation through Sbox.

Assume we are dealing with an n-bit Sbox, the input and output of the Sbox are elements in $(\mathbb{F}_2)^n$. Suppose that the input multiset $\mathbb{X}$ has division property $\mathcal{D}^{1,n}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_0, k_1, \cdots, k_{n-1})$, that is for any $\boldsymbol{u} \in (\mathbb{F}_2)^n$ the parity of $\boldsymbol{\pi_u}(\boldsymbol{x})$ over $\mathbb{X}$ is even only if $W(\boldsymbol{u}) \not\succeq \boldsymbol{k}$. Note that for bit-based division property it holds $W(\boldsymbol{u}) = \boldsymbol{u}$, thus, we do not make a distinction between $W(\boldsymbol{u})$ and $\boldsymbol{u}$ in the following. To compute the division property of the output multiset $\mathbb{Y}$, we first consider a naive approach.

Previous Approach. First by Concatenation function, each element in $\mathbb{X}$ can be converted into an element in $\mathbb{F}_2^n$. Denote output multiset of Concatenation function as $\mathbb{X}^*$, thus, the division property of $\mathbb{X}^*$ is $\mathcal{D}^n_{k_0+k_1+\cdots+k_{n-1}}$ according to Rule 5 in [17]. Secondly, we pass each element in $\mathbb{X}^*$ to the Substitution function Sbox, and denote the output multiset by $\mathbb{Y}^*$ whose elements take a value of $\mathbb{F}_2^n$. According to Proposition 4, the division property of $\mathbb{Y}^*$ is $\mathcal{D}^n_{\left\lceil \frac{k_0+k_1+\cdots+k_{n-1}}{d} \right\rceil}$ where d is the algebraic degree of the Sbox. At last, for any value $y^* = y_0||y_1||\cdots||y_{n-1}$ in $\mathbb{Y}^*$, a Split function creates $y = (y_0, y_1, \cdots, y_{n-1})$ from y^*. Apparently, the output multiset of Split function equals to $\mathbb{Y}$. According to Rule 4 in [17], the division property of $\mathbb{Y}$ is $\mathcal{D}^{1,n}_{\boldsymbol{k^0},\boldsymbol{k^1},\cdots}$ where $\boldsymbol{k^i} = (k_0^i, k_1^i, \cdots, k_{n-1}^i)$ $(i \geq 0)$ denote all solutions of $x_0 + x_1 + \cdots, x_{n-1} = \left\lceil \frac{k_0+k_1+\cdots+k_{n-1}}{d} \right\rceil$.
Example: Take the Sbox used in PRESENT as an example. The PRESENT Sbox is a 4×4 Sbox with algebraic degree three. Assume that the input multiset

to the Sbox has division property $\mathcal{D}^{1,4}_{(0,1,1,1)}$. To compute the output multiset division property, we can proceed in three steps as described above: First by a concatenation function we convert the input multiset into another multiset $\mathbb{X}^*$ whose elements take a value in $\mathbb{F}_2^4$, thus the division property of $\mathbb{X}^*$ is $\mathcal{D}^4_3$. Secondly, make each value in $\mathbb{X}^*$ pass through the Sbox operation and this will result in a multiset $\mathbb{Y}^*$ with division property $\mathcal{D}^4_{\lceil\frac{3}{3}\rceil} = \mathcal{D}^4_1$. Finally, we split each value in $\mathbb{Y}^*$ into a value in $(\mathbb{F}_2)^4$, and we will get a multiset $\mathbb{Y}$ with division property $\mathcal{D}^{1,4}_{(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)}$. Thus, we have obtained four division trails of Sbox: $(0,1,1,1) \xrightarrow{Sbox} (0,0,0,1)$, $(0,1,1,1) \xrightarrow{Sbox} (0,0,1,0)$, $(0,1,1,1) \xrightarrow{Sbox} (0,1,0,0)$ and $(0,1,1,1) \xrightarrow{Sbox} (1,0,0,0)$.

Note that only the algebraic degree is exploited to calculate the division trails of Sbox in this naive approach. From the example illustrated above, if the input multiset to the Sbox has division property $\mathcal{D}^{1,4}_{(0,1,1,1)}$, the corresponding output multiset does not balance on any of the four output bits. However, this is not actually true. Denote the input to PRESENT Sbox as $\boldsymbol{x} = (x_3, x_2, x_1, x_0)$, and the corresponding output as $\boldsymbol{y} = (y_3, y_2, y_1, y_0)$, the algebraic normal form (ANF) of PRESENT Sbox is listed as follows:

$$\begin{cases} y_3 = 1 \oplus x_0 \oplus x_1 \oplus x_3 \oplus x_1x_2 \oplus x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \\ y_2 = 1 \oplus x_2 \oplus x_3 \oplus x_0x_1 \oplus x_0x_3 \oplus x_1x_3 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \\ y_1 = x_1 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \\ y_0 = x_0 \oplus x_2 \oplus x_3 \oplus x_1x_2 \end{cases} \tag{4}$$

Thus,

$$\boldsymbol{\pi}_{(0,0,0,1)}((y_3, y_2, y_1, y_0)) = y_0$$

and

$$\begin{aligned} &\bigoplus_{x\in\mathbb{X}} \boldsymbol{\pi}_{(0,0,0,1)}((y_3, y_2, y_1, y_0)) \\ =&\bigoplus_{x\in\mathbb{X}} y_0 \\ =&\bigoplus_{x\in\mathbb{X}} (x_0 \oplus x_2 \oplus x_3 \oplus x_1x_2) \\ =&\bigoplus_{x\in\mathbb{X}} \boldsymbol{\pi}_{(0,0,0,1)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \boldsymbol{\pi}_{(0,1,0,0)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \boldsymbol{\pi}_{(1,0,0,0)}(\boldsymbol{x}) \oplus \bigoplus_{x\in\mathbb{X}} \boldsymbol{\pi}_{(0,1,1,0)}(\boldsymbol{x}) \\ =&0 + 0 + 0 + 0 \\ =&0 \end{aligned}$$

As illustrated above, the least significant bit y_0 of the output $\boldsymbol{y}$ is balanced. Similarly, we can check that y_2 and y_0y_2 are all balanced. Furthermore, it can be observed that the expressions of y_1 and y_3 all contain monomial $x_0x_1x_2$ whose parity over $\mathbb{X}$ is undetermined according to the initial division property $\mathcal{D}^{1,4}_{(0,1,1,1)}$, thus y_1 and y_3 are not balanced. Based on these observations, the

division property of $\mathbb{Y}$ should be $\mathcal{D}^{1,4}_{(0,0,1,0),(1,0,0,0)}$. In this case we obtain two division trails of PRESENT Sbox, and what is more important is that y_0, y_2 and y_0y_2 are all balanced under this approach.

Our Improved Approach. Now we present a generalized algorithm to calculate division trails of an Sbox based on bit-based division property. In Algorithm 2, $\boldsymbol{x} = (x_{n-1}, \cdots, x_0)$ and $\boldsymbol{y} = (y_{n-1}, \cdots, y_0)$ denote the input and output to an n-bit Sbox respectively, and y_i is expressed as a boolean function of $(x_{n-1}, \cdots, x_0)$.

Algorithm 2: Calculating division trails of an Sbox

Input : The input division property of an n-bit Sbox $\mathcal{D}^{1,n}_{\boldsymbol{k}}$ where $\boldsymbol{k} = (k_{n-1}, \cdots, k_0)$

Output: A set $\mathbb{K}$ of vectors such that the output multiset has division property $\mathcal{D}^{1,n}_{\mathbb{K}}$

```
1  begin
2      S̄ = {k̄ | k̄ ⪰ k}
3      F(X) = {π_k̄(x) | k̄ ∈ S̄}
4      K̄ = ∅
5      for u ∈ (F_2)^n do
6          if π_u(y) contains any monomial in F(X) then
7              K̄ = K̄ ∪ {u}
8          end
9      end
10     K = SizeReduce(K̄)
11     return K
12 end
```

We explain Algorithm 2 line by line:

Line 2-3 According to input division property $\mathcal{D}^{1,n}_{\boldsymbol{k}}$, the parity of monomial $\boldsymbol{\pi}_{\bar{\boldsymbol{k}}}(\boldsymbol{x})$ with $\bar{\boldsymbol{k}} \succeq \boldsymbol{k}$ over $\mathbb{X}$ is undetermined, and we store these monomials in $F(X)$. Thus, the parity of any monomial that does not belong to $F(X)$ is zero.

Line 4 Initialize $\mathbb{K}$ as an empty set.

Line 5-9 For any possible $\boldsymbol{u}$, if boolean function $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ contains any monomial in $F(X)$, the parity of $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ over $\mathbb{X}$ is undetermined, and we store all these vectors in $\bar{\mathbb{K}}$.

Line 10 ***SizeReduce***() function removes all redundant vectors in $\bar{\mathbb{K}}$. Since we are interested in finding a set $\mathbb{K}$ such that for any $\boldsymbol{u} \in \{\boldsymbol{u} | \boldsymbol{u} \not\succeq \boldsymbol{k}$ for all $\boldsymbol{k} \in \mathbb{K}\}$, the parity of $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is zero. Note that for any vector $\boldsymbol{u} \in (\mathbb{F}_2)^n \backslash \bar{\mathbb{K}}$, the parity of $\boldsymbol{\pi}_{\boldsymbol{u}}(\boldsymbol{y})$ is zero, thus, we must have $\{\boldsymbol{u} \mid \boldsymbol{u} \not\succeq \boldsymbol{k}$ for all $\boldsymbol{k} \in \mathbb{K}\} \subset (\mathbb{F}_2)^n \backslash \bar{\mathbb{K}}$, and if we let $\mathbb{K}$= ***SizeReduce***$(\bar{\mathbb{K}})$ it will meet this condition. Otherwise, if there exists a vector $\boldsymbol{u} \in \{\boldsymbol{u} \mid \boldsymbol{u} \not\succeq \boldsymbol{k}$ for all $\boldsymbol{k} \in \mathbb{K}\}$ such that

$\boldsymbol{u} \notin (\mathbb{F}_2)^n \backslash \bar{\mathbb{K}}$, thus, we have $\boldsymbol{u} \in \bar{\mathbb{K}}$, which meants either $\boldsymbol{u} \in \mathbb{K}$ or there exists a vector $\boldsymbol{u}^* \in \mathbb{K}$ such that $\boldsymbol{u} \succeq \boldsymbol{u}^*$ since $\mathbb{K} = \boldsymbol{SizeReduce}(\bar{\mathbb{K}})$. In either case it won't happen $\boldsymbol{u} \in \{\boldsymbol{u} \mid \boldsymbol{u} \not\succeq \boldsymbol{k} \text{ for all } \boldsymbol{k} \in \mathbb{K}\}$, which leads to a contradiction. Therefore, $\mathbb{K}$ is sufficient to characterize the division property of output multiset.

Line 11 Return $\mathbb{K}$ as output.

Given an Sbox and an initial division property $\mathcal{D}_{\boldsymbol{k}}^{1,n}$, Algorithm 2 returns the output division property $\mathcal{D}_{\mathbb{K}}^{1,n}$. Thus for any vector $\boldsymbol{k}^* \in \mathbb{K}$, $(\boldsymbol{k}, \boldsymbol{k}^*)$ is a division trail of the Sbox. If we try all the 2^n possible input multiset division property, we will get a full list of division trails. Table 4 in Appendix B presents a complete list of all the 47 division trails of PRESENT Sbox.

Note that bit-based division property of an Sbox is closely related with Boura and Canteaut's work [6]. However, Boura and Canteaut's work is established on parity set, while our results are directly deduced from bit-based division property.

Representing the Division Trails of Sbox as Linear Inequalities. Each division trail of an n-bit Sbox can be viewed as a $2n$-dimensional vector in $\{0,1\}^{2n} \subset \mathbb{R}^{2n}$ where $\mathbb{R}$ is the real numbers field. Thus, all division trails form a subset P of $\{0,1\}^{2n}$. Next, we compute the H-Representation of Conv(P) by using the inequality_generator() function in the Sage [2] software, and this will return a set of linear inequalities $\mathcal{L}$. However, $\mathcal{L}$ contains too many inequalities which will make the size of corresponding MILP problem too large to solve. Fortunately, we can select a subset $\mathcal{L}^*$ of $\mathcal{L}$ by Algorithm 1 such that the feasible solutions of $\mathcal{L}^*$ restricted in $\{0,1\}^{2n}$ are exactly P.

Example: PRESENT Sbox contains 47 division trails which forms a subset P of $\{0,1\}^8$. By using the inequality_generator() function in the Sage software, a set of 122 linear inequalities will be returned. Furthermore, this set can be reduced by Algorithm 1 and we will get a set $\mathcal{L}^*$ of only 11 inequalities. The 11 inequalities for PRESENT Sbox are listed in Appendix C. In order to get the solutions of $\mathcal{L}^*$ restricted in $\{0,1\}^8$, we only need to specify that all variables can only take values in $\{0,1\}$.

So far, we have studied calculating and modeling division trails of basic operations and Sbox, thus, for block ciphers based on these operations and (or) Sbox, we can construct a set of linear inequalities which characterize one round division property propagation. By repeating this procedure r times, we can get a linear inequality system $\mathcal{L}$ such that all feasible solutions of $\mathcal{L}$ are all r-round division trails.

3.3 Initial Division Property

Integral distinguisher search algorithm often has a given initial division property $\mathcal{D}_{\boldsymbol{k}}^{1,n}$. Even though $\mathcal{L}$ is able to describe all division trails, we are interested in

division trails starting from the given initial division property. Thus, we have to model the initial division property into the linear inequality system. Denote $(a_{n-1}^0, \cdots, a_0^0) \rightarrow \cdots \rightarrow (a_{n-1}^r, \cdots, a_0^r)$ an r-round division trail, $\mathcal{L}$ is thus a linear inequality system defined on variables a_i^j $(i = 0, \cdots, n-1.\ \ j = 0, \cdots, r)$ and some auxiliary variables. Let $\mathcal{D}_{\boldsymbol{k}}^{1,n}$ denote the initial input division property with $\boldsymbol{k} = (k_{n-1}, \cdots, k_0)$, we need to add $a_i^0 = k_i$ $(i = 0, \cdots n-1)$ into $\mathcal{L}$, and thus all feasible solutions of $\mathcal{L}$ are division trails which start from vector $\boldsymbol{k}$.

4 Stopping Rule and Search Algorithm

In this section we first study the stopping rule in the search of integral distinguishers based on division property, and then we convert this stopping rule into an objective function of the MILP problem. At last, we propose an algorithm to determine whether an r-round integral distinguisher exists.

In the division property propagation, we note that only zero vector can propagate to zero vector. Thus if the given initial division property is $\mathcal{D}_{\boldsymbol{k}}^{1,n}$ with $\boldsymbol{k}$ a non-zero vector, and we denote the division property after r-round propagation by $\mathcal{D}_{\mathbb{K}_r}^{1,n}$, then it holds that $\mathbb{K}_r$ does not contain zero vector. In the following, we always assume $\boldsymbol{k} \neq \boldsymbol{0}$, since $\boldsymbol{k} = \boldsymbol{0}$ does not imply any integral property on the input multiset.

4.1 Stopping Rule

Let's first consider a set $\mathbb{X}$ with division property $\mathcal{D}_{\mathbb{K}}^{1,n}$. If $\mathbb{X}$ does not have any useful integral property, that is the Xor-sum of $\mathbb{X}$ does not balance on any bit, thus we have $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi_u}(\boldsymbol{x})$ is unknown for any unit vector $\boldsymbol{u} \in (\mathbb{F}_2)^n$. Since $\mathbb{X}$ has division property $\mathcal{D}_{\mathbb{K}}^{1,n}$, there must exist a vector $\boldsymbol{k} \in \mathbb{K}$ such that $\boldsymbol{u} \succeq \boldsymbol{k}$. Note that $\boldsymbol{u}$ is a unit vector, thus $\boldsymbol{u} = \boldsymbol{k}$, which means $\mathbb{K}$ contains all the n unit vectors. On the other hand, if $\mathbb{K}$ contains all the n unit vectors, then for any $\boldsymbol{0} \neq \boldsymbol{u} \in (\mathbb{F}_2)^n$ there must exist a unit vector $\boldsymbol{e} \in \mathbb{K}$ such that $\boldsymbol{u} \succeq \boldsymbol{e}$, that is $\bigoplus_{\boldsymbol{x}\in\mathbb{X}} \boldsymbol{\pi_u}(\boldsymbol{x})$ is unknown. Thus, $\mathbb{X}$ does not have any integral property.

Proposition 6 (Set without Integral Property). *Assume $\mathbb{X}$ is a multiset with division property $\mathcal{D}_{\mathbb{K}}^{1,n}$, then $\mathbb{X}$ does not have integral property if and only if $\mathbb{K}$ contains all the n unit vectors.*

Denote the output division property after i-round encryption by $\mathcal{D}_{\mathbb{K}_i}^{1,n}$, and the initial input division property by $\mathcal{D}_{\boldsymbol{k}}^{1,n} \stackrel{def}{=} \mathcal{D}_{\mathbb{K}_0}^{1,n}$. If $\mathbb{K}_{r+1}$ for the first time contains all the n unit vectors, the division property propagation should stop and an r-round distinguisher can be derived from $\mathcal{D}_{\mathbb{K}_r}^{1,n}$. In this case, $\mathbb{K}_r$ does not contain all n unit vectors, thus we can always find a unit vector $\boldsymbol{e}$ such that $\boldsymbol{e} \notin \mathbb{K}_r$. Since $\boldsymbol{e}$ is a unit vector, it holds $\boldsymbol{e} \not\succeq \boldsymbol{k}$ for all $\boldsymbol{k} \in \mathbb{K}_r$. Therefore, the parity of $\boldsymbol{\pi_e}(\boldsymbol{x})$ over r-round outputs is even which is a zero-sum property, thus

a balanced bit of the output is found. By repeating this process, all balanced bits can be found.

Based on this observation, we only need to detect whether $\mathbb{K}_r$ contains all unit vectors. According to Proposition 5, in order to check the vectors in $\mathbb{K}_r$, it is equivalent to check the last vectors of all r-round division trails. Denote $(a_{n-1}^0, \cdots, a_0^0) \to \cdots \to (a_{n-1}^r, \cdots, a_0^r)$ an r-round division trail, and let $\mathcal{L}$ denote a linear inequality system whose feasible solutions are all division trails which start with the given initial division property. It is clear that $\mathcal{L}$ is a linear inequality system defined on variables a_i^j $(i = 0, \cdots, n-1.\ \ j = 0, \cdots, r)$ and some auxiliary variables. Thus, we can set the objective function as :

$$Obj : Min\{a_0^r + a_1^r + \cdots a_{n-1}^r\} \tag{5}$$

Now we get a complete MILP problem by setting $\mathcal{L}$ as constraints and Obj as objective function. Note that $\mathbb{K}_i$ does not contain zero vector, in this case, the objective function will never take a value of zero, and the MILP problem will return an objective value greater than zero (if the MILP problem has feasible solutions). In the following we show how to determine whether r-round integral distinguisher exists based on this MILP problem.

4.2 Search Algorithm

Denote $\mathcal{L}$ a linear inequality description of all r-round division trails with the given initial input division property $\mathcal{D}_{\boldsymbol{k}}^{1,n}$. Let the sum of the coordinates of the last vector in the division trail be the objective function Obj as in Equation (5). Denote $M(\mathcal{L}, Obj)$ the MILP problem composed of $\mathcal{L}$ and Obj. Algorithm 3 will return whether r-round integral distinguisher exists.

Our MILP problems are solved by the openly available MILP optimizer Gurobi [1], Algorithm 3 is presented with some Gurobi syntax. We denote the set of last vectors of all division trails by $\mathbb{K}_r$.

Line 2 Initialize $\mathbb{S}$ as all possible output bit positions.

Line 3-24 For an n-bit block cipher, check how many unit vectors there are in $\mathbb{K}_r$. Moreover, we remove the bit position marked by the unit vectors in $\mathbb{K}_r$ from $\mathbb{S}$, and return $\mathbb{S}$ as the output of the algorithm.

- **Line 4** Check whether the MILP problem has a feasible solution. Note that the initial MILP problem always has feasible solutions. However, along with the execution of the procedure, it will add some constraints (Line 13) in the model which will possibly make the MILP problem unsolvable.
- **Line 5** Optimize the MILP problem M by Gurobi.
- **Line 6-18** $M.ObjVal$ is Gurobi syntax which returns the current value of the objective function after M has been optimized. $M.ObjVal = 1$ means we have found a division trail which ends up with a unit vector $\boldsymbol{e}$, thus $\boldsymbol{e} \in \mathbb{K}_r$. $M.getObjective()$ is a Gurobi function which returns the objective function of the model, which is $a_0^r + \cdots + a_{n-1}^r$ in our case. The functionality of Line 8-17 is to choose which variable of $(a_0^r, \cdots, a_{n-1}^r)$

Algorithm 3: Return whether r-round distinguisher exists

```
   Input  : M = M(L, Obj).
   Output: A set S of balanced bit positions.
 1 begin
 2     S = {a_0^r, ··· , a_{n-1}^r}
 3     for i in range(0,n) do
 4         if M has feasible solutions then
 5             M.optimize()
 6             if M.ObjVal = 1 then
 7                 obj = M.getObjective()
 8                 for i in range(0,n) do
 9                     var = obj.getVar(i)
10                     val = var.getAttr('x')
11                     if val = 1 then
12                         S \ {var}
13                         M.addConstr(var = 0)
14                         M.update()
15                         break
16                     end
17                 end
18             else
19                 return S
20             end
21         else
22             return S
23         end
24     end
25     return S
26 end
```

is equal to one in $\boldsymbol{e}$ and add a new constraint $var = 0$ into M, here var denotes the variable taking a value of one. $obj.getVar(i)$ is used to return the i-th variable of obj which is a_i^r in this case. $var.getAttr('x')$ retrieves the value of var under the current solution. Line 12 removes var from $\mathbb{S}$, since we have found $\boldsymbol{e} \in \mathbb{K}_r$ whose nonzero position is var which means var can't be a balanced bit position. $M.addConstr(var = 0)$ adds a new constraints $var = 0$ into M, and this is used to rule out $\boldsymbol{e}$ from $\mathbb{K}_r$. Line 14 updates the model since we have added a new constraint.

Line 19 This step returns $\mathbb{S}$, the execution of this step means the objective value of M is larger than one, that is we can no longer find a division trail with the last vector being a unit vector. In this case, we have found all unit vectors in $\mathbb{K}_r$ which represent undetermined bit positions, and thus we have ruled out all unbalanced bits and get an integral distinguisher.

Line 22 M do not have any feasible solutions means we have ruled out all units vectors of $\mathbb{K}_r$ and made $\mathbb{K}_r$ an empty set along with the execution. In this case, we can return $\mathbb{S}$ as output since we have checked all vectors.

Line 25 If the for loop do not make the procedure exit, return $\mathbb{S}$ as output. Usually, in this case $\mathbb{S}$ is an empty set which means no distinguisher found.

Algorithm 3 always returns a set $\mathbb{S}$ indicating balanced bit positions. For a block cipher with a given initial division property $\mathcal{D}_{\boldsymbol{k}}^{1,n}$, we can construct an r-round linear description of division property propagations and use Algorithm 3 to check whether a distinguisher exists. If for the first time the $(r+1)$-round model returns an empty set, then the longest distinguisher for the given initial division property is r-round.

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

In this section, we show some applications of our technique. All the source codes are avaiable at `https://github.com/xiangzejun/MILP_Division_Property`. We applied our algorithm to SIMON, SIMECK, PRESENT, RECTANGLE, LBlock and TWINE block ciphers. The results are listed in Table 1. The *Round (Previous)* column and *Round (Sect. 5)* column list the number of rounds of the distinguishers of previous and our results. The *Data* column represents the number of active bits of the input pattern of the integral distinguisher, the data complexity of the distinguisher is determined by the initial input division property. *Balanced bits* column represents the number of balanced bits of the distinguisher we found. *Time* presents the time used by Algorithm 3 for searching the corresponding distinguishers, among which s is short for second and m is short for minute. All the experiments are conducted on the following platform: Intel Core i7-2600 CPU @3.40GHz, 8.00G RAM, 64-bit Windows 7 system. Moreover, the distinguishes listed in Table 1 are presented in Appendix E. The table shows that we get improved distinguishers for SIMON48/64/96/128, SIMECK48/64, PRESENT and RECTANGLE. For SIMECK32, LBlock and TWINE our results are consistent with the previous best results. The result of SIMON32 is one round less than the result in [19]. However, we only use bit-based division property here, the 15-round distinguisher found in [19] for SIMON32 used bit-based division property using three subset. If bit-based division property is the only technique adopted, 14-round distinguisher is the longest distinguisher we can find.

5.1 Applications to SIMON and SIMECK

SIMON [3] is a family of lightweight block ciphers published by the U.S. National Security Agency (NSA) in 2013. SIMON adopts Fesitel structure and it has a very compact round function which only involves bit-wise And, Xor and circular shift operations. The structure of one round SIMON encryption is depicted in Fig 1 where S^i denotes left circular shift by i bits.

1-round Description of SIMON: Denote one round division trail of SIMON2n by $(a_0^i, \cdots, a_{n-1}^i, b_0^i, \cdots, b_{n-1}^i) \rightarrow (a_0^{i+1}, \cdots, a_{n-1}^{i+1}, b_0^{i+1}, \cdots, b_{n-1}^{i+1})$. In order to

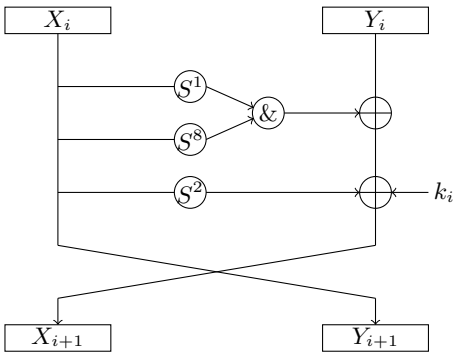

Fig. 1. Feistel Structure of SIMON Round Function

get a linear description of all possible division trails of one round SIMON, we introduce four vectors of auxiliary variables which are $(u_0^i, \cdots, u_{n-1}^i)$, $(v_0^i, \cdots, v_{n-1}^i)$, $(w_0^i, \cdots, w_{n-1}^i)$ and $(t_0^i, \cdots, t_{n-1}^i)$. We denote $(u_0^i, \cdots, u_{n-1}^i)$ the input division property of S^1. Similarly, denote $(v_0^i, \cdots, v_{n-1}^i)$ and $(w_0^i, \cdots, w_{n-1}^i)$ the input division property of S^8 and S^2 respectively. Let $(t_0^i, \cdots, t_{n-1}^i)$ denote the output division property of bit-wise And operation. Subsection 3.1 has modeled Copy, And and Xor functions. According to Equation (1), the following inequalities are sufficient to model the Copy operation used in SIMON2n:

$$\mathcal{L}_1 : a_j^i - u_j^i - v_j^i - w_j^i - b_j^{i+1} = 0 \text{ for } j \in \{0, 1, \cdots, n-1\}$$

Since we consider bit-based division property, division property propagation through circular shift is just a circular shift of the coordinates of the vector. Thus, the division property of the output of S^1 is $(u_1^i, \cdots, u_{n-1}^i, u_0^i)$. Similarly, the division property of the output of S^8 and S^2 are $(v_8^i, \cdots, v_6^i, v_7^i)$ and $(w_2^i, \cdots, w_0^i, w_1^i)$ respectively. We can model bit-wise And operation used in SIMON by the following inequalities according to Equation (2):

$$\mathcal{L}_2 : \begin{cases} t_j^i - u_{j+1}^i \geq 0 & \text{for } j \in \{0, 1, \cdots, n-1\} \\ t_j^i - v_{j+8}^i \geq 0 & \text{for } j \in \{0, 1, \cdots, n-1\} \\ t_j^i - u_{j+1}^i - v_{j+8}^i \leq 0 & \text{for } j \in \{0, 1, \cdots, n-1\} \end{cases}$$

At last, the Xor operations in SIMON2n can be modeled by the following inequalities according to Equation (3):

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

So far, we have modeled all operations used in SIMON, and get an accurate description $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$ of 1-round division trails. By repeating this procedure r times, we can get a linear inequality system $\mathcal{L}$ for r-round division property propagation. Given some initial division property, we can add the corresponding

constrains into $\mathcal{L}$ and estimate whether a useful distinguisher exists by Algorithm 3. The results for SIMON family are listed in Table 1.

For SIMON48/64/96/128, we found the best distinguishers so far. Note that by using bit-based division property under the framework of [19], it is computationally impractical to search distinguishers for these versions. Using Algorithm 3, distinguishers can be searched in practical time.

SIMECK [25] is a family of lightweight block cipher proposed at CHES 2015. The round function of SIMECK is very like SIMON except the rotation constants. We applied our technique to SIMECK, and 15-, 18- and 21-round distinguishers are found for SIMECK32, SIMECK48 and SIMECK64 respectively, which shows that SIMON has better security than SIMECK with respect to division property based integral cryptanalysis.

We found that the 14-round distinguisher of SIMON32 we found is the same as the 14-round distinguisher of SIMON32 in [19] based on bit-based division property. Surprisingly, the 15-round distinguisher for SIMECK32 in [19] is found by bit-based division property using three subsets, however, we also find the same distinguisher for SIMECK32 by only using bit-based division property.

In [9], the authors investigated the differential and linear behavior of SIMON family regarding rotation parameters, and they presented some interesting alternative parameters among which $(1, 0, 2)$ is optimal for the differential and linear characteristics with the restriction that the second rotation parameter is zero. In this paper, we investigated the integral property of this parameter by our technique. The results are listed in Table 2 (h in the *time* column represents hour). The third column lists the rounds of the distinguishers we found. The results show that $(1, 0, 2)$ is a very bad choice with respect to division property based integral cryptanalysis.

Table 2. Results on SIMON(1,0,2).

Cipher	Block size	Round	Data	Balanced bits	time
SIMON32(1,0,2)	32	20	31	1	34.1s
SIMON48(1,0,2)	48	28	47	1	3.2m
SIMON64(1,0,2)	64	36	63	1	10.3m
SIMON96(1,0,2)	96	52	95	3	6.4h
SIMON128(1,0,2)	128	68	127	3	24h

5.2 Applications to PRESENT and RECTANGLE

PRESENT [5] and RECTANGLE [28] are two SP-network block ciphers, of which the linear layers are bit permutations. Fig 2 illustrates one round encryption of PRESENT.

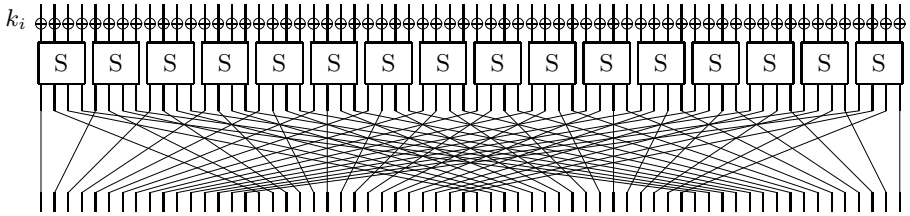

Fig. 2. One Round SP Structure of PRESENT

1-round Description of PRESENT: Denote one round division trail of PRESENT by $(a_{63}^i, \cdots, a_0^i) \to (a_{63}^{i+1}, \cdots, a_0^{i+1})$. We first model the division property propagation of Sbox layer. Denote the division property of the output of Sbox by $(b_{63}^i, \cdots, b_0^i)$. Subsection 3.2 has studied how to calculate the division trails of Sbox and model those trails by linear inequalities. Appendix C shows the 11 inequalities of PRESENT Sbox. For each of the 16 Sboxes of PRESENT, we introduce 11 inequalities and thus the Sbox layer of PRESENT can be modeled by $11 \times 16 = 176$ inequalities which is denoted by $\mathcal{L}_1$. The linear layer of PRESENT is a bit permutation, thus, the division property propagation through linear layer is just a permutation of the coordinates of the vector, that is

$$\mathcal{L}_2 : \begin{cases} a_{16j \mod 63}^{i+1} = b_j^i & j \in \{0, 1, \cdots, 62\} \\ a_j^{i+1} = b_j^i & j = 63 \end{cases}$$

Note that $\mathcal{L}_1$ is a linear inequality system defined on variables $(a_{63}^i, \cdots, a_0^i)$ and $(b_{63}^i, \cdots, b_0^i)$, we can use the equalities in $\mathcal{L}_2$ to replace the variables $(b_{63}^i, \cdots, b_0^i)$ in $\mathcal{L}_1$ in order to save auxiliary variables.

Now we have get a linear inequality system to describe one round division propagation of PRESENT. By repeating this procedure, an r-round linear inequality system can be constructed. For a given initial division property $\mathcal{D}_{\boldsymbol{k}}^{1,64}$, we add this information into the linear inequality system and use Algorithm 3 to estimate whether there exists an integral distinguisher.

The result for PRESENT is listed in Table 1. We found a 9-round integral distinguisher for PRESENT which is two more rounds than the previous best results in [22].

The modeling procedure of RECTANGLE is very like to PRESENT, we only list the result here in Table 1. The previous longest integral distinguisher of RECTANGLE is found by the designers, and they gave a 7-round distinguisher. In this paper we find a 9-round distinguisher which is two more rounds.

5.3 Applications to LBlock and TWINE

This subsection applies our technique to two generalized Feistel block cipher LBlock and TWINE. The round function of these two ciphers are alike, and the round function composed of Copy, Sbox and Xor operations. We have showed

how to model Copy and Xor operations in SIMON and Sbox in PRESENT, thus, we omit the details for these two ciphers due to the limit of space. The number of division trails and linear inequalities required to describe those division trails of LBlock and TWINE Sboxes are presented in Table 3. The $\sharp\{D.C\}$ column represents the number of division trails of the corresponding Sbox, and the $\sharp\{Ine\}$ column represents the number of linear inequalities we found to accurately describe the division trails. Note that we chose the first inequality in the sixth line of Algorithm 1, however, other choice rather than the first one may result in different set of inequlities.

Table 3. Sbox Properties Regrading Division Trails.

Sbox	$\sharp\{D.C\}$	$\sharp\{Ine\}$
PRESENT Sbox	47	11
RECTANGLE Sbox	49	17
LBlock S0	44	11
LBlock S1	44	12
Lblock S2	44	12
LBlock S3	44	11
LBlock S4	44	13
LBlock S5	44	10
LBlock S6	44	12
LBlock S7	44	12
TWINE Sbox	47	11

Our experimental results regarding LBlock and TWINE are listed in Table 1. The distinguishers found in this paper are the same as the distinguishers found for these two ciphers in [26].

Experiments. To illustrate the validity of the technique proposed in this paper, we presented some integral distinguishers found by our technique with a small number of active bits, and we run experiments on these distinguishers. The experiments are presented at Appendix D. Our experiments showed that the distinguishers found by our technique are sound. Moreover, the results on PRESENT and RECTANGLE illustrate that our technique can find quite accurate distinguishers, that is the balanced bits found by Algorithm 3 are exactly in accordance with experimental results. For PRESENT cipher, we retrieved and improved the 5-round distinguisher found in [22], our technique found all the four balanced bits of the outputs of the fifth round given the same input pattern as in [22], while Wu *et al.* could only prove the balancedness of only one bit.

6 Summary and Discussion

In this paper we introduced a new technique to search integral distinguishers based on bit-based division property. We first proposed a new notion *division trail* and used this new notion to characterize the division property propagation, then we showed that it is sufficient to check the last vectors of all r-round division trails in order to estimate whether an r-round distinguisher exists.

Based on the observations on division trails, we proposed to construct a linear inequality system to characterize the division property propagations. We first studied how to model division property propagations of *Copy, And* and *Xor* operations by linear inequalities. For another basic component Sbox used in block ciphers, we studied the bit-based division property propagations for the first time, and we proposed an algorithm to compute the division trails of an Sbox. Moreover, we used those division trails to derive a set of linear inequalities whose feasible solutions are exactly all division trails. Thus, for a block cipher we can construct a linear inequality system whose solutions are all r-round division trails of the cipher, and we used this linear inequality system as constraints of the MILP problem. Then, the stopping rule in the search of integral distinguisher were studied and we converted it into an objective function of an MILP problem. To be specific, we set the sum of the coordinates of the last vector in an r-round division trail as objective function. Thus, we can get a complete MILP problem, based on which we presented an algorithm to estimate whether an r-round integral distinguisher exists by checking how many unit vectors are contained in the last vectors of all division trails.

We applied our technique to SIMON, SIMECK, PRESENT, RECTANGLE, LBlock and TWINE. For SIMON48/64/96/128, SIMECK48/64, PRESENT and RECTANGLE, we get much longer distinguishers than previous results based on division property in the open literature. Moreover, our results on PRESENT and RECTANGLE show that we can get better integral distinguishers by using the algebraic normal form of the Sboxes. Our results show that, by using our technique, we can search integral distinguishers based on bit-based division property in practical time for block ciphers with block size larger than 32, which is impractical under the traditional framework.

In [19], Todo *et al.* also introduced bit-based division property using three subsets, and they found 15-round distinguisher for SIMON32. However, we have not found a way to model this framework by an MILP problem at present. A surprising result is, by using our technique we also derived the 15-round distinguisher of SIMECK32 which are constructed by bit-based division property using three subsets [19]. We also used our technique on some Sbox-based block ciphers such as PRESENT and RECTANGLE, note that their linear layers are all bit permutations. However, this technique can be easily extended to arbitrary linear layers as pointed out by the reviewers, since any linear layer can be viewed as bit-level linear layer which can be treated as bit-wise copy and Xor.

Acknowledgements. We are very grateful to the anonymous reviewers. This work was supported by the National Natural Science Foundation of China (Grant

No. 61379138), the "Strategic Priority Research Program" of the Chinese Academy of Sciences (Grant No. XDA06010701).

A An Example

Let's consider a simple example in this section. Suppose that $A = \{(0,1),(1,0),(1,1)\}$ is a subset of $\{0,1\}^2$ with three points, and we would like to get a linear inequality system $\mathcal{L}$ such that all feasible solutions of $\mathcal{L}$ restricted in $\{0,1\}^2$ are A.

We proceed by using inequality_generator() function in the Sage software to compute the H-Representation of Conv(A). The following is the source code.

```
Points = [[0,1],[1,0],[1,1]]
triangle = Polyhedron(vertices = Points)
for l in triangle.inequality_generator():
    print l
```

As a result, Sage returns three inequalities:

$$\mathcal{L} = \begin{cases} x + y - 1 \geq 0 \\ -y + 1 \geq 0 \\ -x + 1 \geq 0 \end{cases} \tag{6}$$

It is easy to check that the feasible solutions of $\mathcal{L}$ form a triangle with A being its three vertices, and the set of all feasible solutions of $\mathcal{L}$ restricted in $\{0,1\}^2$ is exactly A. Thus, Equation 6 is a description of A.

However, we can use Algorithm 1 to reduce the number of inequalities required. We apply Algorithm 1 to this example and we find that only one inequality is sufficient to accurately describe A:

$$\mathcal{L}^* = \{x + y - 1 \geq 0\} \tag{7}$$

It is easy to check that all solutions of $\mathcal{L}^*$ restricted in $\{0,1\}^2$ are $(0,1)$,$(1,0)$ and $(1,1)$ as expected.

B Division trails of PRESENT and RECTANGLE Sbox

Table 4 and Table 5 present the division trails of PRESENT and RECTANGLE Sboxes respectively.

C Linear inequalities description of PRESENT and RECTANGLE Sbox

The following inequalities are the 11 inequalities used to describe PRESENT Sbox whose feasible solutions are exactly the 47 division trails of PRESENT

Table 4. Division trails of PRESENT Sbox

Input $\mathcal{D}_{\boldsymbol{k}}^{1,4}$	Output $\mathcal{D}_{\mathbb{K}}^{1,4}$
(0,0,0,0)	(0,0,0,0)
(0,0,0,1)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,0,1,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,0,1,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,0,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,0,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,1,0)	(0,0,0,1) (0,0,1,0) (1,0,0,0)
(0,1,1,1)	(0,0,1,0) (1,0,0,0)
(1,0,0,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,0,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,1,0)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,1,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,1,0,0)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,1,0,1)	(0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,1,1,0)	(0,1,0,1) (1,0,1,1) (1,1,1,0)
(1,1,1,1)	(1,1,1,1)

Table 5. Division trails of RECTANGLE Sbox

Input $\mathcal{D}_{\boldsymbol{k}}^{1,4}$	Output $\mathcal{D}_{\mathbb{K}}^{1,4}$
(0,0,0,0)	(0,0,0,0)
(0,0,0,1)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,0,1,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,0,1,1)	(0,0,0,1) (0,1,0,0) (1,0,1,0)
(0,1,0,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(0,1,0,1)	(0,0,1,1) (0,1,0,0) (1,0,0,0)
(0,1,1,0)	(0,0,1,1) (0,1,0,0) (1,0,0,0)
(0,1,1,1)	(0,0,1,1) (0,1,0,0) (1,0,0,1)
(1,0,0,0)	(0,0,0,1) (0,0,1,0) (0,1,0,0) (1,0,0,0)
(1,0,0,1)	(0,0,1,1) (0,1,0,1) (0,1,1,0) (1,0,0,0)
(1,0,1,0)	(0,0,1,0) (0,1,0,1) (1,0,0,0)
(1,0,1,1)	(0,1,1,0) (1,0,1,1) (1,1,0,1)
(1,1,0,0)	(0,0,1,1) (0,1,0,0) (1,0,0,0)
(1,1,0,1)	(0,1,1,0) (1,0,1,0) (1,1,0,1)
(1,1,1,0)	(0,0,1,1) (0,1,0,1) (1,0,0,0)
(1,1,1,1)	(1,1,1,1)

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

$$\mathcal{L}^* = \begin{cases} a_3 + a_2 + a_1 + a_0 - b_3 - b_2 - b_1 - b_0 \geq 0 \\ -a_2 - a_1 - 2a_0 + b_3 + b_1 - b_0 + 3 \geq 0 \\ -a_2 - a_1 - 2a_0 + 4b_3 + 3b_2 + 4b_1 + 2b_0 \geq 0 \\ -2a_3 - a_2 - a_1 + 2b_3 + 2b_2 + 2b_1 + b_0 + 1 \geq 0 \\ -2a_3 - a_2 - a_1 + 3b_3 + 3b_2 + 3b_1 + 2b_0 \geq 0 \\ -b_3 + b_2 - b_1 + b_0 + 1 \geq 0 \\ -2a_3 - 2a_2 - 2a_1 - 4a_0 + b_3 + 4b_2 + b_1 - 3b_0 + 7 \geq 0 \\ a_3 + a_2 + a_1 + a_0 - 2b_3 - 2b_2 + b_1 - 2b_0 + 1 \geq 0 \\ -4a_2 - 4a_1 - 2a_0 + b_3 - 3b_2 + b_1 + 2b_0 + 9 \geq 0 \\ -2a_0 - b_3 - b_2 - b_1 + 2b_0 + 3 \geq 0 \\ a_0 + b_3 - b_2 - 2b_1 - b_0 + 2 \geq 0 \\ a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0 \text{ are binaries} \end{cases} \tag{8}$$

The following inequalities are the 17 inequalities used to describe RECTANGLE Sbox whose feasible solutions are exactly the 49 division trails of RECTANGLE Sbox where $(a_3, a_2, a_1, a_0) \longrightarrow (b_3, b_2, b_1, b_0)$ denotes a division trail.

$$\mathcal{L}^* = \begin{cases} -a_3 - a_2 - 2a_1 - 3a_0 - 2b_3 + b_1 + 2b_0 + 6 \geq 0 \\ -b_3 - b_2 + b_0 + 1 \geq 0 \\ a_3 + a_2 + a_1 + a_0 - b_3 - b_2 - b_1 - b_0 \geq 0 \\ 3a_3 + a_2 - b_3 - 2b_2 - b_1 - 2b_0 + 2 \geq 0 \\ a_2 + a_0 - b_2 - 2b_1 - b_0 + 2 \geq 0 \\ -a_2 - a_1 - a_0 + b_3 + 2b_2 + 2b_0 + 1 \geq 0 \\ -2a_3 - a_1 - a_0 + b_3 + 2b_1 + b_0 + 2 \geq 0 \\ -3a_3 - a_2 - a_1 - 2a_0 + b_3 + 2b_2 + 2b_1 - b_0 + 4 \geq 0 \\ -a_2 - a_1 + b_3 + b_2 + b_1 + 1 \geq 0 \\ -3a_3 - a_2 - a_1 - 2a_0 + 3b_3 + 2b_2 + 2b_1 + b_0 + 2 \geq 0 \\ 2a_2 + 3a_1 - 3b_3 - b_2 - 2b_1 - b_0 + 3 \geq 0 \\ -a_3 - a_2 - a_0 + 2b_3 + 2b_2 + b_1 + b_0 \geq 0 \\ -2a_2 - a_1 - a_0 + 3b_3 + 4b_2 + 2b_1 + 2b_0 \geq 0 \\ a_3 + a_2 + a_1 + a_0 - 2b_3 - 2b_0 + 1 \geq 0 \\ 2a_0 - b_3 - b_2 - b_1 + 1 \geq 0 \\ 3a_3 - 4a_2 - a_1 - a_0 - 2b_3 - b_2 - 3b_1 + 2b_0 + 7 \geq 0 \\ a_3 + a_1 + a_0 + b_3 - 3b_2 - 2b_1 - 2b_0 + 3 \geq 0 \\ a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0 \text{ are binaries} \end{cases} \tag{9}$$

D Experiments on PRESENT and RECTANGLE

For SIMON family block ciphers, we found a 14-round distinguisher of SIMON32 which is in accordance with the distinguisher presented in [19]. For Lblock and TWINE the distinguisher found in this paper are in accordance with the distinguishers presented in [26]. Thus, we believe that the distinguishers found for SIMON, SIMECK, Lblock and TWINE are sound. In the following we only conduct some experiments on PRESENT and RECTANGLE.

PRESENT: We found the following 5-round distinguisher for PRESENT. If we fix the left most 60 bits as random constant and vary the right most 4 bits, then after five round encryption, the four right most bits of the state are balanced.

Input:(cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccaaaa)
Output:(??bbbb)
c: constant bit, a: active bit, ?: unknown bit, b: balanced bit

We run experiment on this distinguisher 2^{12} times. The experimental result returns the four right most bits as balanced bits which is in accordance with our theoretical result.

Note that in [22] Wu *el at.* found a 5-round distinguisher for PRESENT which has the same input pattern with the distinguisher presented here. However, they only proved that the right most bit is balanced. By using our technique, we can find all the four balanced bits.

RECTANGLE: We found the following 6-round distinguisher for RECTANGLE. The input of the distinguisher has 23 active bits, that is the right most six bits of the first, third and fourth rows, and the five right most bits of the second row are active. The output of six rounds encryption will be balanced on 40 bits, that is the first two rows, the two right most bits of the third row and the six left most bits of the last row.

$$\text{Input}: \begin{pmatrix} \texttt{ccccccccccaaaaaa} \\ \texttt{ccccccccccaaaaaa} \\ \texttt{ccccccccccaaaaaa} \\ \texttt{ccccccccccaaaaaa} \end{pmatrix} \longrightarrow \text{Output}: \begin{pmatrix} \texttt{bbbbbbbbbbbbbbbb} \\ \texttt{bbbbbbbbbbbbbbbb} \\ \texttt{????????????????bb} \\ \texttt{bbbbbb??????????} \end{pmatrix}$$

We run experiment on this distinguisher 2^{10} times. The experimental result returns 40 balanced bits which is in accordance with our theoretical result.

E Integral Distinguishers listed in Table 1

For SIMON and SIMECK family block ciphers, all distinguisher can be extended one more round by the technique in [20].

E.1 SIMON32's 13-round Distinguisher

Input:(caaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaa)
Output:(????????????????,bbbbbbbbbbbbbbbb)

E.2 SIMON48's 15-round Distinguisher

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

E.3 SIMON64's 17-round Distinguisher

Input:(caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output:(????????????????????????????????,bbbbbbbbbb?????b?????bbbbbbbbbb)

E.4 SIMON96's 21-round Distinguisher

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

E.5 SIMON128's 25-round Distinguisher

Input:(caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output:(??,
b?b??b?)

E.6 SIMECK32's 14-round Distinguisher

Input:(caaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaa)
Output:(????????????????,bb???bb???bb???b)

E.7 SIMECK48's 17-round Distinguisher

Input:(caaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaa)
Output:(????????????????????????,b???bb????????????bb???)

E.8 SIMECK64's 20-round Distinguisher

Input:(caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)
Output:(????????????????????????????????,bb???b??????????????????????b???b)

E.9 PRESENT's 9-round Distinguisher

Input:(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaacccc)
Output:(???b)

E.10 RECTANGLE's 9-round Distinguisher

$$\text{Input} : \begin{pmatrix} \texttt{caaaaaaaaaaaaaaa} \\ \texttt{caaaaaaaaaaaaaaa} \\ \texttt{caaaaaaaaaaaaaaa} \\ \texttt{caaaaaaaaaaaaaaa} \end{pmatrix} \longrightarrow \text{Output} : \begin{pmatrix} \texttt{bbb?b?bbbbbbbbbb} \\ \texttt{?????????????b?b} \\ \texttt{????????????????} \\ \texttt{????????????????} \end{pmatrix}$$

E.11 LBlock's 16-round Distinguisher

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

E.12 TWINE's 16-round Distinguisher

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

References

1. http://www.gurobi.com/
2. http://www.sagemath.org/
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive 2013, 404 (2013)
4. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. In: Advances in Cryptology-EUROCRYPT 2001, pp. 395–405. Springer (2001)
5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. Springer (2007)
6. Boura, C., Canteaut, A.: Another view of the division property. In: CRYPTO. pp. 654–682. Springer (2016)
7. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. In: Fast Software Encryption. pp. 149–165. Springer (1997)
8. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Fast Software Encryption. pp. 112–127. Springer (2002)
9. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Advances in Cryptology–CRYPTO 2015, pp. 161–185. Springer (2015)
10. Lucks, S.: The saturation attack-a bait for Twofish. In: Fast Software Encryption. pp. 1–15. Springer (2002)
11. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Information Security and Cryptology. pp. 57–76. Springer (2011)

12. Sun, B., Hai, X., Zhang, W., Cheng, L., Yang, Z.: New observation on division property. Science China Information Science (2016), http://eprint.iacr.org/2015/459
13. Sun, S., Hu, L., Song, L., Xie, Y., Wang, P.: Automatic security evaluation of block ciphers with s-bp structures against related-key differential attacks. In: Information Security and Cryptology. pp. 39–51. Springer (2013)
14. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Tech. rep., Cryptology ePrint Archive, Report 2014/747 (2014)
15. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block ciphers. In: Advances in Cryptology–ASIACRYPT 2014, pp. 158–178. Springer (2014)
16. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A lightweight block cipher for multiple platforms. In: Selected Areas in Cryptography. pp. 339–354. Springer (2012)
17. Todo, Y.: Integral cryptanalysis on full MISTY1. In: Advances in Cryptology–CRYPTO 2015, pp. 413–432. Springer (2015)
18. Todo, Y.: Structural evaluation by generalized integral property. In: Advances in Cryptology–EUROCRYPT 2015, pp. 287–314. Springer (2015)
19. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. Cryptology ePrint Archive, Report 2016/285 (2016), http://eprint.iacr.org/
20. Wang, Q., Liu, Z., Varıcı, K., Sasaki, Y., Rijmen, V., Todo, Y.: Cryptanalysis of reduced-round SIMON32 and SIMON48. In: Progress in Cryptology–INDOCRYPT 2014, pp. 143–160. Springer (2014)
21. Wu, S., Wang, M.: Security evaluation against differential cryptanalysis for block cipher structures. IACR Cryptology ePrint Archive 2011, 551 (2011)
22. Wu, S., Wang, M.: Integral attacks on reduced-round PRESENT. In: Information and Communications Security, pp. 331–345. Springer (2013)
23. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Applied Cryptography and Network Security. pp. 327–344. Springer (2011)
24. Xiang, Z., Zhang, W., Lin, D.: On the division property of SIMON48 and SIMON64. International Workshop on Security (2016)
25. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The SIMECK family of lightweight block ciphers. In: Cryptographic Hardware and Embedded Systems–CHES 2015, pp. 307–329. Springer (2015)
26. Zhang, H., Wu, W.: Structural evaluation for generalized feistel structures and applications to LBlock and TWINE. In: Progress in Cryptology–INDOCRYPT 2015, pp. 218–237. Springer (2015)
27. Zhang, H., Wu, W., Wang, Y.: Integral attack against bit-oriented block ciphers. In: International Conference on Information Security and Cryptology. pp. 102–118. Springer (2015)
28. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences 58(12), 1–15 (2015)
29. Zhang, W., Su, B., Wu, W., Feng, D., Wu, C.: Extending higher-order integral: an efficient unified algorithm of constructing integral distinguishers for block ciphers. In: Applied Cryptography and Network Security. pp. 117–134. Springer (2012)