Improved Division Property for Ciphers with Complex Linear Layers

Yongxia Mao[1,2], Wenling Wu[1,2](✉), Bolin Wang[1,2], and Li Zhang[1,2]

[1] Trusted Computing and Information Assurance Laboratory, Institute of Software Chinese Academy of Sciences, Beijing 100190, China
{yongxia2018,wenling,bolin2018,zhangli2021}@iscas.ac.cn
[2] University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. The division property proposed by Todo at EUROCRYPT 2015 as a generalized integral property has been applied to many symmetric ciphers. Automatic search methods of the division property assisted by modeling technique, such as Mixed Integer Linear Programming (MILP) and Boolean Satisfiability Problem (SAT), have become the most popular approach to searching integral distinguishers. The accuracy of the model in searching algorithms has an effect on the search results of integral distinguishers. For the block cipher, constructing an accurate and efficient model of the division property propagation on complex linear layers remains hard. This paper observes that the non-independent propagations of the bit-based division property (BDP) on complex linear layers can generate redundant division trails, which will affect the accuracy of the model if it is not taken into account in modeling. Based on this, we propose a method that can build a more accurate model by handling matrices containing non-independent propagations in the linear layer. To verify the effectiveness of our method, we apply the method to two block ciphers uBlock-128 and MIBS. For uBlock-128, our results improve the previous 8-round integral distinguisher by more balanced bits. For MIBS, a 9-round integral distinguisher is given for the first time, which is 4 rounds longer than the previous best.

Keywords: Division property · Linear layer · Block cipher · MILP · Cryptanalysis

1 Introduction

Integral cryptanalysis was originally proposed by Knudsen et al. [1] at FSE 2002, also known as Square attack [2], and is a powerful cryptanalysis method. Todo [3] further generalized integral cryptanalysis as division property at EUROCRYPT 2015. At ASIACRYPT 2016, Xiang et al. [4] introduced the MILP technique into bit-based division property for the first time, which improved the block size of the block cipher that can be automatically searched. Since then, the automatic modeling tool of integral cryptanalysis has been widely used in evaluating the security of symmetric encryotions, and a series of remarkable results have been

https://doi.org/10.1007/978-3-031-22301-3_6

obtained [5–10]. Recently, Hebborn et al. [11] demonstrated the upper bound on the round number of integral distinguishers on several block ciphers such as PRESENT and SKINNY-64, using the automatic modeling method of BDP at ASIACRYPT 2021.

The automatic search method with modeling technique mainly including MILP and SAT for division property can be summarized as follows. At first, the basic component model of block cipher needs to be established by following the propagation rules of division property. Next, the r-round model is built and the initial division property is given. At last, the entire model is solved with the help of solving tools, such as Gurobi and SAT/SMT solvers. In the process, the number of conditional constraints determines the time that it takes to solve and the round number of the integral distinguisher that can be obtained. For the linear component, it only needs to exchange the position of variables to build constaints when the linear layer is simple, such as PRESENT, GIFT, etc. When the linear layer is complex, such as AES, uBlock, MIBS, etc., it needs to follow the propagation rules of COPY/XOR operations to build the constraints, which often leads to redundancy and errors.

For the problem of how to model the complex linear layer, there are mainly S method [12], ZR method [13] and HW method [14]. The S method is a general method for modeling the division property propagation of complex linear layers. However, the disadvantage is that it does not consider the cancellation between terms, so it can easily introduce invalid division trails resulting in a quicker loss of the balanced property than the cipher itself would. Both the ZR method and HW method can create very accurate models, but their applications have certain limitations. For example, the ZR method needs to construct a one-to-one correspondence between the division trails of the invertible matrix M and the invertible sub-matrices, so it is not suitable for the non-binary linear layer and non-invertible matrices. The HW method can only be modeled by SAT and cannot be applied to the MILP model. As far as the current modeling methods of the division property for S-boxes, the accuracy of the SAT solving model is weaker than that of the MILP model. Hence, the HW method is not suitable for block ciphers with S-boxes.

Our Contribution. In this paper, we analyze why errors arise in the solving model of division property propagations for the complex linear layer. In other words, the *non-independent division property propagation* of variables in the linear layer will produce redundant division trails, which reduces the accuracy of the model, and ultimately affects the judgment of integral distinguishers. Then, we propose a strategy to effectively remove redundant division trails for the *non-independent division property propagation*: replacing the original representation of the linear layer with an equivalent one which only includes the *independent division property propagation*. According to this strategy, an algorithm (Algorithm 2) is proposed to construct the MILP model of BDP propagation for the complex linear layer. Finally, we apply our method to two block ciphers uBlock-128 and MIBS. For uBlock-128, we find an 8-round integral distinguisher where all the 128 bits are balanced. For MIBS, we find a 9-round integral distinguisher with 32 balanced bits which is better than the best known result. We list all our new BDP results obtained in Table 1.

Organization. The rest of this paper is organized as follows. Section 2 introduces some notations of this paper and revisits the definition related to division property. Section 3 presents our observations and the new method of this paper. Section 4 mainly presents improved integral distinguishers on uBlock-128 and MIBS by using our new method. Section 5 is the conclusion and outlook for future work.

Table 1. Number of rounds of the best known integral distinuisher vs. our results on the block cipher uBlock-128 and MIBS.

Cipher	# Rounds	log_2 (Data)	# Balanced bits	Reference
uBlock	7	124	128	[16]
	8	124	64	[17]
	8	124	96	Sect. 4.1
	8	127	128	Sect. 4.1
MIBS	5	8	8	[21]
	5	12	32	Sect. 4.2
	6	32	32	Sect. 4.2
	7	52	32	Sect. 4.2
	8	61	32	Sect. 4.2
	8	63	64	Sect. 4.2
	9	63	32	Sect. 4.2

2 Notations and Division Property

In this section, we are going to show some notations, and recall the fundamental definitions and modeling techniques of division property.

For a block cipher, we use the following notation to represent the integral property of a nibble in the plaintext and ciphertext.

- $\mathcal{C}$: Each bit of the nibble at the plaintext is fixed to constant.
- $\mathcal{A}$: All bits of the nibble at the plaintext are active.
- $\mathcal{B}$: Each bit of the nibble at the ciphertext is balanced.
- $\mathcal{U}$: A nibble at the ciphertext with unknown status.

For the integral property of a single bit in integral distinguishers, we use c, a, b, and u denote a constant bit, an active bit, a balanced bit and an unknown bit, respectively. For a matrix $M \in \mathbb{F}_2^{m \times n}$, we use the notation $M[i][j]$ to represent the element of M located at the i-th row and j-th column, $l_i = M[i]$ to represent the i-th row, and $M[*][j]$ to represent the j-th column. A bold letter represents a vector, e.g., $\boldsymbol{u} \in \mathbb{F}_2^m$. Let $\boldsymbol{k}$ and $\boldsymbol{k'}$ be two vectors in $\mathbb{F}_2^m$, we define $\boldsymbol{k} \succcurlyeq \boldsymbol{k'}$ if $k_i \geq k'_i$ for all i.

Bit Product Function [3]**:** Let $\pi_{\boldsymbol{u}} : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function for any $\boldsymbol{u} \in \mathbb{F}_2^n$. Let $\boldsymbol{x} \in \mathbb{F}_2^n$ be an input of $\pi_{\boldsymbol{u}}$, then $\pi_{\boldsymbol{u}}(\boldsymbol{x})$ is defined as

$$\pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{n-1} x_i^{u_i},$$

where $x_i^0 = 1$ and $x_i^1 = x_i$.

Definition 1 (Bit-based Division Property [15]**).** *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^m$, and $\boldsymbol{k} \in \mathbb{F}_2^m$. When the multiset $\mathbb{X}$ has the division property $D_{\mathbb{K}}^{1,m}$, it fulfils the following conditions:*

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

Definition 2 (Division Trail [4]**).** *Let f be the round function of an iterated block cipher. Assume the input multiset set to the block cipher has initial division property $\mathcal{D}_{\mathbb{K}_0}^{1,n}$, and denote the division property after i rounds through f by $\mathcal{D}_{\mathbb{K}_i}^{1,n}$. Then we have the following chain of division property propagations:*

$$\{\boldsymbol{k}\} \equiv \mathbb{K}_0 \xrightarrow{f} \mathbb{K}_1 \xrightarrow{f} \cdots \xrightarrow{f} \mathbb{K}_r.$$

For any vector $\boldsymbol{k}_{i+1}^* \in \mathbb{K}_{i+1}$, there must exist a vector $\boldsymbol{k}_i^* \in \mathbb{K}_i$ such that $\boldsymbol{k}_i^*$ can propagate to $\boldsymbol{k}_{i+1}^*$. Furthermore, for $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \mathbb{K}_2 \times \cdots \times \mathbb{K}_r)$, if for all $i \in \{0, 1, \cdots, r-1\}$, $\boldsymbol{k}_i$ can propagate to $\boldsymbol{k}_{i+1}$, we call $(\boldsymbol{k}_0 \to \boldsymbol{k}_1 \to \ldots \to \boldsymbol{k}_r)$ a r-round division trail.

MILP Modeling Rule for COPY [12]**.** If $a \xrightarrow{COPY} (b_0, b_1, \cdots, b_{m-1})$ is a division trail of COPY function, it is sufficient to describe the propagation using the following inequalities

$$\begin{cases} a - b_0 - b_1 - \cdots - b_{m-1} = 0, \\ a, b_0, b_1, \cdots, b_{m-1} \text{ are binaries.} \end{cases}$$

MILP Modeling Rule for XOR [12]**.** If $(a_0, a_1, \cdots, a_{m-1}) \xrightarrow{XOR} b$ is a division trail of XOR function, it is sufficient to describe the propagation using the following inequalities

$$\begin{cases} a_0 + a_1 + \cdots + a_{m-1} - b = 0, \\ a_0, a_1, \cdots, a_{m-1}, b \text{ are binaries.} \end{cases}$$

The Judgment Condition of Division Property. If there exists a division trail that satisfies $\boldsymbol{k}_0 \xrightarrow{E_k} \boldsymbol{k}_r = \boldsymbol{e}_j$, where $\boldsymbol{e}_j$ is a unit vector, $j \in \{0, \cdots, n-1\}$, then the j-th bit of ciphertext is unknown. If there is no division trail $\boldsymbol{k}_0 \xrightarrow{E_k} \boldsymbol{k}_r = \boldsymbol{e}_j$, then the j-th bit of ciphertext is balanced.

3 A Method to Reduce Redundant Division Trails for Complex Linear Layers

With the help of the MILP modeling rules, the linear inequalities of the division property propagation for the linear layer can be established. We can easily observe the following rules: for the same linear layer, the more COPY/XOR operations in MILP model are used, the lower accuracy achieved in characterizing division property propagations. For example, obviously, if the linear layer is a simple bit permutation, the model of the linear layer will not produce error and redundancy, because we only need to exchange the position of variables without adding extra constraints to model for describing the division property propagation on linear layer. For complex linear layers, the situation becomes different.

Assuming that $L : (x_0, x_1, x_2, x_3, x_4, x_5) \rightarrow (y_0, y_1, y_2, y_3, y_4, y_5)$ is a linear transformation corresponding to a complex linear layer, and the equivalent linear transformation matrix is M which always is a full rank matrix. In this paper, "complex" also means that there are at least two rows in M that have the element 1 on at least two of the same columns, i.e., there exist i, j s.t. $\sum_k M[i][k] \cdot M[j][k] \geq 2$. If we establish constaints based on the original matrix M, then at least two independent constraints about COPY on the input variables need to be added to the MILP model. In fact, the two COPY constraints are not independent since they occur simultaneously in the constraints about XOR for two different output variables. At this point, an error occurs. Similarly, when $\sum\limits_k M[i][k] \cdot M[j][k] = 4$, we need to add 4 independent constraints on COPY from M. Suppose $M[1] = (0, 1, 1, 1, 1, 1)$ and $M[2] = (1, 1, 0, 1, 1, 1)$. If we set $t_1 = x_1 + x_3$ and $t_2 = x_4 + x_5$ using the method in [18], then only two independent constraints on COPY and two independent constraints on XOR need to be added. However, the two COPY constraints are still not independent in fact because they appear in the XOR constraints of both y_1 and y_2. Errors in the MILP model eventually lead to redundant division trails through L.

Generally speaking, when several output bits contain multiple common input variables during propagations, the model established on some of the bits based on the MILP modeling rule for COPY cannot accurately describe the correlation between them. In other words, some non-independent bits are incorrectly propagated as independent bits if we build the MILP model of linear layers directly from the original matrix. As a result, the division property of multiple outputs is 1 at the same time in places where it should not be. Thus, there will be some redundant division trails. To this end, we give the following definition and observation.

Definition 3. *Let $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ be a linear transformation , defined as $F(x) = y$, and M be the equivalent linear transformation matrix satisfying $M \cdot x = y$. The division property of x and y is denoted as $a = (a_0, a_1, ..., a_{n-1})$ and $b = (b_0, b_1, ..., b_{m-1})$, respectively. That is, $a \xrightarrow{F} b$ is a division trail through F. We call $a_{i_1}, a_{i_2}, ..., a_{i_s} \xrightarrow{F} b_{j_1}, b_{j_2}, ..., b_{j_t}$ is a non-independent division prop-*

erty propagation through F, if $M[j_*][i_*] = 1$ for all $i_* \in \{i_1, i_2, ..., i_s\}$, $j_* \in \{j_1, j_2, ..., j_t\}$, $n \geq s \geq 2$, $m \geq t \geq 2$, i.e., $\sum_{k=0}^{n-1} M[j_1][k] \cdot M[j_2][k] \cdot \ldots \cdot M[j_t][k] \geq 2$, where $\{i_1, i_2, ..., i_s\}$ and $\{j_1, j_2, ..., j_t\}$ are two index sets. Otherwise, it is called an independent division property propagation.

Observation. When modeling the non-independent division property propagations, redundant division trails will be generated if we use the MILP model of independent propagations to characterize that of non-independent propagations. If all the common variables that lead to non-independent division property propagations in the linear function F are all replaced at once by variables $T_1, \cdots, T_N$ newly introduced, the propagation through F that are described in new expressions containing variables $T_1, \cdots, T_N$ will be transformed into independent division property propagations. In this way, the redundant division trails can be effectively reduced.

Example 1. Assume that the linear transformation $P : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ has following expressions:

$$\begin{aligned}
z_0 &= y_0 + y_1 + y_3 + y_4 + y_6 + y_7,\\
z_1 &= y_1 + y_2 + y_3 + y_4 + y_5 + y_6,\\
z_2 &= y_0 + y_1 + y_2 + y_4 + y_5 + y_7,\\
z_3 &= y_1 + y_2 + y_3 + y_6 + y_7,\\
z_4 &= y_0 + y_2 + y_3 + y_4 + y_7,\\
z_5 &= y_0 + y_1 + y_3 + y_4 + y_5,\\
z_6 &= y_0 + y_1 + y_2 + y_5 + y_6,\\
z_7 &= y_0 + y_2 + y_3 + y_5 + y_6 + y_7,
\end{aligned}$$

where $(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7)$ and $(z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7)$ are the inputs and outputs, respectively. Let the corresponding division property be $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$ and $(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7)$, respectively.

We take b_1 and b_3 as examples, and focus on division trails with the form of $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) \to (*, b_1, *, b_3, *, *, *, *)$. Let the input division property be $(0, 1, 1, 1, 0, 0, 1, 0)$, so we only consider the output division property with form $(0, *, 0, *, 0, 0, 0, 0)$. According to traditional models,

$$\begin{aligned}
b_1 &= a_1 + a_2 + a_3 + a_4 + a_5 + a_6,\\
b_3 &= a_1 + a_2 + a_3 + a_6 + a_7.
\end{aligned}$$

a_1, a_2, a_3, a_6 need to establish constraints separately by using the MILP modeling rule for COPY, so the output division property is $(0, 1, 0, 1, 0, 0, 0, 0)$, $(0, 1, 0, 0, 0, 0, 0, 0)$, and $(0, 0, 0, 1, 0, 0, 0, 0)$. For b_1, when i (≤ 4) variables of a_1, a_2, a_3, a_6 take 1, the output division property $(0, 1, 0, 1, 0, 0, 0, 0)$ will be obtained. However, since $(0, 1, 0, 1, 0, 0, 0, 0) \succcurlyeq (0, 1, 0, 0, 0, 0, 0, 0)$, $(0, 1, 0, 1, 0, 0, 0, 0)$ is redundant.

We noticed that $(*, a_1, a_2, a_3, *, *, a_6, *) \to (*, b_1, *, b_3, *, *, *, *)$ is the non-independent division property propagation, and a_1, a_2, a_3, a_6 are the common variables of b_1, b_3. If we denote $t = a_1 + a_2 + a_3 + a_6$, then $b_1 = t + a_4 + a_5$ and $b_3 = t + a_7$ are new expressions. At this point, we only need to do COPY operation once on t. For b_1, when any i variables of a_1, a_2, a_3, a_6 take 1, $i < 4$, that is, the t contained in b_1 takes 1, the other t contained in b_3 must take constant 0 after COPY operation, hence the output division property must be $(0, 1, 0, 0, 0, 0, 0, 0)$. After variable substitution, the output division property of the form $(0, *, 0, *, 0, 0, 0, 0)$ contains only $(0, 1, 0, 0, 0, 0, 0, 0)$, $(0, 0, 0, 1, 0, 0, 0, 0)$. Similarly, redundant vectors like the above can be removed by updating the linear layer expression using a series of variable substitutions.

Before building the constraints of the MILP model of the linear layer, we first pair the rows of the implementation matrix that is a linear transformation matrix so that we can extract the common variables for each pair. It is not difficult to find that for the entire reduction process, the more common variable that can be extracted, the better the effect of removing redundant trails and the more accurate the model. Following the reduction idea, We present Algorithm 1 for screening and pairing the rows of the linear transformation matrix.

Algorithm 2 is used to build the MILP model of the BDP propagation for the linear layer. Let the initial matrix of the linear layer be M, the matrix corresponding to the new variable that are introduced by replacing the common variable be B. A new matrix P with size $n \times (n + n/2)$ is generated in Algorithm 2 which is equivalent to M, and the last $n/2$ columns of P correspond to new binary variable $T_1, \cdots, T_N$. Additionally, we point out that Algorithm 2 can also make models using SAT method based on Algorithm 1, and they are equivalent.

Algorithm 1. Row pairing algorithm for a linear transformation matrix

```
Input: An implementation matrix $M$ of the linear layer
Output: A row partition of $M$
1: $count = 0_{n \times n}$   //a $n \times n$ all-zero matrix
2: for $i \in \{0, n\}, j \in \{0, n\}$ do
3:     if $i < j$ then
4:         $count[i][j] = \sum_{k=0}^{n-1} l_i^k \wedge l_j^k$
5:     else
6:         $count[i][j] = 0$
7:     end if
8: end for
9: while True do
10:    $Global\_max = max(count)$
11:    if $Global\_max = 0$ then
12:        break
13:    end if
14:    if $row\_max = Global\_max$ then //the row $row\_max$ is located
15:        $Row = (row\_max = Global\_max)[0]$
16:    end if
17:    for $j \in J = \{j | count[Row][j] = row\_max\}$ do
18:        if $max(count[j]) < row\_max$ then//the column $row\_max$ is located
```

```
19:             $Col = j$; break
20:         end if
21:     end for
22:     if $len(Col) = 0$ then
23:         $Col = J[0]$
24:     end if
25:     $result \leftarrow (Row, Col)$
26:     $count[Row] = 0, count[Col] = 0, count[*][Row] = 0, count[*][Col] = 0$
27: end while
28: return $result$
```

Algorithm 2. Construct the MILP model of linear layer BDP propagation

```
Input: An Implementation matrix $M$ of the linear layer
Output: The MILP model $\mathcal{M}$ of BDP propagation
 1: Build an empty MILP model $\mathcal{M}$
 2: $\mathcal{M}.var \leftarrow a_i, b_i, u_i$    //$a_i, b_i$ denote the input and output BDP of linear
    layer, $u_i$ denotes a new binary variable introduced by COPY
 3: $B = [\,]$   //a $n \times \frac{n}{2}$ all-zero matrix
 4: for $(i, j)$ in $result$ do
 5:     $B[i][N] = 1, B[j][N] = 1$
 6:     $l = \left(l_i^0 \wedge l_j^0, l_i^1 \wedge l_j^1, \ldots, l_i^n \wedge l_j^n\right)$
 7:     $M[i] = M[i] + l, M[j] = M[j] + l$
 8:     $Constr \leftarrow T_N = \sum_{k=0}^{n-1} x_k \cdot \left(l_i^k \wedge l_j^k\right)$
 9:     $Constr \leftarrow y_i = T_N + \sum_{k=0}^{n-1} x_k \cdot \left(l_i^k \bigoplus l^k\right), y_j = T_N + \sum_{k=0}^{n-1} x_k \cdot \left(l_j^k \bigoplus l^k\right)$
10:     $\mathcal{M}.con \leftarrow T_N = \sum_{k=0}^{n-1} a_k \cdot \left(l_i^k \wedge l_j^k\right)$
11: end for
12: $P = [M, B]$
13: for $j \in (0, n + n/2)$, $i \in (0, n)$ do
14:     $\mathcal{M}.con \leftarrow a_j = \sum_{k=0}^{n-1} u_k \cdot P[i][j]$
15: end for
16: for $i \in (0, n)$,$j \in (0, n + n/2)$ do
17:     $\mathcal{M}.con \leftarrow b_i = \sum_{k=0}^{n-1} u_k \cdot P[i][j]$
18: end for
19: return $\mathcal{M}$
```

Traditional models usually added the corresponding constraints based on the initial expression of the linear layer and combined with the BDP model of COPY/XOR operations. If there exist non-independent division property propagations in it, then redundant vectors must be produced, for example the method of [12]. Hong et al. [18] proposed a method using the optimal implementation of the linear layer to reduce the number of COPY/XOR constraints, and finally obtained the same integral distinguisher as the best at the time. [18] is an example of reducing errors caused by non-independent division property propagations, and it also supports our observation in this section, but the method of [18] is effective for the cipher like Midori-64, Skinny-64 and LED (because the non-independent propagation in the linear layer of these ciphers contains at most 3 common variables, our model contains i common variables, $i \geq 2$). In other words, for the cipher with complex linear layers, such as uBlock and MIBS ($i \geq 3$), only a part of the redundancy can be reduced using method in [18].

4 Applications

In this section, for two block ciphers uBlock-128 and MIBS, we first briefly introduce their encryption structures, then apply our new method to them, and finally show the complete process of constructing MILP models for them.

4.1 Application to uBlock-128

uBlock-128. uBlock is a block cipher family proposed by Wu et al. [16]. It adopts a SP network and supports 128-bit and 256-bit block lengths. Moreover, uBlock-128 supports 128-bit and 256-bit key lengths, and the number of encryption rounds are 16 and 24, respectively. Let the round function of uBlock-128 be $f = P \circ X \circ S$, where S represents the S layer, X represents the cyclic shift and XOR operation in the middle, and P represents the last nibble-based permutation. The round function is shown in Fig. 1, where s represents a 4-bit nonlinear S-box, and $\lll 4$ represents that a block rotates 4 bits to the left in units of 32. PL_{128} and PR_{128} represent two 8-byte vector permutations respectively, where $PL_{128} = \{1,3,4,6,0,2,7,5\}$ and $PR_{128} = \{2,7,5,0,1,6,4,3\}$.

uBlock is the winner in the National Cryptographic Algorithm Design Competition held by the Chinese Association for Cryptologic Research in 2019 because of its adaptability to software and hardware platforms, simple and effective hardware implementations, and strong security. For the integral cryptanalysis, based on Todo et al.'s conclusion on the division property of the (l,d,m)-SPN in [3], the uBlock designers presented a 7-round integral distinguisher [16]. Besides, the current optimal integral distinguisher is the 8-round distinguisher found by Tian et al. [17] by using the optimized representation of S-box in division property propagation.

Searching the Integral Distinguisher of uBlock-128. Let the input of the X layer be $x = (x_3, x_2, x_1, x_0)$ and the output be $y = (y_3, y_2, y_1, y_0)$. Then the linear matrix corresponding to the transformation X can be expressed as the juxtaposition of two nibble-based matrices $M_{16\times16}$, denoted as

$$M_{16\times16} = \begin{bmatrix} A_{8\times16} \\ B_{8\times16} \end{bmatrix}.$$

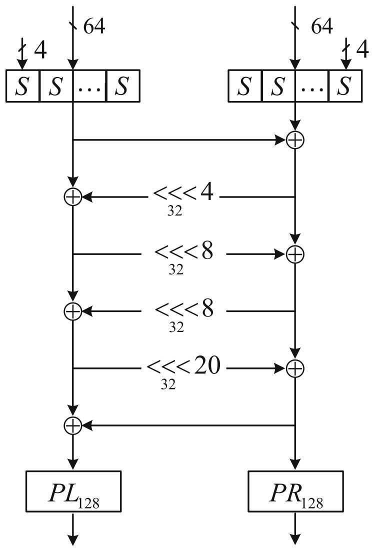

Fig. 1. Round function of uBlock-128.

$$A = \begin{bmatrix} 0&0&1&1&1&0&1&1&1&1&0&1&0&1&1&1\\ 1&0&0&1&1&1&0&1&1&1&1&0&1&0&1&1\\ 1&1&0&0&1&1&1&0&1&1&1&1&0&1&0&1\\ 0&1&1&0&0&1&1&1&1&1&1&1&1&0&1&0\\ 1&0&1&1&0&0&1&1&0&1&1&1&1&1&0&1\\ 1&1&0&1&1&0&0&1&1&0&1&1&1&1&1&0\\ 1&1&1&0&1&1&0&0&0&1&0&1&1&1&1&1\\ 0&1&1&1&0&1&1&0&1&0&1&0&1&1&1&1 \end{bmatrix} \quad B = \begin{bmatrix} 1&1&0&1&0&1&1&1&1&0&1&1&0&0&1&1\\ 1&1&1&0&1&0&1&1&1&1&0&1&1&0&0&1\\ 1&1&1&1&0&1&0&1&1&1&1&0&1&1&0&0\\ 1&1&1&1&1&0&1&0&0&1&1&1&0&1&1&0\\ 0&1&1&1&1&1&0&1&0&0&1&1&1&0&1&1\\ 1&0&1&1&1&1&1&0&1&0&0&1&1&1&0&1\\ 0&1&0&1&1&1&1&1&1&1&0&0&1&1&1&0\\ 1&0&1&0&1&1&1&1&0&1&1&0&0&1&1&1 \end{bmatrix}$$

In other words, the input x is transformed by the matrix $A_{8\times16}$ and then output y_2 and y_3, and x is also transformed by the matrix $B_{8\times16}$ and then output y_0 and y_1. It can be observed from the initial matrix A and B that any two of the rows contain multiple common columns with a constant 1. Thus the division property propagation of $(x_0, *, x_2, *) \rightarrow y_0, y_2$ and $(*, x_1, *, x_3) \rightarrow y_1, y_3$ are both non-independent. After operating on the linear transformation matrix A and B by using Algorithm 1 and Algorithm 2, we get the new matrix representation corresponding to the X transformation under the basis $(x_3, x_2, x_1, x_0, T_1, T_0)$. That is

$$A^{'} = \begin{bmatrix} 0&0&1&0&1&0&0&0&1&1&0&0&0&1&0&0&0&0&0&1&0&0&1&1\\ 0&0&0&1&0&1&0&0&0&1&1&0&0&0&1&0&1&0&0&0&1&0&0&1\\ 0&0&0&0&1&0&1&0&0&0&1&1&0&0&0&1&1&1&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&0&1&0\\ 1&0&0&0&0&0&1&0&0&1&0&0&1&1&0&0&0&0&1&1&0&0&0&1\\ 0&1&0&0&0&0&0&1&0&0&1&0&0&1&1&0&1&0&0&1&1&0&0&0\\ 1&0&1&0&0&0&0&0&0&0&0&1&0&0&1&1&0&1&0&0&1&1&0&0\\ 0&1&0&1&0&0&0&0&1&0&0&0&1&0&0&1&0&0&1&0&0&1&1&0 \end{bmatrix}$$

$$B^{'} = \begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&1&1 \\ 0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&1&1&0&0&1&0&0&1 \\ 0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&1&1&1&0&0&1&0&0 \\ 1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&1&0 \\ 0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&1 \\ 0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&1&1&1&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&1&0&0&1&1&1&0 \\ 1&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&1&1&1 \end{bmatrix}$$

where $T_0 = x_0 + x_2$, $T_1 = x_1 + x_3$. The X transformation can be expressed as follows

$$\begin{aligned} y_0 &= L_2 \cdot x_0 + L_1 \cdot x_2 + L_5 \cdot x_2 + L_0 \cdot T_0 + L_3 \cdot T_0 + L_6 \cdot T_0 + L_7 \cdot T_0 \\ y_1 &= L_2 \cdot x_1 + L_1 \cdot x_3 + L_5 \cdot x_3 + L_0 \cdot T_1 + L_3 \cdot T_1 + L_6 \cdot T_1 + L_7 \cdot T_1 \\ y_2 &= L_0 \cdot x_0 + L_1 \cdot x_0 + L_5 \cdot x_0 + L_2 \cdot x_2 + L_4 \cdot x_2 + L_3 \cdot T_0 + L_6 \cdot T_0 + L_7 \cdot T_0 \\ y_3 &= L_0 \cdot x_1 + L_1 \cdot x_1 + L_5 \cdot x_1 + L_2 \cdot x_3 + L_4 \cdot x_3 + L_3 \cdot T_1 + L_6 \cdot T_1 + L_7 \cdot T_1 \end{aligned}$$

where L_i is a 32×32 left cyclic shift matrix, e.g.

$$L_2 = \begin{bmatrix} 0\,0\,1\,0\,0 \ldots 0\,0\,0\,0 \\ 0\,0\,0\,1\,0 \ldots 0\,0\,0\,0 \\ 0\,0\,0\,0\,1 \ldots 0\,0\,0\,0 \\ \vdots\,\vdots\,\vdots\,\vdots\,\vdots \quad \vdots \quad \vdots\,\vdots \\ 0\,1\,0\,0\,0 \ldots 0\,0\,0\,0 \end{bmatrix}.$$

It is easy to observe that any two rows of A' and B' have at most one common column with 1 in the first n columns. Therefore, the representation of X is transformed into the new form which not contains non-independent division property propagations.

Let the input division property of X be $(a_3, a_2, a_1, a_0, U_1, U_0) \in (\mathbb{F}_2^{32})^6$ and the output division property be $(b_3, b_2, b_1, b_0) \in (\mathbb{F}_2^{32})^4$. We can easily obtain the linear inequality constraints in terms of new expressions. Firstly, the following constraints corresponding to the substitution variables need to be added to the MILP model $\mathcal{M}$:

$$\begin{cases} U_0 = a_0 + a_2, \\ U_1 = a_1 + a_3, \\ U_1, U_0 \text{ } \textit{are binaries.} \end{cases}$$

Then, according to the linear matrices $A^{'}$ and $B^{'}$ with independent division property propagations, we can easily write the corresponding constraints. For example, for y_0, y_2, the first column of $A^{'}$ and $B^{'}$ corresponds to x_2^7, then we have the COPY constraint

$$\begin{cases} a_2^7 = A_0^7 + A_1^7 + \cdots + A_5^7, \\ A_0^7, \cdots, A_5^7 \text{ } \textit{are binaries.} \end{cases}$$

The last column of $A^{'}$ and $B^{'}$ corresponds to T_0^7, then we have the COPY constraint

$$\begin{cases} U_0^7 = u_0^0 + u_1^0 + \cdots + u_6^0, \\ u_0^0, \cdots, u_6^0 \textit{ are binaries.} \end{cases}$$

The first row of $B^{'}$ corresponds to y_0^7, then we have the XOR constraint

$$\begin{cases} b_0^7 = A_0^6 + A_0^2 + \cdots + t_1^0 + t_0^0, \\ b_0^7 \textit{ is binary.} \end{cases}$$

For the nonlinear layer of uBlock-128, we need to use the SAGE tool to convert division trails of the S-box into some linear inequalities, and then reduce them through the Greedy Algorithm (refer to [4] for more details). Algorithm 3 describes the whole process of building a MILP model for the BDP of uBlock-128. Based on Algorithm 3, for uBlock-128, we verified the 7-round integral distinguisher in the design document and 6-, 7-, 8-round integral distinguishers in [17], and obtained better results shown in the following.

8-Round Integral Distinguishers. When the least significant 4 bits of the input are constant and other positions are active, the output after 8 rounds has 96 balanced bits; when the most significant 4 bits of the input are constant and other positions are active, the output after 8 rounds has 96 balanced bits; when the least significant 1 bits of the input are constant and other positions are active, the output after 8 rounds are all balanced.

$$\begin{bmatrix} \mathcal{C} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \end{bmatrix} \xrightarrow{8R} \begin{bmatrix} bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \\ bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \\ bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \\ bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \end{bmatrix}$$

$$\begin{bmatrix} \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{C} \end{bmatrix} \xrightarrow{8R} \begin{bmatrix} bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \\ bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \\ bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \\ bubb & bubb & bubb & bubb & bubb & bubb & bubb & bubb \end{bmatrix}$$

$$\begin{bmatrix} \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} \\ \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \mathcal{A} & \text{aaac} \end{bmatrix} \xrightarrow{8R} \begin{bmatrix} \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \\ \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} & \mathcal{B} \end{bmatrix}$$

Algorithm 3 A MILP model for BDP propagation of uBlock-128

```
Input: S-box, the linear layer
Output: A MILP model $\mathcal{M}$ for BDP propagation of uBlock-128.
1: Build an empty MILP model $\mathcal{M}$.
2: $\mathcal{M}.var \leftarrow a_i, b_i, U_i, u_i, A_i$ //$a_i, b_i$ denote the input and output BDP of linear
   layer, the rest are newly introduced binary variables
3: //Generating Constrained Inequalities for S-boxes
4: Call Algorithm 2 in [4] to calculate the division trail of S-box: $V = \{a_i \to b_i\}$
5: Use $inequality\_generator()$ in SAGE to generate inequalities of $V$ : $\mathcal{L}(a_i, b_i)$
6: Reduce $\mathcal{L}(a_i, b_i)$ to $\mathcal{L}'(a_i, b_i)$ by using the Greedy Algorithm
7: $\mathcal{M}.con \leftarrow \mathcal{L}^{'}(a_i, b_i)$
8: //Generating Constraned Inequalities for Linear Layers
9: Write the implementation matrix $M$ of linear layer
10: Call Algorithm 2 to generate constraints: $\mathcal{L}^{''}(b_i, a_i, U_i, u_i, A_i)$
11: $\mathcal{M}.con \leftarrow \mathcal{L}^{''}(b_i, a_i, U_i, u_i, A_i)$
12: for $i$ in range$(0, n)$ do
13:     $\mathcal{M}.con \leftarrow \mathcal{L}'(a_i, b_i)$
14:     $\mathcal{M}.con \leftarrow \mathcal{L}^{''}(b_i, a_i, U_i, u_i, A_i)$
15:     $\mathcal{M}.con \leftarrow \mathcal{L}^{'''}(PL_{128}(b_i, a_{i+1}), PR_{128}(b_i, a_{i+1}))$
16: end for
17: Return $\mathcal{M}$
```

4.2 Application to MIBS

MIBS. MIBS is a lightweight block cipher proposed by Izadi M et al. at CANS 2009 [19]. Its overall encryption structure uses the Feistel network, and the round function adopts the SP network. The cipher has a 64-bit block length and supports 64-bit and 80-bit two key lengths. The number of iterative rounds is 32. All iterative operations in MIBS are based on nibble. The round function includes a XOR subkey, a S-box layer and a linear layer M denoted as $M = L \circ XOR \circ P$, where L represents the left and right permutations, XOR represents XOR with the input of the right half, P represents the linear transformation in the middle. The round function of MIBS is shown in Fig. 2.

MIBS has good resistance to differential cryptanalysis, linear cryptanalysis and integral cryptanalysis. The current best cryptanalytic result for MIBS is the 18-round linear cryptanalysis on MIBS-80 with the time complexity $2^{76.13}$, but the success probability is only 72.14% [20]. The existing integral attacks on it are mainly obtained by the derivation of the structure, and the known optimal integral distinguisher is a 5-round one proposed by Li et al. [21].

Searching the Integral Distinguisher of MIBS. Let the input of transformation P be $x = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$ and the output be $y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0)$. The division property propagation is expressed as $(a_7, a_6, a_5,$

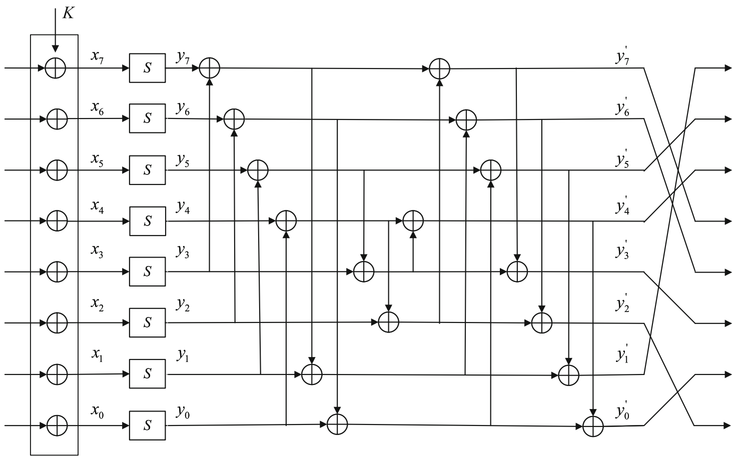

Fig. 2. Round function of MIBS.

$a_4, a_3, a_2, a_1, a_0) \xrightarrow{P} (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$. The transformation matrix of P is expressed as follows:

$$M = \begin{bmatrix} 1&1&1&0&1&1&0&1 \\ 0&1&1&0&0&1&1&1 \\ 0&0&1&1&1&0&1&1 \\ 1&0&0&1&1&1&0&1 \\ 1&1&0&0&1&1&1&0 \\ 1&0&1&1&0&1&1&1 \\ 0&1&1&1&1&1&1&0 \\ 1&1&0&1&1&0&1&1 \end{bmatrix}.$$

To convert the non-independent division property propagations involved in transformation P into independent propagations, the following new variables are introduced in using Algorithm 2.

$$\begin{cases} T_0 = x_0 + x_3 + x_4 + x_7, \\ T_1 = x_1 + x_2 + x_3 + x_6, \\ T_2 = x_0 + x_1 + x_4 + x_5, \\ T_3 = x_0 + x_2 + x_5 + x_6. \end{cases}$$

Transform the matrix into

$$M' = \begin{bmatrix} 1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0 \end{bmatrix}.$$

Take $x = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0, T_3, T_2, T_1, T_0)$ as new variables, and $y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0)$ is the output. The division property propagation is expressed as $(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0, U_3, U_2, U_1, U_0) \xrightarrow{P} (b_7, b_6, b_5,\ b_4, b_3,$ $b_2,\ b_1,\ b_0)$. Create model constraints according to the new expression. The new variable constraints that need to be added are

$$\begin{cases} U_0 = a_0 + a_3 + a_4 + a_7, \\ U_1 = a_1 + a_2 + a_3 + a_6, \\ U_2 = a_0 + a_1 + a_4 + a_5, \\ U_3 = a_0 + a_2 + a_5 + a_6, \\ U_3, \cdots, U_0 \textit{ are binaries.} \end{cases}$$

Using Algorithm 4, we can find an integral distinguisher up to 9 rounds with 63 active bits and 32 balanced bits. For the 8-round MIBS, the output of ciphertext are all balanced if plaintexts are chosen as 63 active bits and one constant bit. The following shows a list of integral distinguishers found by Algorithm 4. In particular, the 6-, 7-, 8- and 9-round integral distinguisher of MIBS all have longer rounds than the currently known ones.

$$\begin{bmatrix} \mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C}, \\ \mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{A},\mathcal{A},\mathcal{A} \end{bmatrix} \xrightarrow{5R} \begin{bmatrix} \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B}, \\ \mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U} \end{bmatrix}$$

$$\begin{bmatrix} \mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C},\mathcal{C}, \\ \mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A} \end{bmatrix} \xrightarrow{6R} \begin{bmatrix} \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B}, \\ \mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U} \end{bmatrix}$$

$$\begin{bmatrix} \mathcal{C},\mathcal{C},\mathcal{C},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A}, \\ \mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A} \end{bmatrix} \xrightarrow{7R} \begin{bmatrix} \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B}, \\ \mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U} \end{bmatrix}$$

$$\begin{bmatrix} \text{ccca},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A}, \\ \mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A} \end{bmatrix} \xrightarrow{8R} \begin{bmatrix} \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B}, \\ \mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U} \end{bmatrix}$$

$$\begin{bmatrix} \text{caaa},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A}, \\ \mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A} \end{bmatrix} \xrightarrow{8R} \begin{bmatrix} \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B}, \\ \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B} \end{bmatrix}$$

$$\begin{bmatrix} \text{caaa},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A}, \\ \mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A},\mathcal{A} \end{bmatrix} \xrightarrow{9R} \begin{bmatrix} \mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B}, \\ \mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U} \end{bmatrix}$$

Algorithm 4. A MILP model for BDP propagation of MIBS

```
Input: S-box, the linear layer
Output: A MILP model $\mathcal{M}$ for BDP propagation of MIBS
 1: Build an empty MILP model $\mathcal{M}$.
 2: $\mathcal{M}.var \leftarrow a_i, d_i, b_i, c_i, U_i, u_i, A_i$ //$a_i, b_i$ denote the input and output BDP of
    linear layer, the rest are newly introduced binary variables
 3: //COPY Operation in the Left of Feistel Structure
 4: $\mathcal{M}.con \leftarrow \mathcal{L}(a_i = d_i + a_{i+1})$
 5: // Generating Constrained Inequalities for S-boxes
 6: Use Algorithm 2 [4] to calculate the division trail of S-box: $V = \{d_i \rightarrow b_i\}$
 7: Use $inequality\_generator()$ in SAGE to generate inequalities of $V$ : $\mathcal{L}(d_i, b_i)$
 8: Reduce $\mathcal{L}(d_i, b_i)$ to $\mathcal{L}'(d_i, b_i)$ by using the Greedy Algorithm
 9: $\mathcal{M}.con \leftarrow \mathcal{L}^{'}(d_i, b_i)$
10: //Generating Constraned Inequalities for transformation $P$
11: Write the implementation matrix $M$ of $P$
12: Call Algorithm 2 to generate constraints: $\mathcal{L}^{''}(b_i, c_i, U_i, u_i, A_i)$
13: $\mathcal{M}.con \leftarrow \mathcal{L}^{''}(b_i, c_i, U_i, u_i, A_i)$
14: // XOR Operation in the Right of Feistel Network
15: $\mathcal{M}.con \leftarrow \mathcal{L}^{'''}(a_i + c_i = a_{i+1})$
16: for $i$ in range$(0, n)$ do
17:     $\mathcal{M}.con \leftarrow \mathcal{L}(a_i = d_i + a_{i+1})$
18:     $\mathcal{M}.con \leftarrow \mathcal{L}'(d_i, b_i)$
19:     $\mathcal{M}.con \leftarrow \mathcal{L}^{''}(b_i, c_i, U_i, u_i, A_i)$
20:     $\mathcal{M}.con \leftarrow \mathcal{L}^{'''}(a_i + c_i = a_{i+1})$
21: end for
22: Return $\mathcal{M}$
```

5 Conclusion

In this paper, we proposed a method to improve the accuracy of modeling the BDP propagation of complex linear layers using MILP model, which can also make models using SAT, and showed the effectiveness of this approach by applying it to two block ciphers uBlock-128 and MIBS. For uBlock-128, we found an integral distinguisher with the same round number as the longest known one, but our results has more balance bits. For MIBS, our method can attack more rounds than previous generic integral attacks. However, we cannot guarantee that the number of active bits is a tight lower bound for the 6-, 7- and 8-round integral distinguisher of MIBS due to the problem of solving time. Therefore, continuing to optimize the entire MILP model and improve the solving efficiency, and applying to other block ciphers with the complex linear layer are issues of our further investigations.

Acknowledgement. The authors would like to thank Prof. Guomin Yang and the anonymous reviewers for their detailed and very helpful comments and suggestions to improve this article. This work is supported by the National Natural Science Foundation of China (No. 62072445).

A Linear Inequalities for S-Boxes in uBlock-128

The following inequalities are the 12 inequalities used to describe uBlock S-box in MILP model of BDP, and $(a_3, a_2, a_1, a_0) \rightarrow (b_3, b_2, b_1, b_0)$ denotes a division trail of S-box.

$$\begin{cases} a_3 + a_2 + a_1 + a_0 - b_3 - b_2 - b_1 - b_0 \geq 0 \\ -3a_3 - a_2 - 2a_1 - 4a_0 + 3b_3 + b_2 + 2b_1 - b_0 \geq -5 \\ 2a_3 - a_0 - 2b_3 - b_2 - b_1 + b_0 \geq -2 \\ -4a_3 - 3a_2 - 2a_1 - 2a_0 - b_3 + 3b_2 + b_1 + 2b_0 \geq -6 \\ -a_1 + 2a_0 - b_3 - b_2 + b_1 - 2b_0 \geq -2 \\ -a_3 - a_2 - 2a_0 + b_3 + 2b_2 + 3b_1 + 2b_0 \geq 0 \\ a_3 + a_0 + b_3 - 2b_2 - 2b_1 - b_0 \geq -2 \\ a_1 + 2a_0 - b_3 - b_2 - b_1 - b_0 \geq -1 \\ a_3 - b_1 - b_0 \geq -1 \\ -a_3 - a_1 + b_3 + 2b_2 + b_1 + b_0 \geq 0 \\ a_1 - b_3 - b_1 \geq -1 \\ a_2 - b_2 - b_1 \geq -1 \end{cases}$$

B Linear Inequalities for S-Boxes in MIBS

The following inequalities are the 12 inequalities used to describe MIBS S-box in MILP model of BDP, and $(d_3, d_2, d_1, d_0) \rightarrow (b_3, b_2, b_1, b_0)$ denotes a division trail of S-box.

$$\begin{cases} d_3 + d_2 + 4d_1 + d_0 - 2b_3 - 2b_2 - 2b_1 - 2b_0 \geq -1 \\ 3d_2 - b_3 - b_2 - b_1 - b_0 \geq -1 \\ -d_3 - 2d_2 - 2d_1 - d_0 - b_3 - 2b_2 + 4b_1 - b_0 \geq -6 \\ -d_3 - 2d_2 - 2d_1 - d_0 + 5b_3 + 4b_2 + 5b_1 + 5b_0 \geq 0 \\ -d_3 - d_2 - d_1 - b_3 + 3b_2 - 2b_1 - b_0 \geq -4 \\ -d_3 - d_0 - 2b_3 - b_2 - b_1 + 3b_0 \geq -3 \\ d_3 + b_3 - b_2 - b_1 - b_0 \geq -1 \\ -d_3 - d_2 - d_0 + b_3 + 2b_2 + 2b_1 + b_0 \geq -1 \\ -d_1 - b_3 - b_2 + 2b_1 - b_0 \geq -2 \end{cases}$$

References

1. 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
2. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052343
3. 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
4. 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
5. Wang, Q., Hao, Y., Todo, Y., Li, C., Isobe, T., Meier, W.: Improved division property based cube attacks exploiting algebraic properties of superpoly. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 275–305. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_10
6. Liu, M., Yang, J., Wang, W., Lin, D.: Correlation cube attacks: from weak-key distinguisher to key recovery. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 715–744. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_23
7. Wang, S., Hu, B., Guan, J., Zhang, K., Shi, T.: MILP-aided method of searching division property using three subsets and applications. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 398–427. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34618-8_14
8. Hao, Y., Leander, G., Meier, W., Todo, Y., Wang, Q.: Modeling for three-subset division property without unknown subset. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 466–495. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_17
9. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Lower bounds on the degree of block ciphers. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12491, pp. 537–566. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_18
10. Hu, K., Sun, S., Wang, M., Wang, Q.: An algebraic formulation of the division property: revisiting degree evaluations, cube attacks, and key-independent sums. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12491, pp. 446–476. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64837-4_15
11. Hebborn, P., Lambin, B., Leander, G., Todo, Y.: Strong and tight security guarantees against integral distinguishers. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13090, pp. 362–391. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92062-3_13
12. Sun, L., Wang, W., Wang, M.Q.: MILP-aided bit-based division property for primitives with non-bit-permutation linear layers. IET Inf. Secur. **14**, 12–20 (2020)
13. Zhang, W.Y., Rijmen, V.: Division cryptanalysis of block ciphers with a binary diffusion layer. IET Inf. Secur. **13**, 87–95 (2019)
14. Hu, K., Wang, Q.J., Wang, M.Q.: Finding bit-based division property for ciphers with complex linear layers. IACR Trans. Symmetric Cryptol. **2020**, 396–424 (2020)

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. Wu, W.L., Zhang, L., Zheng, Y.F., Li, L.C.: The block cipher uBlock. J. Cryptol. Res. **6**(6), 690–703 (2019). (in Chinese)
17. Tian, W., Hu, B.: Integral cryptanalysis on two block ciphers Pyjamask and uBlock. IET Inf. Secur. **14**, 572–579 (2020)
18. Hong, C., Zhang, S., Chen, S., Lin, D., Xiang, Z.: More accurate division property propagations based on optimized implementations of linear layers. In: Yu, Yu., Yung, M. (eds.) Inscrypt 2021. LNCS, vol. 13007, pp. 212–232. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88323-2_11
19. Izadi, M., Sadeghiyan, B., Sadeghian, S.S., Khanooki, H.A.: MIBS: a new lightweight block cipher. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 334–348. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_22
20. Bay, A., Nakahara, J., Vaudenay, S.: Cryptanalysis of reduced-round MIBS block cipher. In: Heng, S.-H., Wright, R.N., Goi, B.-M. (eds.) CANS 2010. LNCS, vol. 6467, pp. 1–19. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17619-7_1
21. Li, Y.J., Sun, Q., Ou, H.W., et al.: Improved integral attacks on MIBS-64 block cipher. J. Cryptol. Res. **8**(4), 669–679 (2021). (in Chinese)