

New Algorithm for Modeling S-box in MILP Based Differential and Division Trail Search

Yu Sasaki^(✉) and Yosuke Todo

NTT Secure Platform Laboratories,
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
{sasaki.yu,todo.yosuke}@lab.ntt.co.jp

Abstract. This paper studies an automated differential-trail search against block ciphers in which the problem of finding the optimal trail is converted to one of finding the optimal solution in a mixed-integer-linear programming (MILP). The most difficult part is representing differential properties of an S-box, known as differential distribution table (DDT), with a system of inequalities. Previous work builds the system by using a general-purpose mathematical tool, SAGE Math. However, the generated system for general-purpose contains a lot of redundant inequalities for the purpose of differential-trail search, thus inefficient. Hence, an auxiliary algorithm was introduced to minimize the number of inequalities by hoping that it minimizes the runtime to solve the MILP. This paper proposes a new algorithm to improve this auxiliary algorithm. The main advantage is that while the previous algorithm does not ensure the minimum number of inequalities, the proposed algorithm does ensure it. Moreover it enables the users to choose the number of inequalities in the system. In addition, this paper experimentally shows that the above folklore “minimizing the number of inequalities minimizes the runtime” is not always correct. The proposed algorithm can also be used in the MILP-based division-trail search, which evaluates the bit-based division property for integral attacks.

Keywords: Differential trail · Division trail · Automated search tool · S-box · Mixed integer linear programming · Greedy algorithm

1 Introduction

Symmetric-key primitives like block ciphers are one of the most fundamental parts of cryptography. A lot of new designs have been proposed continuously to investigate good designs achieving both of high security and efficiency.

Differential cryptanalysis developed by Biham and Shamir [1] is one of the most generic cryptanalytic approaches. It is now almost mandatory for designers to evaluate security against differential cryptanalysis. Resistance against differential cryptanalysis can increase by using non-linear operations during the computation. The S-box, predefined substitution table, is a typical design choice to introduce non-linearity. A popular approach for evaluating differential cryptanalysis against an S-box based design is as follows.

- Evaluate the maximum probability of differential propagation in a single S-box denoted by p_S^{max} . Namely for S-box $S : \{0,1\}^n \mapsto \{0,1\}^n$, p_S^{max} is defined as

$$p_S^{max} \triangleq \max_{\Delta_i, \Delta_o} \left\{ \frac{\#\{x \in \{0,1\}^n \mid S(x) \oplus S(x \oplus \Delta_i) = \Delta_o\}}{2^n} \right\}. \quad (1)$$

- Search for the differential propagation pattern for the entire algorithm that minimizes the number of S-boxes with difference, denoted by active S-boxes. This process is known as differential trail search. Let N_{AS} be the lower bound of the number of active S-boxes.
- The probability of a differential trail is upper-bounded by $N_{AS} \times P_S^{max}$.

The differential trail search is the most difficult part. Some designs, e.g. AES, adopt a clever computation structure such that the minimum number of active S-boxes can be calculated easily, which is called **wide trails strategy**. However, it cannot be applied to any algorithm especially for complicated computation structure.

In 2011, Mouha et al. proposed an automated differential path search method by using mixed-integer-linear programming (MILP) [2], which generates lower bounds of the number of active S-boxes. At that time, the search tool could not consider the differential property of the S-box, thus could not apply to bit-oriented ciphers such as PRESENT [3] and LS-designs [4].

This restriction was later solved by Siwei et al. [5,6], which describes the possible and impossible differential propagations of the S-box with a system of inequalities. The goal of our research is improving the S-box description of [5,6], thus we explain their method deeply.

S-box Modeling in Previous Work. Suppose that the S-box we want to analyze is the 4-bit S-box defined in Table 1. Its differential distribution table (DDT), i.e. the value of $\#\{x\}$ in Eq. (1) for each of (Δ_i, Δ_o) , is given in Table 2. Also suppose that x_3, x_2, x_1, x_0 are four binary variables in which $x_j = 0$ and $x_j = 1$ for $j \in \{0, 1, 2, 3\}$ represent that j -th input bit to the S-box is inactive and active, respectively. Similarly y_3, y_2, y_1, y_0 are four binary variables to represent the active status of the S-box output. Here the goal is building a system of linear inequalities with respect to $(x_3, x_2, x_1, x_0, y_3, y_2, y_1, y_0)$ so that the solution space matches the non-zero entries in DDT. From Table 2, input difference $0x1$ never propagates to output difference $0x2$. Thus $(x_3, x_2, x_1, x_0, y_3, y_2, y_1, y_0) = (00010010)$ must be removed from the search range. On the other hand $\Delta_i = 0x1$ can propagate to $\Delta_o = 0x1$ with non-zero probability. Thus $(x_3, x_2, x_1, x_0, y_3, y_2, y_1, y_0) = (00010001)$ must be included in the solution space. Similarly, from $2^8 = 256$ patterns of $(x_3, x_2, x_1, x_0, y_3, y_2, y_1, y_0)$, we now have a set of patterns that must be excluded from the solution space (150 entries with 0 in Table 2).

The above can be generalized to the following problem; For a given subspace $\mathcal{R} \subset \mathbb{F}_2^n$, we want to build a system of inequalities so that $\mathbb{F}_2^n - \mathcal{R}$ matches a solution space. Two approaches are known to build such a system of inequalities.

SAGE Math: Due to this highly generic problem, a mathematical tool, *SAGE Math*, equips the function to generate such a system of inequalities. Namely it takes as input a subset of \mathbb{F}_2^n , and returns a system of inequalities with the form of $\alpha_0^i x_0 + \alpha_1^i x_1 + \cdots + \alpha_{n-1}^i x_{n-1} + \alpha_n^i \geq 0$, where α_j^i is an integer coefficient for the i th inequality. Here, SAGE Math assumes an *integer programming*, namely it regards that each of x_0, x_1, \dots, x_{n-1} is an integer variable. However, the differential-trail search is *01-integer programming*, namely each variable takes only 0 or 1. As a result, the system generated by SAGE Math is too heavy and contains a lot of overlap when the value of each variable is limited to 0 or 1. Hence, it requires an auxiliary code to exclude the overlapping inequalities.

Logical Computation Model: It is also possible to directly build a system of inequalities for 01-integer programming without using the existing mathematical tool. Suppose that we want to remove $(x_3, x_2, x_1, x_0, y_3, y_2, y_1, y_0) = (00010010)$ from the solution space. Then, we can specify $x_3 + x_2 + x_1 - x_0 + y_3 + y_2 - y_1 + y_0 \geq -1$, which removes this pattern while keeps all the other patterns in the solution space. Moreover, suppose that (00010011) should also be removed. In this case, two patterns (00010010) and (00010011) can be removed together, *i.e.* $(0001001*)$ can be removed by $x_3 + x_2 + x_1 - x_0 + y_3 + y_2 - y_1 \geq -1$. Such a merged form yields the overlap issue in the logical computation model as well as SAGE Math. Namely, if $(0001001*)$ is removed, we no longer need to remove (00010010) and (00010011) one by one.

Table 1. An example of S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	4	8	7	1	9	3	2	E	0	B	6	F	A	5	D	C

In both approaches, a straightforward system of inequalities to remove \mathcal{R} contains too many overlaps, e.g. SAGE Math generates more than 300 inequalities for 4-bit S-box while only 20 to 30 inequalities are sufficient to strictly define $\mathbb{F}_2^4 - \mathcal{R}$. Previous work [5, 6] argued that *minimizing the number of inequalities* is important to minimize the runtime to solve MILP and proposed using the *greedy algorithm* to exclude redundant inequalities. More precisely, the greedy algorithm picks an inequality which excludes more elements in \mathcal{R} from the solution space than any other inequalities. Then the same procedure continues for the remaining inequalities and elements in \mathcal{R} until all the elements in \mathcal{R} are excluded from the solution space. For the sake of simplicity, we call the algorithm to minimize the number of inequalities *reduction algorithm*.

The use of the greedy algorithm has two drawbacks. First, it does not guarantee the optimal choice. Second, there are many inequalities having a tied score during the execution of the greedy algorithm but the authors [5, 6] did not specify the choice of this case, thus their algorithm does not have reproducibility.

Table 2. Differential Distribution Table (DDT)

		Δ_i															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	0	0	0	0	2	0	0	2	2	2	4	0	0	2	
2	0	0	0	2	2	0	2	2	0	4	0	2	0	2	0	0	
3	0	2	0	0	0	2	2	2	2	0	0	0	0	2	0	4	
4	0	0	0	2	0	2	0	0	0	0	2	4	0	2	2	2	
5	0	4	2	2	0	2	0	2	0	2	2	0	0	0	0	0	
6	0	0	2	0	0	0	4	2	0	0	2	0	2	2	2	0	
7	0	0	0	2	2	2	2	0	2	0	4	0	2	0	0	0	
8	0	2	2	4	2	0	2	0	0	0	0	0	0	0	0	2	
9	0	0	0	2	0	0	0	2	4	2	0	0	2	0	2	2	
a	0	0	2	0	2	0	0	4	2	0	2	2	0	0	0	2	
b	0	2	0	0	2	2	0	2	0	0	0	2	2	0	4	0	
c	0	2	0	0	2	0	0	0	2	2	2	0	0	4	2	0	
d	0	2	4	2	0	0	0	0	2	0	0	2	2	2	0	0	
e	0	0	2	0	4	2	0	0	0	2	0	0	2	2	0	2	
f	0	0	2	0	0	4	2	0	2	2	0	2	0	0	2	0	

Indeed, we could not reproduce the same result as [5, 6] and this is a part of our motivation to develop a new algorithm.

Our Contributions. The current paper proposes a new reduction algorithm when the S-box is modeled with MILP. More precisely, we first use either SAGE Math or logical computation model to obtain a large system of inequalities including redundant ones. Then as the reduction algorithm, we use our new algorithm instead of the greedy algorithm in previous work. Interestingly, the new reduction algorithm is based on MILP. Namely, we convert the problem of minimizing the number of inequalities for excluding \mathcal{R} to the problem of minimizing the sum of involved inequalities in some MILP problem.

The new algorithm inherits the advantage of the MILP such that the optimal solution is obtained. Previous work [6] listed the number of inequalities which they obtained by applying the greedy algorithm to 4-bit S-boxes in various ciphers; Kline [7], Piccolo [8], TWINE [9], PRINCE [10], MIBS [11], PRESENT [3], LED [12], LBlock [13], Serpent [14]. For comparison, we also apply the new reduction algorithm to those S-boxes. The results are shown in Table 3. The new reduction algorithm finds a smaller set of inequalities for all S-boxes but for TWINE. For completeness, we also apply our reduction algorithm to several other ciphers recently designed, which includes LILLIPUT [15], Midori [16], Minalpher [17], RECTANGLE [18], SKINNY [19].

With the new reduction algorithm, the user can choose the number of equalities which will be incorporated into the system. This property ensures the reproducibility of the results. Namely, every user can obtain the system with the same number of equalities by simply running the existing MILP solver.

This feature enables us to perform experiments of solving the same problem with various numbers of inequalities to represent the S-box, which would reveal the relationship between the number of inequalities and the runtime for the entire differential-trail search problem. The results show that when the number

Table 3. Number of inequalities to exclude \mathcal{R} for various 4-bit S-boxes

Sbox	#inequalities				Sbox	#inequalities			
	SAGE	Math	Previous	Ours		SAGE	Math	Previous	Ours
Kline	311	22	21	21	LBlock S6	205	27	24	24
Piccolo	202	23	21	21	LBlock S7	205	27	24	24
TWINE	324	23	23	23	LBlock S8	205	28	24	24
PRINCE	300	26	22	22	LBlock S9	205	27	24	24
MIBS	378	27	23	23	Serpent S0	327	23	21	21
PRESENT/LED	327	22	21	21	Serpent S1	327	24	21	21
LBlock S0	205	28	24	24	Serpent S2	325	25	21	21
LBlock S1	205	27	24	24	Serpent S3	368	31	27	27
LBlock S2	205	27	24	24	Serpent S4	321	26	23	23
LBlock S3	205	27	24	24	Serpent S5	321	25	23	23
LBlock S4	205	28	24	24	Serpent S6	327	22	21	21
LBlock S5	205	27	24	24	Serpent S7	368	30	27	27
Lilliput	324	—	23	23	Minalpher	338	—	22	22
Midori S0	239	—	21	21	RECTANGLE	267	—	21	21
Midori S1	367	—	22	22	SKINNY	202	—	21	21

of inequalities for each S-box is too small, the runtime for the entire algorithm is significantly longer, which runs contrary to the previous belief that minimizing the number of inequalities for each S-box is the best.

Paper Outline. The remaining of this paper is structured as follows. Section 2 explains how to search for differential trails with MILP. Section 3 explains the previous reduction algorithm based on the greedy algorithm. Section 4 presents our new reduction algorithm based on MILP. Section 5 shows the experiments that minimizing the number of inequalities is not the best strategy. Section 6 discusses the application to the division-trail search.

2 Differential Trail Search with MILP

In this section, we explain how the problem of finding the best differential trail is converted to the problem of solving MILP.

Suppose that we build an MILP model for a block cipher whose block size is b bits and the number of rounds is r . Let s_0, s_1, \dots, s_{b-1} be b bits of the plaintext. Similarly, let $s_{b*j+0}, s_{b*j+1}, \dots, s_{b*j+b-1}$, $j = 1, 2, \dots, r$ be b bits of the state after round j , thus $s_{b*r+0}, s_{b*r+1}, \dots, s_{b*r+b-1}$ are b bits of the ciphertext.

To make an MILP model, binary variables, $x_i \in \{0, 1\}$ where $i = 0, 1, \dots, b*r + b - 1$, are firstly introduced to represent whether the bit s_i is active or not. $x_i = 0$ indicates that s_i is inactive while $x_i = 1$ indicates that s_i is active.

Secondly, the valid differential propagation is modeled by using inequalities among variables x_i . It is more convenient to discuss an example. Here, we consider the toy example in Fig. 1, in which the block size is 8 bits and round function consists of application of 4-bit S-box for the top 4 bits and the bottom 4 bits, xoring the top 4 bits to the bottom, and swap those 4 bits.

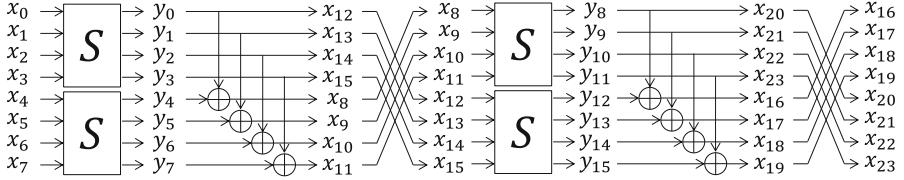


Fig. 1. Binary variables for 2-round toy cipher

Each input bit to the first round and the second round is modeled by x_0 - x_7 and x_8 - x_{15} , respectively. The last swap is the permutation of bit positions, thus the state before the swap in the first round can be described with x_8 - x_{15} . Due to the requirement of the complex method, new binary variables y_0 - y_7 are introduced to represent the active status of each output bit from S-box. Figure 1 includes those binary variables.

Model for Linear Part. Swap operation, or any other permutation of bit positions, does not require any inequalities. When we refer to top 4 bits before the swap operation in the first round, we can directly refer to $x_{12}, x_{13}, x_{14}, x_{15}$.

MILP accepts using equation to construct the model. Thus the relation $y_0 = x_{12}$, $y_1 = x_{13}$, $y_2 = x_{14}$, and $y_3 = x_{15}$ can be modeled directly.

Then we consider modeling valid differential propagation for 2-bit xor, i.e. $a \oplus b = c$. We can exclude impossible propagation patterns one by one with one inequality. The impossible patterns are $(a, b, c) = (1, 0, 0), (0, 1, 0), (0, 0, 1), (1, 1, 1)$.

- $(a, b, c) = (1, 0, 0)$ can be removed by $-a + b + c \geq 0$. Indeed, $(a, b, c) = (1, 0, 0)$ does not satisfy this inequality, and all the other patterns remain in the solution space.
- $(a, b, c) = (1, 0, 0)$ can be removed by $a - b + c \geq 0$.
- $(a, b, c) = (0, 1, 0)$ can be removed by $a + b - c \geq 0$.
- $(a, b, c) = (1, 1, 1)$ can be removed by $-a - b - c \geq -2$.

In the end, by using the above four inequalities, the valid differential propagation for $a \oplus b = c$ is modeled. By replacing (a, b, c) with (y_0, y_4, x_8) , (y_1, y_5, x_9) , (y_2, y_6, x_{10}) , and (y_3, y_7, x_{11}) , the xor operations of the first round can be modeled with $4 * 4 = 16$ inequalities.

Model for 4-Bit S-box. The model for S-box has already been explained in Sect. 1. In this paper, we focus on using SAGE Math. The user first analyzes DDT of the target S-box and makes a list of valid patterns of $(x_3x_2x_1x_0y_3y_2y_1y_0)$. The user then passes the list to SAGE Math to obtain a system of inequalities in which the solution space corresponds to the given patterns. The system looks as follows, which is a result of simulating DDT in Table 2.

Line 1: An inequality $(-1, 0, 0, 0, 0, 0, 0, 0)x + 1 \geq 0$

Line 2: An inequality $(0, -1, 0, 0, 0, 0, 0, 0)x + 1 \geq 0$

Line 3: An inequality $(0, 0, -1, 0, 0, 0, 0, 0)x + 1 \geq 0$

Algorithm 1. Pseudo-Algorithm for Reduction Algorithm 1 (Greedy Algorithm) in [6]

Require: \mathcal{H}, \mathcal{X}

Ensure: \mathcal{O} (a new list of inequalities)

- 1: Initialize \mathcal{O} to the empty list.
 - 2: **while** \mathcal{X} is not empty **do**
 - 3: Pick up an inequality in \mathcal{H} which maximizes the number of removed impossible patterns in \mathcal{X} .
 - 4: Add the inequality to \mathcal{O} .
 - 5: Erase the inequality from \mathcal{H} .
 - 6: Erase the removed impossible patterns from \mathcal{X} .
 - 7: **end while**
 - 8: **return** \mathcal{O}
-

Line 4: An inequality $(0, 0, 0, -1, 0, 0, 0, 0)x + 1 \geq 0$

Line 5: An inequality $(0, 0, 0, 0, -1, 0, 0, 0)x + 1 \geq 0$

Line 6: An inequality $(0, -1, -1, -1, -1, -1, 0, -1)x + 5 \geq 0$

Line 7: An inequality $(-1, 0, -1, -1, -1, -1, 0, -1)x + 5 \geq 0$

⋮

Line 323: An inequality $(-1, -1, -1, 0, -1, 1, -1, 1)x + 4 \geq 0$

Line 324: An inequality $(-1, -2, -1, -3, -1, -3, -2, -3)x + 13 \geq 0$

Each line indicates 8 coefficients for each variable and the constant value. For example, the last line denotes $-x_3 - 2x_2 - x_1 - 3x_0 - y_3 - 3y_2 - 2y_1 - 3y_0 + 13 \geq 0$. Thus each S-box can be modeled with 324 inequalities, and all the S-layers of the r -round toy cipher can be modeled with $324 \times 2 \times r$ inequalities.

A natural concern is that using 324 inequalities per S-box is too costly. Indeed, the system generated by SAGE Math assumes that each variable can take any integer, while they take only 0 or 1 in differential search. Thus, a lot of inequalities are actually unnecessary. To reduce the number of inequalities, Siwei et al. [5, 6] introduced the greedy algorithm as follows.

Let \mathcal{H} be a list of inequalities generated by SAGE Math. Let \mathcal{X} be a list of impossible differential propagation patterns to be removed, which initialized to \mathcal{R} . Their algorithm generates a new list of inequalities, \mathcal{O} , which is much smaller than \mathcal{H} . Intuitively, \mathcal{O} is first initialized to the empty set, and they add inequality of \mathcal{H} to \mathcal{O} one by one so that at each timing the number of removed impossible patterns of \mathcal{X} is maximized. The pseudo-algorithm is given in Algorithm 1.

In this paper, we call the algorithm to reduce the number of inequalities *reduction algorithm*, which is the main object of this paper.

Solving the System. After the system of inequalities is generated, we need to find an optimal solution. This part is generally done by using existing software to solve MILP. A number of MILP solvers are available such as Gurobi Optimizer [20], SCIP [21] and CPLEX Optimization Studio [22]. Some of them are commercial products but many of them offer a free license for academic organizations.

3 Problems of Reduction Algorithm in Previous Work

In general, the greedy algorithm does not guarantee the optimality of the solution, and this actually applies to the problem of finding minimal representation to describe DDT. We provide a counterexample to demonstrate this fact. Let \mathcal{R} be a set of 8 elements in \mathbb{F}_2^4 such that $\mathcal{R} \triangleq \{0001, 0101, 0111, 0110, 1100, 1101, 1111, 1011\}$, and here the goal is finding a minimal representation of \mathcal{R} . Figure 2(a) represents \mathcal{R} in the Karnaugh map.

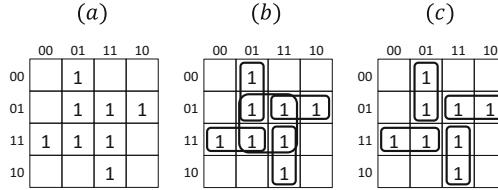


Fig. 2. An example that greedy algorithm does not minimize constraints

In Fig. 2(b), the greedy algorithm is applied. Firstly, four elements in the center ($0101, 0111, 1101, 1111$) are simplified to $*1*1$, and then four elements remain. As a result, \mathcal{R} is represented by five elements $\{*1*1, 0*01, 110*, 011*, 1*11\}$. However, it is easy to see that the last four elements are sufficient to cover \mathcal{R} (Fig. 2(c)), which shows that the greedy algorithm is not optimal.

4 New Reduction Algorithm

In this section, we describe our new reduction algorithm to search for the representation of an n -bit S-box with the minimum number of inequalities. We assume that the following analysis finished before we run the reduction algorithm.

- DDT (with 2^{2n} entries) is computed. Let $x_{n-1}, x_{n-2}, \dots, x_0$ and $y_{n-1}, y_{n-2}, \dots, y_0$ be n binary variables to denote whether each of input and output bit is active or not, respectively. Then the attacker obtains a set of all impossible differential propagation patterns of $(x_{n-1}, x_{n-2}, \dots, x_0, y_{n-1}, y_{n-2}, \dots, y_0)$. This set is denoted by \mathcal{R} . Suppose that there are $|\mathcal{R}|$ elements in \mathcal{R} . We denote each element of \mathcal{R} by $R_0, R_1, R_2, \dots, R_{|\mathcal{R}|-1}$.
- A large size of the system of inequalities to represent $\mathbb{F}_2^{2n} - \mathcal{R}$ is obtained by using either SAGE Math or the logical computation model. We denote the number of inequalities before the reduction algorithm is applied by N .

In the end, we know $|\mathcal{R}|$ patterns that must be excluded from the solution space and we know N inequalities whose intersection achieves $\mathbb{F}_2^{2n} - \mathcal{R}$ but containing many redundant inequalities.

Overview. The overview of our reduction algorithm is as follows. First, for each impossible pattern R_i , we compute which inequalities can exclude R_i from the solution space. Second, for each R_i we make a constraint such that R_i must be excluded from the solution space by at least 1 inequality. Finally, under these constraints we minimize the number of inequalities.

Initial Process. We first perform a small pre-process. For each R_i , we check which of N inequalities exclude the pattern from the solution space. Let $\bar{\mathcal{R}}_i$ be a set of inequalities that can exclude R_i . For example, we consider the situation summarized in Table 4, which indicates as follows.

- R_0 can be excluded with inequalities 2, 8, N . Namely, $\bar{\mathcal{R}}_0 = \{2, 8, N\}$.
 - R_1 can be excluded with inequalities 2, 3, 7. Namely, $\bar{\mathcal{R}}_1 = \{2, 3, 7\}$.
 - ⋮
 - $R_{|\mathcal{R}|-1}$ can be excluded with inequalities 1, 3, 4, 9. Namely, $\bar{\mathcal{R}}_{|\mathcal{R}|-1} = \{1, 3, 4, 9\}$.

Table 4. Example of precomputation analysis

	Patterns in \mathcal{R}												
	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	\dots	$R_{ \mathcal{R} -1}$	
Inequality 1	0	0	0	1	1	0	1	0	0	0	\dots	1	
Inequality 2	1	1	0	1	0	0	0	0	0	1	\dots	0	
Inequality 3	0	1	1	0	0	1	0	0	1	0	\dots	1	
Inequality 4	0	0	0	0	0	0	1	0	0	0	\dots	1	
Inequality 5	0	0	0	1	0	0	0	0	1	0	\dots	0	
Inequality 6	0	0	1	0	0	0	0	0	0	0	\dots	0	
Inequality 7	0	1	0	0	1	1	0	0	0	0	\dots	0	
Inequality 8	1	0	0	1	0	0	0	1	0	0	\dots	0	
Inequality 9	0	0	1	0	0	0	0	1	0	0	\dots	1	
\vdots							\vdots						
Inequality N	1	0	0	1	0	0	1	0	1	1	\dots	0	

Declaration of Variables. In this MILP, we use N binary variables z_1, z_2, \dots, z_N , in which $z_i = 1$ denotes that inequality i is included in the system and $z_i = 0$ denotes that inequality i will not be used in the system.

Objective Function. The goal is minimizing the number of inequalities adopted in the system, which can be represented by

$$\text{minimize} \quad \sum_{i=1}^N z_i. \quad (2)$$

Constraints. The only constraint we need is ensuring that each impossible pattern is removed with at least one inequality. Thus we have $|\mathcal{R}|$ constraints in

which the sum of z_i with $i \in \bar{\mathcal{R}}_i$ is greater than or equal to 1. With the above example, we have the following constraints.

$$\begin{aligned} z_2 + z_8 + z_N &\geq 1, & \text{as the constraint for } R_0, \\ z_2 + z_3 + z_7 &\geq 1, & \text{as the constraint for } R_1, \\ &\vdots & \vdots \\ z_1 + z_3 + z_4 + z_9 &\geq 1, & \text{as the constraint for } R_{|\mathcal{R}|-1}. \end{aligned}$$

Applications. We applied the new reduction algorithm to a system of inequalities generated with SAGE Math against various S-boxes. The results are shown in Table 3. Compared to the previous reduction algorithm based on the greedy algorithm, a smaller number of inequalities can be achieved for most of the applications. As a proof of context, the system of 21 inequalities to describe the DDT of PRESENT and LED is listed below.

$$\begin{aligned} -1x3 - 1x2 + 0x1 - 1x0 - 1y3 + 0y2 + 1y1 + 0y0 + 3 &\geq 0, \\ -1x3 + 0x2 - 1x1 - 1x0 + 1y3 + 0y2 - 1y1 + 0y0 + 3 &\geq 0, \\ 0x3 - 2x2 - 2x1 - 2x0 - 1y3 + 2y2 - 1y1 - 1y0 + 7 &\geq 0, \\ -3x3 + 2x2 - 2x1 - 1x0 + 1y3 - 2y2 - 2y1 - 1y0 + 8 &\geq 0, \\ 1x3 - 1x2 + 1x1 + 2x0 - 2y3 - 2y2 + 1y1 - 2y0 + 5 &\geq 0, \\ 0x3 - 1x2 - 1x1 + 1x0 - 1y3 + 0y2 - 1y1 + 1y0 + 3 &\geq 0, \\ 2x3 + 3x2 - 2x1 - 4x0 - 4y3 - 4y2 - 1y1 + 1y0 + 11 &\geq 0, \\ 2x3 - 1x2 + 2x1 + 2x0 + 2y3 + 3y2 - 1y1 - 1y0 + 0 &\geq 0, \\ -2x3 + 1x2 + 1x1 + 3x0 + 1y3 - 1y2 + 1y1 + 2y0 + 0 &\geq 0, \\ -1x3 + 1x2 + 1x1 - 1x0 + 0y3 + 0y2 + 0y1 - 1y0 + 2 &\geq 0, \\ 0x3 + 2x2 - 2x1 + 1x0 - 1y3 - 1y2 - 2y1 - 2y0 + 6 &\geq 0, \\ 2x3 + 3x2 + 3x1 + 2x0 + 1y3 - 4y2 + 1y1 + 1y0 + 0 &\geq 0, \\ 1x3 + 2x2 + 2x1 + 0x0 - 1y3 + 1y2 - 1y1 + 1y0 + 0 &\geq 0, \\ 0x3 - 2x2 - 2x1 + 3x0 + 4y3 + 1y2 + 4y1 + 1y0 + 0 &\geq 0, \\ 2x3 + 2x2 - 1x1 + 2x0 - 1y3 + 3y2 + 2y1 - 1y0 + 0 &\geq 0, \\ 1x3 + 3x2 - 2x1 - 2x0 + 3y3 - 4y2 + 1y1 + 4y0 + 0 &\geq 0, \\ 1x3 - 3x2 - 2x1 - 2x0 + 3y3 - 4y2 + 1y1 - 3y0 + 10 &\geq 0, \\ -1x3 + 3x2 + 3x1 - 1x0 + 2y3 + 2y2 + 2y1 - 1y0 + 0 &\geq 0, \\ 2x3 - 2x2 + 3x1 - 4x0 - 1y3 - 4y2 - 4y1 + 1y0 + 11 &\geq 0, \\ 1x3 - 2x2 + 3x1 - 2x0 + 1y3 + 4y2 + 3y1 + 4y0 + 0 &\geq 0, \\ -2x3 - 1x2 - 1x1 + 2x0 - 2y3 + 0y2 - 2y1 - 1y0 + 7 &\geq 0. \end{aligned}$$

Extension. With the new reduction algorithm, the users can specify the number of inequalities generated in the system by adding the following changes.

1. Add a constraint $\sum_{i=1}^N z_i = N_t$ where N_t is the number of inequalities to be included.
2. Leave the objective function empty.

5 Experiments

The new algorithm in Sect. 4 enables us to choose the number of inequalities in the system. In this section, we run the experiments which solve the fixed MILP model by choosing various numbers of inequalities in order to check how the number of inequalities is related to the runtime of the entire MILP. Sasaki and Todo presented how to model differential-trail search for LILLIPUT [23] in details. We determined to adopt their model to minimize inaccuracy. Section 5.1 briefly explains the specification of LILLIPUT. Then in Sect. 5.2 we report the experimental results.

5.1 LILLIPUT Specification

Block cipher LILLIPUT [15] was designed by Berger et al. in 2015, and adopts 16-branch extended generalized Feistel network [24] with the block-shuffle mechanism [25]. The block size and the key size are 64 bits and 80 bits, respectively. The state consists of 16 branches of size 4 bits. 64-bit plaintext is first loaded to sixteen nibbles $X_{15}, X_{14} \dots, X_0$, and those are updated by the round function 30 times. The round function is illustrated in Fig. 3.

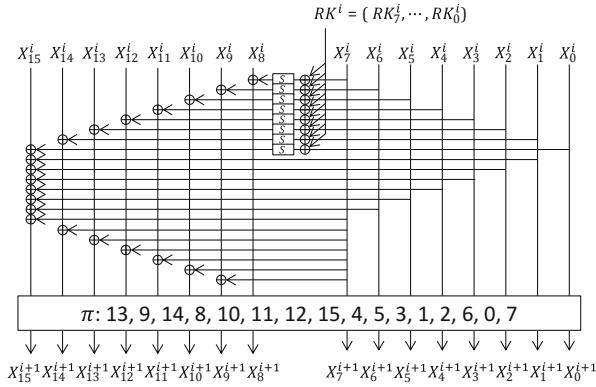


Fig. 3. Round function of LILLIPUT

At first, eight nibbles of round key are xored to each of eight nibbles in the right half of the state, and the results are xored to the left half of the state. Let RK_i^j and X_i^j be the i -th nibble of the j -th round key RK^j and j -th round state X^j , respectively. Then, the nonlinear layer can be defined as $X_{8+i}^j \leftarrow X_{8+i}^j \oplus S(X_{7-i}^j \oplus RK_i^j)$, $i = 0, 1, \dots, 7$, where $S(\cdot)$ is a 4-bit to 4-bit S-box defined in Table 1.

After the non-linear update, some linear update applying xor between branches is performed, which is defined as follows.

$$\begin{aligned} X_{15}^j &\leftarrow X_{15}^j \oplus X_7^j \oplus X_6^j \oplus X_5^j \oplus X_4^j \oplus X_3^j \oplus X_2^j \oplus X_1^j, \\ X_{15-i}^j &\leftarrow X_{15-i}^j \oplus X_7^j \text{ for } i = 1, 2, \dots, 6. \end{aligned}$$

Finally, nibble positions are permuted according to π defined as

$$\begin{aligned} (13, 9, 14, 8, 10, 11, 12, 15, 4, 5, 3, 1, 2, 6, 0, 7) \\ \leftarrow \pi(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15). \end{aligned}$$

Table 5. Runtime of differential-trail search for 5-round and 6-round LILLIPUT.

Time for 5 rounds								
#inequalities	23	33	43	53	63	73	83	93
Time(sec)	75.68	106.30	19.95	28.33	14.77	24.77	20.45	18.92
#inequalities	103	113	123	133	143	153	163	173
Time(sec)	26.15	18.70	15.97	20.45	29.08	54.72	20.48	21.53
#inequalities	183	193	203	213	223	233	243	253
Time(sec)	20.51	68.53	28.35	22.96	24.38	20.55	27.33	28.91
#inequalities	263	273	283	293	303	313	323	
Time(sec)	28.66	30.50	25.96	29.14	33.46	33.49	34.41	

Time for 6 rounds								
#inequalities	23	33	43	53	63	73	83	93
Time(sec)	14068.13	1125.83	1333.97	2941.26	1351.91	2640.21	2424.82	1092.05
#inequalities	103	113	123	133	143	153	163	173
Time(sec)	1205.98	1411.40	1495.90	1330.74	1395.17	1376.73	1538.08	1805.86
#inequalities	183	193	203	213	223	233	243	253
Time(sec)	1841.54	1272.60	1893.98	2402.03	3938.68	5401.42	3110.85	3060.52
#inequalities	263	273	283	293	303	313	323	
Time(sec)	2203.71	2060.90	3565.50	3443.10	3558.77	4784.93	5104.19	

5.2 Runtime of Differential Trail Search for LILLIPUT

We first generated the system of inequalities to model the DDT of LILLIPUT's S-box with SAGE Math, which returned 324 inequalities. We then ran the reduction algorithm described in Sect. 4, which revealed that all valid propagations can be described with 23 inequalities in minimum as summarized in Table 3.

The goal of this section is performing experiments to detect the relationship between the number of inequalities of the single DDT and the runtime of the entire differential-trail search problem. We tested 5-round differential-trail search and 6-round differential-trail search with $23 + 10i$ inequalities per S-box, where we change i within the range of $0 \leq i \leq 30$. The results are summarized in Table 5 and in Fig. 4.

The results clearly show that the runtime is significantly slow when the number of inequalities for each S-box is almost minimum. It is even slower than the case that no reduction is performed to the system of inequalities.

This phenomenon can be intuitively explained as follows. The MILP solver itself optimizes the system by considering the entire system instead of focusing on each S-box. Thus, leaving some room for the solver to optimize can result in the minimum runtime for the entire system. However, including too many inequalities also consumes time to optimize it. Thus, as the number of inequalities becomes significantly larger, the runtime gets gradually bigger.

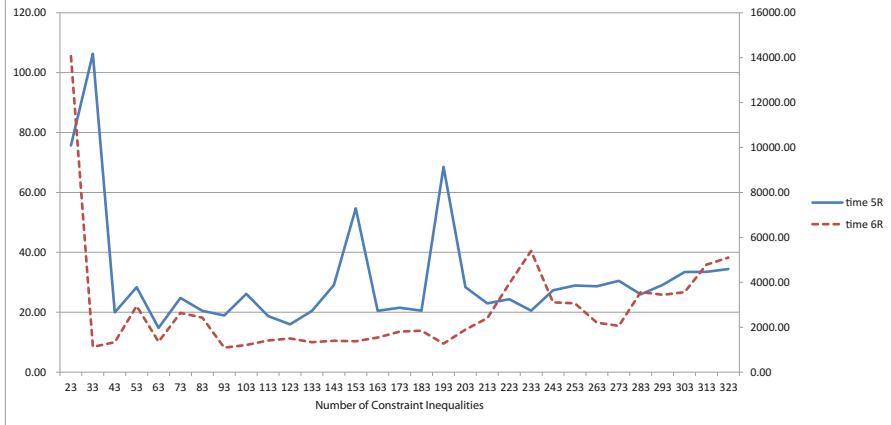


Fig. 4. Graphical representation of the experimental results. The left vertical axis is for 5 rounds and the right vertical axis is for 6 rounds.

6 Application to Division-Trail Search

Division property was invented by Todo [26] as a new tool to evaluate the property used in integral cryptanalysis in a more general form. The concept was later extended to bit-based division property by Todo and Morii [27], which needs to evaluate the S-box property with the division property defined in each bit.

To evaluate the S-box, it first obtains the algebraic normal form of the S-box, for example is described as

$$\begin{aligned}y_0 &= 1 + x_0 + x_1 + x_0x_1 + x_2 + x_3, \\y_1 &= x_0 + x_0x_1 + x_2 + x_0x_2 + x_3, \\y_2 &= x_1 + x_2 + x_0x_3 + x_1x_3 + x_1x_2x_3, \\y_3 &= x_0 + x_1x_3 + x_0x_2x_3,\end{aligned}$$

and the propagation of the division property is summarized in a table, which is a counterpart of DDT in differential cryptanalysis. For example, the table of the division property for the above S-box is shown in Table 6. In this table, u and v are the input and output division property, respectively, and the propagation from u to v labeled x is possible. Otherwise, the propagation is impossible.

Xiang et al. found that the division trails can be searched with MILP [28] and the search was extended to bit-based division property by Sun et al. [29]. Then, modeling the above table, namely excluding the impossible patterns in the above table is necessary. Due to the same format as modeling DDT, it can be done by generating a large matrix with SAGE Math or logical computation model, and then applying the reduction algorithm. Here, our new reduction algorithm can be applied similarly as modeling DDT.

Table 6. Possible propagations of the division property for an S-box.

$u \setminus v$	0	1	2	4	8	3	5	9	6	A	C	7	B	D	E	F
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3		x	x			x	x	x	x	x	x	x	x	x	x	x
5			x		x	x	x	x	x	x	x	x	x	x	x	x
9				x	x	x	x	x	x	x	x	x	x	x	x	x
6				x		x	x	x	x	x	x	x	x	x	x	x
A				x	x	x	x	x	x	x	x	x	x	x	x	x
C				x	x	x	x	x	x	x	x	x	x	x	x	x
7					x					x	x	x	x	x	x	x
B							x			x	x	x	x	x	x	x
D						x	x	x	x	x	x	x	x	x	x	x
E					x		x	x	x	x	x	x	x	x	x	x
F																x

7 Concluding Remarks

In this paper, we revisited the reduction algorithm to model DDT by using MILP. Compared to previous one with the greedy algorithm, our method which is based on another small MILP problem can ensure the minimal number of inequalities to model DDT. It also enables users to choose the number of inequalities included in the system.

We then solved a 5-round and 6-round differential bound search for LILLIPUT with various numbers of inequalities per S-box. The results show that, on the contrary to the previous belief, minimizing the number of inequalities increases the runtime of the entire problem significantly.

We also discussed that the same reduction algorithm can be used to model the possible propagation patterns of the division property for S-box.

Many researches have been done on the automated search with MILP. Because most of the papers do not explain how many inequalities are used to model DDT, it is hard to distinguish which research can be improved. However, we believe that the general knowledge provided by this paper helps optimizing the speed of the differential search, hence helps future implementors.

References

1. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, New York (1993). doi:[10.1007/978-1-4613-9314-6](https://doi.org/10.1007/978-1-4613-9314-6)
2. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-34704-7_5](https://doi.org/10.1007/978-3-642-34704-7_5)

3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). doi:[10.1007/978-3-540-74735-2_31](https://doi.org/10.1007/978-3-540-74735-2_31)
4. Grosso, V., Leurent, G., Standaert, F.-X., Varici, K.: LS-designs: bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015). doi:[10.1007/978-3-662-46706-0_2](https://doi.org/10.1007/978-3-662-46706-0_2)
5. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (Related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). doi:[10.1007/978-3-662-45611-8_9](https://doi.org/10.1007/978-3-662-45611-8_9)
6. 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. Cryptology ePrint Archive, Report 2014/747 (2014)
7. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: a new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-25286-0_1](https://doi.org/10.1007/978-3-642-25286-0_1)
8. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-23951-9_23](https://doi.org/10.1007/978-3-642-23951-9_23)
9. Suzuki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: a lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-35999-6_22](https://doi.org/10.1007/978-3-642-35999-6_22)
10. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-34961-4_14](https://doi.org/10.1007/978-3-642-34961-4_14)
11. 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). doi:[10.1007/978-3-642-10433-6_22](https://doi.org/10.1007/978-3-642-10433-6_22)
12. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-23951-9_22](https://doi.org/10.1007/978-3-642-23951-9_22)
13. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011). doi:[10.1007/978-3-642-21554-4_19](https://doi.org/10.1007/978-3-642-21554-4_19)
14. Biham, E., Anderson, R., Knudsen, L.: Serpent: a new block cipher proposal. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 222–238. Springer, Heidelberg (1998). doi:[10.1007/3-540-69710-1_15](https://doi.org/10.1007/3-540-69710-1_15)
15. Berger, T.P., Francq, J., Minier, M., Thomas, G.: Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: lilliput. IEEE Trans. Comput. **65**, 2074–2089 (2015)
16. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: a block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 411–436. Springer, Heidelberg (2015). doi:[10.1007/978-3-662-48800-3_17](https://doi.org/10.1007/978-3-662-48800-3_17)

17. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M.: Minalpher v1.1. Submitted to CAESAR (2015)
18. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Cryptology ePrint Archive, Report 2014/084 (2014)
19. Beierle, C., et al.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg (2016). doi:[10.1007/978-3-662-53008-5_5](https://doi.org/10.1007/978-3-662-53008-5_5)
20. Gurobi Optimization Inc.: Gurobi Optimizer 6.5. Official webpage (2015). <http://www.gurobi.com/>
21. Gamrath, G., Fischer, T., Gally, T., Gleixner, A.M., Hendel, G., Koch, T., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Vigerske, S., Weninger, D., Winkler, M., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 3.2. Technical report 15–60, ZIB, Takustr. 7, 14195 Berlin (2016)
22. Ilog, I.: IBM ILOG CPLEX Optimization Studio V12.7.0 documentation. Official webpage (2016). <https://www-01.ibm.com/software/websphere/products/optimization/cplex-studio-community-edition/>
23. Sasaki, Y., Todo, Y.: New differential bounds and division property of shape LILIPUT: block cipher with extended generalized Feistel network. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, Springer, (to appear 2016). Pre-proceedings version was distributed at the workshop
24. Berger, T.P., Minier, M., Thomas, G.: Extended generalized feistel networks using matrix representation. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 289–305. Springer, Heidelberg (2014). doi:[10.1007/978-3-662-43414-7_15](https://doi.org/10.1007/978-3-662-43414-7_15)
25. Suzuki, T., Minematsu, K.: Improving the generalized feistel. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 19–39. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-13858-4_2](https://doi.org/10.1007/978-3-642-13858-4_2)
26. 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). doi:[10.1007/978-3-662-46800-5_12](https://doi.org/10.1007/978-3-662-46800-5_12)
27. 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). doi:[10.1007/978-3-662-52993-5_18](https://doi.org/10.1007/978-3-662-52993-5_18)
28. 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). doi:[10.1007/978-3-662-53887-6_24](https://doi.org/10.1007/978-3-662-53887-6_24)
29. Sun, L., Wang, W., Liu, R., Wang, M.: MILP-Aided Bit-Based Division Property for ARX-Based Block Cipher. Cryptology ePrint Archive, Report 2016/1101 (2016)