Improving the MILP-based Security Evaluation Algorithms against Differential Cryptanalysis Using Divide-and-Conquer Approach

Chunning Zhou[1,2], Wentao Zhang[1,2], Tianyou Ding[1,2], and Zejun Xiang[3]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhouchunning, zhangwentao}@iie.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Faculty of Mathematics and Statistics, Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan, China

Abstract. In recent years, Mixed Integer Linear Programming (MILP) has been widely used in cryptanalysis of symmetric-key primitives. For differential and linear cryptanalysis, MILP can be used to solve the two problems: calculation of the minimum number of differential/linear active S-boxes, and search for the best differential/linear characteristics. There are already numerous papers published in this area which either find differential characteristics with good probabilities or ones with small numbers of active S-boxes. However, the efficiency is not satisfactory enough for many symmetric-key primitives. In this paper, we will greatly improve the efficiency of the search algorithms for both the two problems based on MILP. Solving the problems of the calculation of the minimum number of differential/linear active S-boxes and the search for the best differential/linear characteristics can be equivalent to solving an MILP model whose feasible region is the set of all possible differential/linear characteristics. However, searching the whole feasible region is inefficient and high-probability differential/linear characteristics are likely to appear on the smaller feasible region with a low number of active S-boxes at some round. Inspired by the idea of divide-and-conquer approach, we divide the whole feasible region into smaller ones and separately search them. We apply our method to 5 lightweight block ciphers: PRESENT, GIFT-64, RECTANGLE, LBLOCK and TWINE. For each cipher, we obtain better results than the best-known ones. For the calculation of the minimum number of differential active S-boxes, we can reach 31-round PRESENT, 28-round GIFT-64 and 17-round RECTANGLE respectively. For the search for the best differential characteristics, we can reach 23, 14, 15, 21 and 17 rounds for the five ciphers respectively. Based on the duality between the differential cryptanalysis and the linear cryptanalysis, we leave the case for linear cryptanalysis in our future work.

Keywords: Block Cipher · Differential Cryptanalysis · MILP · Divide-and-Conquer

1 Introduction

As a fundamental primitive of cryptography, block ciphers have received extensive attention from academia and industry. The most important criteria for designing a block cipher is to ensure that it can resist all known attacks, especially differential and linear cryptanalysis [5, 9].

For evaluating the security of a block cipher against differential and linear cryptanalysis, there are usually two approaches. One is to calculate the minimum number of active S-boxes to obtain an upper bound of the maximum probability (or linear bias). The other approach is to calculate the maximum probability of differential characteristics and the maximum linear bias of linear characteristics. For some block ciphers, this needs a huge workload and impossible to be accomplished in reasonable time.

To calculate the maximum probability and search for the best differential characteristics, Matsui proposed a branch and bound search algorithm [10]. Matsui's search algorithm is one of the most powerful and classic search tools, but difficult to implement in some cases. In recent years, a simple tool, Mixed Integer Linear Programming (MILP) is widely used in this area. Wu *et al.* [20], Mouha *et al.* [11] and Sun *et al.* [16] modeled the differential behavior of a block cipher as an MILP problem, which can obtain a lower bound of the minimum number of active S-boxes. In [17], Sun *et al.* introduced two systematic methods for generating the linear inequalities describing the S-box operation more accurately, and proposed a heuristic algorithm for finding actual (related-key) differential characteristics. Then in [14], Sun *et al.* constructed the MILP model whose feasible region is exactly the set of all possible (related-key) differential characteristics by using the convex hull computation method. Therefore, their model can be used to obtain the exact minimum number of active S-boxes and the best differential characteristic.

The MILP model built for a r-round cipher cannot be solved in reasonable time when r is large, thus a lot of papers are published to improve the efficiency. A simple split approach was introduced in [11, 17], which splits r rounds into two parts: the first r_1 rounds and the last $(r-r_1)$ rounds. In [15], Sun *et al.* restricted the differential patterns of S-boxes in the feasible region of the model to obtain good differential characteristics, but not the best. Then in [23], Zhang *et al.* added the constraints derived from the bounding condition of Matsui's algorithm to the model, resulting in the model solved faster. Although the feasible regions are reduced, Zhang *et al.*'s models are still time-consuming when r is large. Therefore, it is of great importance to improve the efficiency of the MILP-based security evaluation algorithms against differential/linear cryptanalysis.

1.1 Our Contributions

In this paper, by using the idea of divide-and-conquer approach, we greatly improve the efficiency of the MILP-based search algorithms for both the minimum number of active S-boxes and the best differential characteristics. Note that we only focus on security evaluation against differential cryptanalysis in this paper and leave the case against linear cryptanalysis for our future work. The main contributions are as follows:

1. We propose a new expression of an S-box for modeling the problem of the calculation of the minimum number of active S-boxes. The models built by using the new expression are solved faster than the ones using Sun *et al.*'s expression [17]. For example, the running time is reduced by a factor of 4 for 9-round PRESENT.
2. Inspired by the idea of divide-and-conquer approach, we divide the feasible region of the original model into smaller ones and separately search them. Since high-probability differential characteristics are likely to have a low number of active S-boxes at some round, the smaller feasible regions are partitioned with specific active S-boxes information (*e.g.*, number, indices and input differences). By searching all smaller feasible regions, equivalently, solving the corresponding submodel, we obtain the optimal objective value of the original model. In the search process, we use the branch-and-bound techniques to cut duplicate and unnecessary smaller feasible regions, which reduces the search space significantly.
3. We apply our improved search algorithm to five lightweight block ciphers: PRESENT [6], GIFT [2], RECTANGLE [22], LBLOCK [21] and TWINE [18]. For each cipher, we obtain better results than the previous known results from MILP method. For the minimum number of differential active S-boxes, we reach 31-round PRESENT, 28-round GIFT-64 and 17-round RECTANGLE respectively. For the best differential characteristics, we reach 23, 14, 15, 21 and 17 rounds for the 5 ciphers respectively. To compare with previous work, we solve the original models for the same targets and list the experimental results in Table 1-3. For the first problem, we build the original models using Sun *et al.*'s framework [17] but our new expression of an S-box. For the second problem, we build the original models using Sun *et al.*'s framework [14]. For both the problems, we use Sasaki *et al.*'s reduction algorithm [13] to reduce the inequalities in the convex hull computed from SageMath software [1]. From the tables, we see that for each cipher, we can cover more rounds with less time. Thus our search algorithm is more efficient than solving the original model.

1.2 Organization

In Section 2, we introduce Mouha *et al.*'s [11] and Sun *et al.*'s [16, 17, 14] model frameworks for solving the problems in cryptography. In Section 3, we introduce a new expression of an S-box and give some notations. In Section 4 and 5, we introduce the basic search process using the idea of divide-and-conquer approach, then improve it using branch-and-bound techniques. In Section 6, we apply our search algorithm to five block ciphers and discuss its efficiency. In Section 7, we conclude the paper and provide some ideas for future work. The experimental results are presented in Appendices.

Table 1: Results on the minimum number of differential active S-boxes

PRESENT			GIFT-64			RECTANGLE		
Rounds	Time(original model)	Time (Sect. 6)	Rounds	Time(original model)	Time (Sect. 6)	Rounds	Time(original model)	Time (Sect. 6)
11	21.5h	56s	11	12.6h	0.55h	13	17.22h	0.22h
12	> 24h	57s	12	> 24h	0.57h	14	> 24h	0.51h
31	-	0.04h	28	-	1.02h	17	-	5.96h

Table 2: Results on the best differential characteristics (1)

PRESENT			GIFT-64			RECTANGLE		
Rounds	Time(original model)	Time (Sect. 6)	Rounds	Time(original model)	Time (Sect. 6)	Rounds	Time(original model)	Time (Sect. 6)
10	16h	1h	7	3h	0.47h	10	5.17h	0.15h
11	> 24h	1.08h	8	> 24h	8.52h	11	> 24h	0.24h
23	-	35.88h	14	-	36.25h	15	-	10.34h

Table 3: Results on the best differential characteristics (2)

LBLOCK			TWINE		
Rounds	Time(original model)	Time (Sect. 6)	Rounds	Time(original model)	Time (Sect. 6)
12	14h	0.02h	13	18.94h	1.44h
13	> 24h	0.22h	14	> 24h	4.34h
21	-	29.77h	17	-	27.42h

2 Related Work

In this section, we introduce Mouha *et al.*'s [11] and Sun *et al.*'s[16, 17, 14]'s model framework.

2.1 Mouha *et al.*'s Model Framework

For word-oriented block ciphers, Mouha *et al.* considered truncated differences, and introduced a model framework for calculating a lower bound of the minimum number of active S-boxes. They use a 0-1 variable to describe a word-level difference, such that the variable equals to 1 if and only if the input word is non-zero. Assume a cipher is composed of 3 word-oriented operations: XOR, linear transformation and S-box, they introduced the following equations to describe the word-level difference propagation through a cipher.

Equations Describing the XOR Operation Let a, b and c denote the word-level input and corresponding output differences of the XOR operation, the following equations are used to describe the XOR operation:

$$\begin{cases} a + b + c \geq 2d_{\oplus}, \\ d_{\oplus} \geq a, d_{\oplus} \geq b, d_{\oplus} \geq c, \end{cases}$$

where $d_{\oplus}$ is a dummy variable taking values in $\{0, 1\}$. If each one of a, b, and c represents one bit, an addition inequality $a + b + c \leq 2$ is needed.

Equations Describing the Linear Transformation Assume $(x_{i_0}, x_{i_1}, ..., x_{i_{m-1}})$ and $(y_{j_0}, y_{j_1}, ..., y_{j_{m-1}})$ be the word-level input and output difference of the linear transformation L respectively. Given the differential branch number $\mathcal{B}_{\mathcal{D}}$ of L, the linear transformation is described by:

$$\begin{cases} \sum_{k=0}^{m-1} x_{i_k} + \sum_{k=0}^{m-1} y_{j_k} \geq \mathcal{B}_{\mathcal{D}} d_L, \\ d_L \geq x_{i_k}, d_L \geq y_{j_k}, k \in \{0, ..., m-1\}, \end{cases}$$

where d_L is a dummy variable taking values in $\{0, 1\}$.

Objective Function The objective function is to minimize the number of active S-boxes, that is, the sum of all variables representing word-level input differences of S-boxes of each round.

Additional Constraints To avoid a trivial solution, it needs an additional constraint to ensure that at least one S-box is active. Besides, all dummy d-variables, and the variables representing the plaintext differences are restricted to be 0-1 variables.

Because this framework did not consider the bitwise S-box and the bitwise permutation layer, it is not applicable to bit-oriented ciphers. In Section 2.2, we will introduce Sun *et al*'s framework [16, 17, 14] for bit-oriented block ciphers.

2.2 Sun *et al.*'s Model Framework

In[16, 17, 14], Sun *et al.* considered the bit-level difference and used a 0-1 variable to denoted it, such that the variable equals to 1 if and only if the bit-level difference is nonzero.

Framework for Calculating the Minimum Number of Active S-boxes To model the S-box operation, Sun *et al.*[16] used a 0-1 variable A_t to denote the word-level input difference of an S-box, such that $A_t = 1$ if and only if the input word of the S-box is nonzero, $1 \leq t \leq N_S \times r$, where N_S is the number of S-boxes each round, and r is the number of rounds. Therefore, the objective function is to minimize the sum of all A_t.

Suppose $(x_{i_0}, ..., x_{i_{w-1}})$ and $(y_{j_0}, ..., y_{j_{v-1}})$ be the input and output differences of an $w \times v$ S-box marked by A_t, the following equations (1) - (3) are introduced to describe the S-box:

$$\begin{cases} A_t - x_{i_k} \geq 0, k \in \{0, ..., w-1\}, \\ x_{i_0} + x_{i_1} + ... + x_{i_{w-1}} - A_t \geq 0, \end{cases} \tag{1}$$

and

$$\begin{cases} \sum_{k=0}^{w-1} x_{i_k} + \sum_{k=0}^{v-1} y_{j_k} \geq \mathcal{B}_{\mathcal{S}} d_S, \\ d_S \geq x_{i_k}, k \in \{0, ..., w-1\}, \\ d_S \geq y_{j_k}, k \in \{0, ..., v-1\}, \end{cases} \tag{2}$$

where d_S is a dummy variable taking values in $\{0, 1\}$, and the branch number $\mathcal{B}_{\mathcal{S}}$ of an S-box is defined as

$$\mathcal{B}_{\mathcal{S}} = \min_{a \neq b} \{\mathrm{wt}((a \oplus b) || S(a) \oplus S(b)) : a, b \in \mathbb{F}_2^w\},$$

where wt(·) is the standard Hamming weight. For the bijective S-box, additional constraints are derived:

$$\begin{cases} wy_{j_0} + wy_{j_1} + ... + wy_{j_{v-1}} - (x_{i_0} + x_{i_1} + ...x_{i_{w-1}}) \geq 0, \\ vx_{i_0} + vx_{i_1} + ... + vx_{i_{w-1}} - (y_{j_0} + y_{j_1} + ...y_{j_{v-1}}) \geq 0. \end{cases} \tag{3}$$

Using the constraints and the objective function above, an MILP model is built to calculate a lower bound of the minimum number of active S-boxes for bit-oriented ciphers.

However, a feasible solution of the model constructed above is not guaranteed to be a valid differential characteristic because the constraints describing the S-box operation are rough. To describe difference propagations of an S-box more accurately, Sun *et al.* [17] proposed the convex hull computation method to remove the invalid differential characteristics from the feasible region of the model. They treated a possible difference propagation $(x_{i_0}, ..., x_{i_{w-1}}) \rightarrow (y_{j_0}, ..., y_{j_{v-1}})$ of an $w \times v$ S-box as a point in $\mathbb{F}_2^{w+v}$:

$$(x_{i_0}, ..., x_{i_{w-1}}, y_{j_0}, ..., y_{j_{v-1}}) \in \mathbb{F}_2^{w+v}.$$

And all possible difference propagations of the S-box constitute a set of finitely many discrete points. By computing the H-Representation of the convex hull of the set with the help of SageMath software [1], inequalities are generated to remove the impossible difference propagations of the S-box. Then in [14], Sun *et al.* proved that the feasible region of the model built by using the convex hull computation method is exactly the set of all valid differential characteristics. Generally, the number of inequalities computed from SageMath is very large. In [17, 14], Sun *et al.* introduced a greedy algorithm to select a small number of inequalities. However, the number of their inequalities is not always the minimum. Then in [13], Sasaki *el at.* proposed a MILP-based reduction algorithm to minimize the number of the inequalities. These reduced inequalities are used to exactly describe the S-box operation. Note that when using convex hull computation method to model the S-box, the constraints (2) and (3) can be omitted since all impossible difference propagations are removed, and all variables involved are restricted to be 0-1 variables.

Framework for Searching for the Best Differential Characteristic In [14], Sun *et al.* introduced a model framework for finding the best differential characteristic for block ciphers.

Taking the PRESENT S-box as an example, they treated the possible difference propagation $(x_0, x_1, x_2, x_3) \rightarrow (y_0, y_1, y_2, y_3)$ with the probability Pr as a point:

$$(x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3, p_0, p_1) \in \mathbb{F}_2^{10},$$

where

$$(p_0, p_1) = \begin{cases} (0, 0), \text{if Pr} = 1; \\ (0, 1), \text{if Pr} = 2^{-2}; \\ (1, 1), \text{if Pr} = 2^{-3}, \end{cases} \tag{4}$$

that is to say, $\Pr = 2^{-(p_0+2p_1)}$. Using the convex hull computation method and Sasaki *el at.*'s reduction algorithm, the inequalities are obtained to exactly describe all the possible 10-dimension points. To find the best differential characteristic, that is, the differential characteristic with the maximal probability, the objective function is to minimize the sum of $p_0 + 2p_1$.

3 Preliminaries

In this paper, we evaluate the security of a block cipher against differential attacks. Calculating the minimum number of active S-boxes and searching for the best differential characteristic are two ways to achieve it. As introduced in Section 2, each cryptographic problem above can be modeled as an MILP problem. In this section, we first introduce a new expression of an S-box for modeling the first problem, then present some notations.

3.1 New Expression of an S-box for Modeling the Problem of the Calculation of the Minimum Number of Active S-boxes

To calculate the minimum number of active S-boxes of a block cipher, we need to model all possible difference propagations $(x_{i_0}, ..., x_{i_{w-1}}, y_{j_0}, ..., y_{j_{v-1}})$ and the input word A_t of an $w \times v$ S-box. Two expressions are provided:

1. Sun *et al.* [17] treated a possible difference propagation as a point in $\mathbb{F}_2^{w+v}$, then used the convex hull computation method to describe all possible difference propagations(see Section 2.2). Besides, they used the equation (1) to describe A_t;
2. We introduce a new expression. For each point in $\mathbb{F}_2^{w+v}$ introduced by Sun *et al.*, we extend it to a $(w + v + 1)$-dimension point by adding one bit A_t:

$$(x_{i_0}, ..., x_{i_{w-1}}, y_{j_0}, ..., y_{j_{v-1}}, A_t) \in \mathbb{F}_2^{w+v+1}.$$

   Then using the convex hull computation method, we obtain the inequalities for describing all the possible $(w + v + 1)$-dimension points. In this method, the 5 inequalities in the equation (1) can be omitted.

For comparison, we apply each of the two expressions above to model the problems for PRESENT, GIFT-64 and RECTANGLE. For each expression, we use Sasaki *et al.*'s reduction algorithm [13] to reduce the inequalities computed from SageMath. We list the experimental results in Table 4 and 5. From the tables, we see that for each cipher, the number of inequalities describing an S-box using our method is less than Sun *et al.*'s, and the models of ours are solved easier. Taking the PRESENT S-box as an example, it needs 19 inequalities to describe an S-box using our expression, while 21 + 5 inequalities using Sun *et al.*'s. The time of solving the model for 9-round PRESENT using our expression is reduced by a factor of 4 compared with Sun *et al.*'s.

Table 4: Comparison of the number of inequalities for an S-box

PRESENT		GIFT-64		RECTANGLE	
Number ([17])	Number (ours)	Number ([17])	Number (ours)	Number ([17])	Number (ours)
$5+21$	19	$5+21$	22	$5+21$	22

Table 5: Comparison of the time of solving a model

Rounds	PRESENT		GIFT-64		RECTANGLE	
	Time(s) ([17])	Time(s) (ours)	Time(s) ([17])	Time(s) (ours)	Time(s) ([17])	Time(s) (ours)
1	0	0	0	0	0	0
2	0	0	0	4	0	0
3	3	2	1	3	1	1
4	10	13	6	17	2	8
5	208	42	5	34	8	9
6	784	429	67	56	14	37
7	1905	665	236	138	54	74
8	4048	3255	2232	421	502	229
9	35549	8333	6214	4753	2163	562
10	-	-	16518	14217	10504	5019
Total time	11.81h	3.54h	7.2h	5.46h	3.68h	1.65h

3.2 Notations

We denote the problem of the calculation of the minimum number of active S-boxes as the AS-Problem, and the problem of the search of the best differential characteristic as DC-Problem. As introduced in Section 2 and 3.1, each of the two problems can be equivalent to an MILP model. We call this model as the original model. We use Gurobi Optimizer [12] to solve the MILP model $\mathcal{M}$. In Gurobi, the functions $\mathcal{M}$`.addConstr()` is used to add constraints into $\mathcal{M}$, and $\mathcal{M}$`.objVal()` returns the optimal objective value of $\mathcal{M}$.

For a block cipher, we have the following notations:

r:	the number of rounds;
n:	the block size;
N_S:	the number of S-boxes each round;
N_A:	the number of active S-boxes;
D_i^{in}:	the input difference of the ith round;
D_i^{out}:	the output difference of the ith round;
DP_i^{in}:	the input difference patterns of S-boxes at the ith round;

4 Basic Search Process Using Divide-and-Conquer Approach

From previous work, the original model built for solving the AS/DC-Problem for a r-round block cipher becomes difficult to be solved when r is large. The feasible region of the original model is the set of all possible r-round differential characteristics. Inspired by the idea of divide-and-conquer approach, we divide the feasible region into smaller ones, then separately solve these submodels with smaller feasible regions.

4.1 Basic Search Process for SP-network Ciphers

For SP-network ciphers, we use Function `BuildModel-SP`() to build the submodels corresponding to the smaller feasible regions we divide, and Algorithm 1 to obtain the optimal objective value of the original model. In the function, we use $A_{i,j}$ to denote the word-level input difference of the jth S-box at the ith round, and $x_{i,j}$ to denote the jth bit input difference of the ith round, namely, the jth bit output difference of the $(i-1)$th round.

Dividing the Feasible Region of the Original Model For SP-network cipher, the differential characteristics have greater than or equal to 1 active S-box at each round, and those with the highest probability are likely to have 1 or 2 active S-boxes at some round. Therefore, we first divide the feasible region of the original model built for a r-round cipher into 3 separate parts:

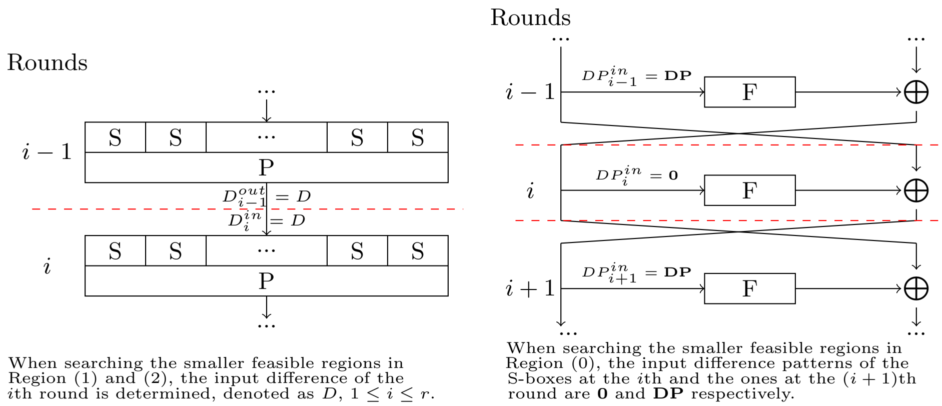
Region (1) In this region, the differential characteristics have greater than or equal to 1 active S-box at each round, but exactly 1 active S-box at some round;

Region (2) In this region, the differential characteristics have greater than or equal to 2 active S-boxes at each round, but exactly 2 active S-boxes at some round;

Region (3) In this region, the differential characteristics have greater than or equal to 3 active S-boxes at each round.

For Region (1) and (2), we traverse $i \in \{1, 2, ..., r\}$, s.t. there are exactly 1 and 2 active S-boxes at the ith round respectively. Then, we respectively traverse the indices and the input differences of the active S-boxes, which further divides the feasible region into smaller ones. In the smaller feasible region, the input difference of the ith round/the output difference of the $(i-1)$th round is determined by the indices and the input differences of the active S-boxes at the ith round.

Example 1. Take 64-bit PRESENT with the 4×4 S-box as an example, $N_S = 16$. Suppose that at some round, only the 1-st and the 2-nd S-boxes are active with the input differences $0x1$ and $0x2$ respectively. The input difference of this round/output difference of the last round is $0x1200000000000000$.



(a) Split for SP-network ciphers (b) Split for Feistel ciphers

Fig. 1: Split r rounds into two parts

Building Submodels Corresponding to Smaller Feasible Regions When searching the smaller feasible region in Region (a), $a \in \{1, 2\}$, the input difference of some round is determined, that is, $D_i^{in} = D \in \mathbb{F}_2^n, i \in \{1, 2, ..., r\}$, thus the corresponding submodel can be built: $\mathcal{M} \leftarrow$ `BuildModel-SP`(r, a, i, D, ϕ). However, we build two smaller submodels instead since they are easier to be solved. The two submodels are built by splitting the r rounds into the first $i-1$ rounds and the last $r-i+1$ rounds, as shown in Fig.1a.

- $\mathcal{M}^1 \leftarrow$ `BuildModel-SP`$(i-1, a, i-1, \phi, D)$. This model is built for an $(i-1)$-round cipher, with greater than or equal to a active S-boxes at each round except for the last round, and with D_{i-1}^{out} equals to D;
- $\mathcal{M}^2 \leftarrow$ `BuildModel-SP`$(r-i+1, a, 1, D, \phi)$. This model is built for a $(r-i+1)$-round cipher, with greater than or equal to a active S-boxes at each round except for the first round, and with D_1^{in} equals to D;

Linking the two submodels, we obtain the optimal objective value of the $\mathcal{M}$:

$$\mathcal{M}^1.objVal + \mathcal{M}^2.objVal. \tag{5}$$

For Region (3), we build the submodel by calling `BuildModel-SP`$(r, 3, \phi, \phi, \phi)$. This model is built for a r-round cipher, with greater than or equal to 3 active S-boxes at each round. The process of searching all smaller feasible regions is shown in Algorithm 1. Among the values storing in $ObjList$, the minimum one is exactly the optimal objective value of the original model.

4.2 Basic Search Process for Feistel Ciphers

For Feistel ciphers, we use Function `BuildModel-Feistel()` to build the submodels corresponding to the smaller feasible regions we divide, and Algorithm

Function BuildModel-SP($r, N_A, i, D_i^{in}, D_i^{out}$)

Data: r: the model is built for a r-round cipher;
N_A: the model is constrained with some round has greater than or equal to N_A active S-boxes;
i, $D_i^{in} = d_{i,0}^{in} d_{i,1}^{in} ... d_{i,n-1}^{in}$, $D_i^{out} = d_{i,0}^{out} d_{i,1}^{out} ... d_{i,n-1}^{out}$: the model is constrained with the ith round has determined input difference D_i^{in} and output difference D_i^{out};
Result: the submodel with additional constraints for SP-network ciphers.

Build an original model $\mathcal{M}$ for solving the AS/DC-Problem for a r-round SP-network cipher;
for $j \leftarrow 1$ **to** r **and** $j \neq i$ **do**
    $\mathcal{M}$.`addConstr`($\sum_{t=1}^{N_S} A_{j,t} \geq N_A$); `// the` j`th round has greater than or equal to` N_A `active S-boxes`
end
if $D_i^{in} \neq \phi$ **then**
    $\mathcal{M}$.`addConstr`($x_{i,j} = d_{i,j}^{in}, j = 0, 1, ..., n-1$); `// fix the input difference of the` i`th round`
end
if $D_i^{out} \neq \phi$ **then**
    $\mathcal{M}$.`addConstr`($x_{i+1,j} = d_{i,j}^{out}, j = 0, 1, ..., n-1$); `// fix the output difference of the` i`th round`
end
return $\mathcal{M}$;

Algorithm 1: The basic search process for SP-network ciphers.

Data: The AS/DC-Problem for a r-round SP-network cipher.
Result: The optimal objective value of the original model.

1 $ObjList = []$; `// a list storing the optimal objective values of the submodels corresponding to the smaller feasible regions`
`// Line 2-13: search Region (1) and (2)`
2 **for** $N_A \leftarrow 1$ **to** 2 **do**
3   **for** $i \leftarrow 1$ **to** r **do**
4   **forall the** *possible indices of the* N_A *active S-boxes* **do**
5   **forall the** *possible input differences of the* N_A *active S-boxes* **do**
6                 $D \leftarrow$ determined by the indices and the input differences of the N_A active S-boxes;
7                 $\mathcal{M}^1 \leftarrow$ `BuildModel-SP`($i-1, N_A, i-1, \phi, D$);
8                 $\mathcal{M}^2 \leftarrow$ `BuildModel-SP`($r-i+1, N_A, 1, D, \phi$);
9   Add ($\mathcal{M}^1$.`objVal()` + $\mathcal{M}^2$.`objVal()`) to $ObjList$;
10   **end**
11   **end**
12   **end**
13 **end**
14 $\mathcal{M}^3 \leftarrow$ `BuildModel-SP`($r, 3, \phi, \phi, \phi$); `// Search Region (3)`
15 Add $\mathcal{M}^3$.`objVal()` to $ObjList$;
16 **return** min($ObjList$);

Function BuildModel-Feistel($r, N_A, i, DP_i^{in}, DP_{i+1}^{in}$)

Data: r: the model is built for a r-round cipher;
N_A: the model is constrained with some round has greater than or equal to N_A active S-boxes;
i, $DP_i^{in} = (dp_{i,0}^{in}, dp_{i,1}^{in}, ..., dp_{i,N_S-1}^{in})$, $DP_{i+1}^{in} = (dp_{i+1,0}^{in}, dp_{i+1,1}^{in}, ..., dp_{i+1,N_S-1}^{in})$: the model is constrained with the ith round and the $(i+1)$th round have determined input difference patterns of S-boxes DP_i^{in} and DP_{i+1}^{in} respectively;
Result: The submodel with additional constraints for Feistel ciphers.

1 Build an original model $\mathcal{M}$ for solving the AS/DC-Problem for a r-round Feistel cipher;
2 **for** $j \leftarrow 1$ **to** r **and** $j \neq i$ **and** $j \neq i+1$ **do**
3     $\mathcal{M}$.`addConstr`($\sum_{t=1}^{N_S} A_{j,t} \geq N_A$); `// the` j`th round has greater than or equal to` N_A `active S-boxes`
4 **end**
5 **if** $DP_i^{in} \neq \phi$ **then**
6     $\mathcal{M}$.`addConstr`($A_{i,j} = dp_{i,j-1}^{in}, j = 1, 2, ..., N_S$); `// fix the input difference patterns of the S-boxes at the` i`th round`
7 **end**
8 **if** $DP_{i+1}^{in} \neq \phi$ **then**
9     $\mathcal{M}$.`addConstr`($A_{i+1,j} = dp_{i+1,j-1}^{in}, j = 1, 2, ..., N_S$); `// fix the input difference patterns of the S-boxes at the` $(i+1)$`th round`
10 **end**
11 **return** $\mathcal{M}$;

Algorithm 2: The basic search process for Feistel ciphers.

Data: The AS/DC-Problem for a r-round Feistel cipher.
Result: A lower bound of the optimal objective value of the original model.

1 $ObjList = []$; `// a list storing lower bounds of the optimal objective values of the submodels corresponding to smaller feasible regions`
`// Line 2-11: Search Region (0)`
2 **for** $i \leftarrow 1$ **to** r **do**
3   **forall the** *possible number of the active S-boxes at the* $(i+1)$*th round* **do**
4   **forall the** *possible indices of the active S-boxes at the* $(i+1)$*th round* **do**
5   **DP** $\leftarrow$ determined by the number and the indices of the active S-boxes at the $(i+1)$th round;
6             $\mathcal{M}^1 \leftarrow$ `BuildModel-Feistel`($i, 0, i-1, \mathbf{DP}, \mathbf{0}$);
7             $\mathcal{M}^2 \leftarrow$ `BuildModel-Feistel`($r-i+1, 0, 1, \mathbf{0}, \mathbf{DP}$);
8   Add ($\mathcal{M}^1$.`objVal`() + $\mathcal{M}^2$.`objVal`()) to $ObjList$;
9   **end**
10   **end**
11 **end**
12 $\mathcal{M}^3 \leftarrow$ `BuildModel-Feistel`($r, 1, \phi, \phi, \phi$); `// Search Region (1)`
13 Add $\mathcal{M}^3$.`objVal`() to $ObjList$;
14 **return** min($ObjList$);

2 to obtain a lower bound of the optimal objective value of the original model. In the function, we also use $A_{i,j}$ to denote the word-level input difference of the jth S-box at the ith round.

Dividing the Feasible Region of the Original Model For a r-round Feistel cipher, the differential characteristics have greater than or equal to 0 active S-boxes at each round, and those with the highest probability are likely to have 0 active S-box at some round. Therefore, we first divide the feasible region of the original model into 2 separate parts:

Region (0) In this region, the differential characteristics have greater than or equal to 0 active S-box at each round, but exactly 0 active S-box at some round;

Region (1) In this region, the differential characteristics have greater than or equal to 1 active S-boxes at each round.

For Region (0), we traverse $i \in \{1, 2, ..., r\}$, s.t. the ith round has 0 active S-box. To further divide the feasible region, we then traverse the number and the indices of the active S-boxes at the $(i + 1)$th round. In the smaller feasible region, input difference patterns of S-boxes at the ith round and the ones at the $(i+1)$ round are all determined. If we further divide the feasible region based on the input differences of the active S-boxes, the number of the resulting smaller feasible regions will be too large. Since the ith round is free, the input difference patterns of S-boxes at the $(i - 1)$th round equal to the ones at the $(i + 1)$th round. Note that when $i = r$, we refer $r + 1$ to $r - 1$.

Example 2. Take 64-bit LBLOCK with the 4×4 S-box as an example, $N_S = 8$. Suppose that there are 0 active S-box at the ith round, and only the 1st and the 2nd S-boxes are active at the $(i + 1)$th round, $i \in \{1, 2, ..., r\}$. The input difference patterns of S-boxes at the ith round and the ones at the $(i + 1)$th round are $(0, 0, 0, 0, 0, 0, 0, 0)$ and $(1, 1, 0, 0, 0, 0, 0, 0)$ respectively.

Building SubModels Corresponding to Smaller Feasible Regions When searching the smaller feasible region in Region (0), there is one round $i \in \{1, 2, ..., r\}$, s.t. $DP_i^{in} = \mathbf{0}, DP_{i+1}^{in} = \mathbf{DP} = (dp_0, dp_1, ..., dp_{N_S-1}), dp_j \in \{0, 1\}$. Thus we split the r rounds into the first i rounds and the last $r - i + 1$ rounds, as shown in Fig.1b, and build the two submodels:

- $\mathcal{M}^1 \leftarrow$ `BuildModel-Feistel`$(i, 0, i - 1, \mathbf{DP}, \mathbf{0})$. This submodel is built for an i-round cipher with greater than or equal to 0 active S-boxes at each round except for the $(i - 1)$th round and the ith round, and with the input difference patterns of S-boxes at the $(i - 1)$th round and the ones at the ith round equal to $\mathbf{DP}$ and $\mathbf{0}$ respectively;
- $\mathcal{M}^2 \leftarrow$ `BuildModel-Feistel`$(r-i+1, 0, 1, \mathbf{0}, \mathbf{DP})$. This submodel is built for a $r-i+1$-round cipher with greater than or equal to 0 active S-boxes at each round except for the 1st and the 2nd round, and with the input difference patterns of S-boxes at the 1st round and the ones at the 2nd round equal to $\mathbf{0}$ and $\mathbf{DP}$ respectively;

Because the input difference of the $(i+1)$th round is indeterminate, the two submodels we split can't be linked. By adding the optimal objective values of $\mathcal{M}^1$ and $\mathcal{M}^2$, we obtain a lower bound, rather than the exact value of the submodel corresponding to the smaller feasible region.

For Region (1), we build the corresponding submodel by calling `BuildModel-Feistel`$(r, 1, \phi, \phi, \phi)$. This submodel is built for a r-round cipher with greater than or equal to 1 active S-boxes at each round. The process of searching all smaller feasible regions is shown in Algorithm 2, in which the minimum one of the values storing in $ObjList$ is chosen as a lower bound of the optimal objective value of the original model.

4.3 Weakness of the Basic Search Process

In Algorithm 1 and 2, we respectively solve the AS/DC-Problem for SP-network and Feistel ciphers by solving several submodels with additional constraints. However, the number of the submodels need to be solved is large, and some submodels are time-consuming. Thus the basic search process needs a lot of calculation. In addition, because the two parts we split cannot be linked when searching Region (0) for Feistel ciphers, we cannot guarantee the value obtained is the optimal one. To make the search algorithm more efficient, we will improve the basic search processes in the next section.

5 Improving the Search Process Using Branch-and-Bound Techniques

In this section, we improve the search process using branch-and-bound techniques which are similar to Matsui's[10]. The main ideas are:

1. If we obtain a lower bound of the optimal objective value of the corresponding submodel no less than the current optimal objective value of the original model before searching a smaller feasible region, we can conclude that no better solution exists in this smaller feasible region. Thus we don't need to solve the corresponding submodel.
2. While the problems for different rounds are solved independently in previous work, we aim to use the knowledge obtained via solving the problems for $1, 2, ..., r-1$ rounds before solving the problems for r round

The improved search processes are shown in Algorithm 3 and 4 for SP-network and Feistel ciphers respectively. In these two search processes, some smaller feasible regions are cut, thus the number of submodels to be solved is reduced. Also, we obtain the optimal objective value of the original model for Feistel ciphers by solving more submodels. For the convenience of illustration, we mainly introduce the improved search process for SP-network ciphers. The techniques used in the case for Fesitel ciphers are similar.

For the original model built for a r-round block cipher, we use $Obj[r]$, $LbObj[r]$ and $CurObj[r]$ to respectively store the exact value, a lower bound and a current value of the optimal objective value. Before searching a smaller feasible region, we assign a value to the lower bound of the optimal objective value of the corresponding submodel. If the lower bound is no less than $CutObj[r]$, we cut the smaller feasible region for there is no better solution. Therefore, we aim to assign tighter bounds to lower bounds of the optimal objective values of the submodels.

5.1 Methods for Assigning Lower Bounds of the Optimal Objective Values of the Submodels

When searching the smaller feasible regions for SP-network ciphers, we build the submodels by calling `BuildModel-SP`(). For the submodels, we use an array to store lower bounds of the optimal objective values, and provide 6 methods to assign the array.

For the submodel $\mathcal{M} \leftarrow$ `BuildModel-SP`$(r, N_A, i, D_i^{in}, D_i^{out}), N_A \in \{1, 2, 3\}$, $i \in \{1, 2, ..., r, \phi\}$, $D_i^{in}, D_i^{out} \in \mathbb{F}_2^n$, we use $LbObj[r][N_A][i][D_i^{in}][D_i^{out}]$ to store a lower bound of the optimal objective value, and provide Method 1 - 4 for assigning values to the array. Specially, for the submodels corresponding to the smaller feasible regions in Region (1) and (2), *i.e.*, $N_A \in \{1, 2\}, D_i^{in} = D \in \mathbb{F}_2^n, D_i^{out} = \phi$, we further provide Method 5 - 6.

Method 1 Since the feasible region of $\mathcal{M}$ is a subset of the feasible region of the submodel built by calling `BuildModel-SP` $(r, N_A, \phi, \phi, \phi)$,

$$LbObj[r][N_A][i][D_i^{in}][D_i^{in}] \leftarrow LbObj[r][N_A][\phi][\phi][\phi] \tag{6}$$

Method 2 As mentioned in [11, 17], we simply split r rounds into the r_1 rounds and the $(r - r_1)$ rounds, $1 \le r_1 \le r - 1$. Then

$$LbObj[r][N_A][i][D_i^{in}][D_i^{in}] \leftarrow$$

$$\begin{cases} \max\limits_{i \le r_1 \le r} (LbObj[r_1][N_A][i][D_i^{in}][D_i^{out}] + LbObj[r - r_1][N_A][\phi][\phi][\phi]),\ 1 \le i < r, \\ \max\limits_{1 \le r_1 \le r} (LbObj[r_1][N_A][r_1][D_i^{in}][D_i^{out}] + LbObj[r - r_1][N_A][\phi][\phi][\phi]),\ i = r, \\ \max\limits_{1 \le r_1 \le r} (LbObj[r_1][N_A][i][D_i^{in}][D_i^{out}] + LbObj[r - r_1][N_A][\phi][\phi][\phi]),\ i = \phi. \end{cases} \tag{7}$$

Method 3 We observe that the models built by using the frameworks of Mouha *et al.* [11] (see Section 2.1) and Sun *et al.* [16] (see Section 2.2) are solved quickly and they can respectively provide lower bounds of the minimum number of active S-boxes for word-oriented and bit-oriented ciphers. By using their frameworks, we build the other model $\mathcal{M}'$, and

$$LbObj[r][N_A][i][D_i^{in}][D_i^{in}] \leftarrow \mathcal{M}'.objVal \times c,$$

where $c = 1$ if we aim to solve the AS-Problem, $c = -log_2 \Pr_{max}$ if we aim to solve the DC-Problem, and $\Pr_{max}$ is the maximum differential probability of a single S-box. Note that if the cipher is bit-oriented with the branch number of S-boxes equals to 2, this method is inefficient.

Method 4 We assign another value to $LbObj[r][N_A][i][D_i^{in}][D_i^{in}]$ by updating the first value on the right side of the equation (7). To achieve it, we solve the corresponding submodel for r_1 rounds if it has not been solved yet. In this method, we can obtain multiple results using the equation (7) as r_1 increments from the minimum number to r.

Method 5 Based on the equation (5),

$$\begin{gathered} LbObj[r][N_A][i][D][\phi] \leftarrow \\ LbObj[i-1][N_A][i-1][\phi][D] + LbObj[r-i+1][N_A][1][D][\phi], \end{gathered} \tag{8}$$

where the two values on the right side are initially assigned as the maximum value of the results from Method 1 and 2.

Method 6 Based on the search process in Algorithm 3, we have searched the candidate solutions satisfying the i'th round has exactly N_A active S-boxes before searching the smaller feasible region in Region (N_A), $N_A \in \{1,2\}, i' \in \{1,2,...,r\}$. Thus we can assume that the candidate solutions have no less than N_A+1 active S-boxes at the i'th round for finding better solutions. By adding the additional constraints to $\mathcal{M}$, we obtain the other submodel $\mathcal{M}'$. If we obtain a lower bound of the optimal objective value of $\mathcal{M}'$ no less than $CurObj[r]$, we assign $CurObj[r]$ to $LbObj[r][N_A][i][D][\phi]$. Since solving $\mathcal{M}'$ is time-consuming, we provide 2 methods for obtaining a lower bound of the optimal objective value of $\mathcal{M}'$, denoted as $LbObj_{\mathcal{M}'}$.

Method 6-1 If the cipher is word-oriented or bit-oriented with the branch number of S-boxes equals to 2, we obtain $LbObj_{\mathcal{M}'}$ using Method 3.

Method 6-2 We split the r rounds into 3 parts then combine them. To build the submodel for each part by calling `BuildModel-SP`(), we provide 3 methods for splitting the r rounds. Take $r=10, i=3$ as an example, the input difference of the 3th round is D, and we assume that round $i'=4,5,6,7$ has greater than or equal to N_A+1 active S-boxes.

- Split 1: (1) round 1 to 3; (2) round 4 to 7; (3) round 8 to 10. Then $LbObj_{\mathcal{M}'} \leftarrow$

  $$LbObj[3][N_A][\phi][\phi][\phi] + LbObj[4][N_A+1][\phi][\phi][\phi] + LbObj[3][N_A][\phi][\phi][\phi].$$

- Split 2: (1) round 1 to 2; (2) round 3 to 7; (3) round 8 to 10. Then $LbObj_{\mathcal{M}'} \leftarrow$

  $$LbObj[2][N_A][2][\phi][D] + LbObj[5][N_A+1][1][D][\phi] + LbObj[3][N_A][\phi][\phi][\phi],$$

- Split 3: (1) round 1 to 2. (2) round 3 to 8; (3) round 9 to 10. Since we don't know the number of active S-boxes at the 8th round, we consider two cases: one is the 8th round has exactly N_A active S-boxes, the other is the 8th round has greater than or equal to N_A+1 active S-boxes. In the two cases, we respectively obtain $LbObj^1_{\mathcal{M}'}$ and $LbObj^2_{\mathcal{M}'}$ by using the lower bound array. If $LbObj^1_{\mathcal{M}'} \geq CurObj[r]$ and $LbObj^2_{\mathcal{M}'} \geq CurObj[r]$, we conclude that $LbObj_{\mathcal{M}'} \geq CurObj[r]$ and obtain a good lower bound. If $LbObj^1_{\mathcal{M}'} \geq CurObj[r]$ but $LbObj^2_{\mathcal{M}'} < CurObj[r]$, we assume that the 8th round has greater than or equal to $N_A + 1$ active S-boxes and split again the last two parts: (2) round 3 to 9; (3) round 10. Then we obtain new $LbObj^1_{\mathcal{M}'}$ and $LbObj^2_{\mathcal{M}'}$. If there is still no good $LbObj_{\mathcal{M}'}$, we split again the parts (2) and (3) until a good value is obtained or part (2) is from round 3 to $r = 10$.

5.2 Detailed Description of the Improved Search Process

In this section, we provide the detailed description of Algorithm 3 which is used to solve the AS/DC-Problem for a r-round SP-network cipher.

Before searching all smaller feasible regions, we initialize $LbObj[r]$ and $CurObj[r]$. If $LbObj[r] \geq CurObj[r]$, there is no better solution than the current one in the whole search space. In this case, we return $CurObj[r]$ as $Obj[r]$.

- Because all $Obj[k], k = 1, ..., r-1$ are supposed to be known,

$$LbObj[r] \leftarrow \max_{1 \leq r_1 \leq r-1} (Obj[r_1] + Obj[r - r_1])$$

- Initialize $CurObj[r]$ based on the optimal solution of the original model for $r-1$ rounds. Once we have solved the problem for $r-1$ rounds, we obtain the difference patterns of S-boxes of the optimal solution, denoted as $(DP^*_1, DP^*_2, ..., DP^*_r)$. In the original model for r rounds, by fixing the difference patterns of S-boxes at the first $r-1$ rounds just as $(DP^*_1, DP^*_2, ..., DP^*_r)$, we obtain a current optimal objective value as $CurObj[r]$.

Because the submodels built by calling `BuildModel-SP`$(r, N_A, \phi, \phi, \phi)$, $(N_A = 1, 2, 3)$ are time-consuming, we initialize $LbObj[r][N_A][\phi][\phi][\phi]$ in a short time. These values will be used for estimating lower bounds of the optimal objective values of other submodels.

- For $N_A = 1$, $LbObj[r][1][\phi][\phi][\phi] \leftarrow LbObj[r]$.
- For $N_A = 2, 3$, we use Method 3 if the cipher is word-oriented or bit-oriented with the branch number of S-boxes bigger than 2. Otherwise, we use Method 4 in which we solve the submodels for r_1 rounds, $r_1 = 1, 2, ..., R$, with R chosen to allow us solving the submodels in short time. Usually R is less than 7 for 64-bits block ciphers.

In Line 6 - 22, we search the smaller feasible regions in Region (1) and (2). To cut more smaller feasible regions using branch and bound techniques, we aim to update the $CurObj[r]$ as early as possible. To achieve it, we change the order of the smaller feasible regions we will search. On one hand, we can preferentially search the smaller feasible regions where optimal solutions of the original model are more likely to appear. If we can predict that the optimal solution is more likely to obtained when the candidate solution has only 2 active S-boxes at some round from some prior knowledge, we preferentially search the smaller feasible

Algorithm 3: The improved search process for SP-network ciphers.

Data: The AS/DC-Problem for a r-round cipher.
Result: The optimal objective value of the original model.

initialize $LbObj[r]$, $CurObj[r]$;
if $LbObj[r] \geq CurObj[r]$ **then**
  return $Obj[r] \leftarrow CurObj[r]$;
end
initialize $LbObj[r][N_A][\phi][\phi][\phi], N_A = 1, 2, 3$;
for N_A ***in*** $[1, 2]$ **or** $[2, 1]$ **do**
  for i ***in*** $[\lceil n/2 \rceil, \lceil n/2 \rceil + 1, \lceil n/2 \rceil - 1, ...,]$ **do**
    forall the *possible indices of the* N_A *active S-boxes* **do**
      forall the *possible input differences of the* N_A *active S-boxes* **do**
        $D \leftarrow$ determined by the indices and the input differences of the N_A active S-boxes;
        $LbObj[r][N_A][i][D][\phi] \leftarrow$ the maximum value of the results from Method 1, 2, 5, 6-2;
        $k = 1$;
        while $LbObj[r][N_A][i][D][\phi] < CurObj[r]$ **and** $k \leq 5$ **do**
          $LbObj[r][N_A][i][D][\phi] \leftarrow \max(LbObj[r][N_A][i][D][\phi],$ `UpdateLowerBound`$(k, r, N_A, i, D, CurObj[r]))$;
          $k = k + 1$
        end
        if $LbObj[r][N_A][i][D][\phi] < CurObj[r]$ **then**
          $CurObj[r] \leftarrow LbObj[r][N_A][i][D][\phi]$;
        end
      end
    end
  end
end
if $LbObj[r][3][\phi][\phi][\phi] < CurObj[r]$ **then**
  Update $LbObj[r][3][\phi][\phi][\phi]$ using Method 4;
end
$CurObj[r] \leftarrow \min(CurObj[r], LbObj[r][3][\phi][\phi][\phi])$;
return $Obj[r] \leftarrow CurObj[r]$;

Function `UpdateLowerBound`$(k, r, N_A, i, D_i^{in}, CurObj[r])$; `// the 5 functions are listed in the ascending order of their solving time`
begin
  case $k = 1$
    Update the values on the right side of equation (8) by using Method 3;
    return the result from Method 5;
  endsw
  case $k = 2$
    return the result from Method 3;
  endsw
  case $k = 3$
    return the result from Method 6-1;
  endsw
  case $k = 4$
    Update the values used in Method 6-2 by using Method 4;
    return the result from Method 6-2;
  endsw
  case $k = 5$
    Update the values on the right side of equation (8) by using Method 4;
    return the result from Method 5;
  endsw
end

regions in Region (2). In this case, we modify $N_A \in [1,2]$ to $N_A \in [2,1]$ in Line 6. On the other hand, we can preferentially search the smaller feasible regions for which the submodels are easier to be solved. When searching the smaller feasible region, we solve two submodels for $(i-1)$ and $(r-i+1)$ rounds respectively in the equation (5). As $|(i-1)-(r-i+1)|$ is smaller, the total time to solve the two submodels is smaller. Therefore, we modify $i \in [1,2,...,r]$ to $i \in [\lceil r/2 \rceil, \lceil r/2 \rceil+1, \lceil r/2 \rceil-1, ...,]$. When searching a fixed smaller feasible region, we initialize a lower bound of the optimal objective value of the corresponding submodel, then update it by using Function `UpdateLowerBound`(). The last step of the function is assigning the exact value to the lower bound. If the updated lower bound is still less than $CurObj[r]$, we update $CurObj[r]$ by the lower bound which equals to optimal objective value of the submodel corresponding to the smaller feasible region.

When searching Region (3), a lower bound of the optimal objective value of the corresponding submodel, *i.e.*, $LbObj[r][3][\phi][\phi][\phi]$ have been assigned before. If $LbObj[r][3][\phi][\phi][\phi]$ is no less than $CurObj[r]$, the smaller feasible region is cut. Otherwise, we use Method 4 to update $LbObj[r][3][\phi][\phi][\phi]$. Generally, this smaller feasible region is cut for lightweight block ciphers.

After searching all smaller feasible regions, we return $CurObj[r]$ as the optimal objective value of the original model.

Similar to SP-network ciphers, we improve the search process for Feistel ciphers in Algorithm 4. For the submodels built by calling `BuildModel-Feistel` $(r, N_A, i, DP_i^{in}, DP_{i+1}^{in})$, $N_A \in \{0,1\}, i \in \{1,2,...,r,\phi\}$, we use $LbObj[r][N_A][i]$ $[DP_i^{in}][DP_{i+1}^{in}]$ to store a lower bound of the optimal objective value. Specifically, because the two smaller submodels we split can't be linked when searching the smaller feasible region in Region(0), we update $CurObj[r]$ by solving the submodel corresponding to the smaller feasible region(Line 13 in Algorithm 4).

6 Applications to PRESENT, GIFT-64, RECTANGLE, LBLOCK and TWINE

In this section, we apply our search algorithm to PRESENT, GIFT-64, RECTANGLE, LBLOCK and TWINE. For each of the five ciphers, we obtain the exact minimum number of active S-boxes ($\#AS$) and the best differential characteristic with the probability $\Pr_{max}$, as summarized in Table 6-7. For each cipher, our results cover more rounds than the known results, and the best differential characteristics are given in Appendix B. We set the parameter MIPFocus in GUROBI to 2 to improve the solving time(in this setting, the solver tends to find an optimal solution rather than a feasible one).

6.1 Experimental Results

PRESENT The branch number of the PRESENT S-box is 3, thus in the search process, the branches may be cut early by using Method 3. From the experimen-

Algorithm 4: Improved search process for Feistel ciphers.

Data: The AS/DC-Problem for a r-round Feistel cipher.
Result: The optimal objective value of the original model.

1 **initialize** $LbObj[r]$, $CurObj[r]$;
2 **if** $LbObj[r] \geq CurObj[r]$ **then**
3   **return** $Obj[r] \leftarrow CurObj[r]$.;
4 **end**
5 **initialize** $LbObj[r][N_A][\phi][\phi][\phi], N_A = 0, 1$;
`// Line 6-18: Search Region (0)`
6 **for** i ***in*** $[\lceil r/2 \rceil, \lceil r/2 \rceil + 1, \lceil r/2 \rceil - 1, ...,]$ **do**
7   **forall the** *possible number of the active S-boxes at the* $(i+1)th$ *round* **do**
8   **forall the** *possible indices of the active S-boxes at the* $(i+1)th$ *round* **do**
9             $\mathbf{DP} \leftarrow$ determined by the number and the indices of the active S-boxes at the $(i+1)$th round;
10   **initialize** and **update** $LbObj[r][0][i][\mathbf{0}][\mathbf{DP}]$;
11   **if** $LbObj[r][0][i][\boldsymbol{0}][\boldsymbol{DP}] < CurObj[r]$ **then**
12                 $\mathcal{M} \leftarrow$ `BuildModel-Feistel`$(r, 0, i, \mathbf{0}, \mathbf{DP})$;
13                 $LbObj[r][0][i][\mathbf{0}][\mathbf{DP}] \leftarrow \mathcal{M}.objVal()$;
14                 $CurObj[r] \leftarrow \min(CurObj[r], \mathcal{M}.objVal())$;
15   **end**
16   **end**
17   **end**
18 **end**
`// Line 19-21: Search Region (1)`
19 **while** $LbObj[r][1][\phi][\phi][\phi] < CurObj[r]$ **do**
20   Update $LbObj[r][1][\phi][\phi][\phi]$;
21 **end**
22 $CurObj[r] \leftarrow \min(CurObj[r], LbObj[r][1][\phi][\phi][\phi])$;
23 **return** $Obj[r] \leftarrow CurObj[r]$;

tal results, we observe that the optimal objective values are more likely to be obtained in Region(1) and Region(2) for $r \leq 5$ and $r > 5$ respectively. Thus, when solving the problems for more than 5 rounds, we modify the parameter $N_A = [1,2]$ to $N_A = [2,1]$ in Line 6 of Algorithm 3, namely, we preferentially search the feasible region with only 2 active S-boxes at some round.

We obtain the minimum number of active S-boxes for up to 31-round (full) PRESENT. Though our results are the same as those in [16], they didn't prove the results are the exact values since their description for an S-box is rough. In [19], the authors provided the best differential characteristics for 5 to 10 rounds, and good ones for 11 to 15 rounds, while we find the best differential characteristics for up to 23 rounds. Although the weight of the best differential characteristic for 15-round PRESENT is larger than the cipher's block size 64, it can be used to analyze the differential clustering effect [8] of PRESENT, whose clustering effect is very strong as shown in [19, 22].

GIFT-64 The GIFT S-box has 4 differential probabilities: $0, \frac{2}{16}, \frac{4}{16}, \frac{6}{16}$, thus we model the DC-Problem for GIFT-64 using the method in [24]. Similar to the problems for PRESENT, we modify the parameter $N_A = [1,2]$ to $N_A = [2,1]$ in Line 6 of Algorithm 3 when solving the problems for more than 7-round GIFT-64.

We obtain the minimum number of active S-boxes for up to 28(full) rounds, and the best differential characteristics for up to 14 rounds. In [2, 24], the authors provided the good differential characteristics with probabilities $2^{-44.415}$, 2^{-60}, 2^{-64} for 9, 12, 13 rounds respectively, while we find the best ones with probabilities $2^{-42}, 2^{-58}, 2^{-62}$ respectively.

RECTANGLE For RECTANGLE, we obtain the minimum number of active S-boxes for up to 17 rounds, and the best differential characteristics for up to 15 rounds. In our experimental results, the optimal objective values of the problems are obtained when the candidates have only 1 active S-boxes at some rounds. Note that the index of the 1st active S-box has no influence on the optimal objective value of the submodel when searching the smaller feasible region in Region (1) and (2) due to the feature of the round function of RECTANGLE.

LBLOCK and TWINE LBLOCK and TWINE are two similar nibble-oriented lightweight block ciphers, the branches can be cut early by using Method 3 in the search process. We find the best differential characteristics for up to 21-round LBLOCK and 17-round TWINE respectively. The optimal objective values of the problems for both the ciphers are obtained when the candidates have 0 active S-boxes at some round.

6.2 Discussion of the Efficiency of Our Search Algorithm

In Section 6.1, we solved the AS/DC-Problem for a r-round cipher by solving several submodels for shorter rounds. In this section, we discuss the efficiency of our search algorithm in two ways. One is studying the number of the submodels

Table 6: Experimental results of PRESENT and GIFT.

Rounds	PRESENT				GIFT-64			
	#AS	Time(s)	Pr_{max}	Time(s)	#AS	Time(s)	Pr_{max}	Time(s)
1	1	0	2^{-2}	0	1	0	$2^{-1.415}$	0
2	2	0	2^{-4}	0	2	0	$2^{-3.415}$	33
3	4	5	2^{-8}	4	3	2	2^{-7}	129
4	6	5	2^{-12}	12	5	109	$2^{-11.415}$	317
5	10	34	2^{-20}	78	7	84	2^{-17}	701
6	12	3	2^{-24}	399	10	212	$2^{-22.415}$	367
7	14	1	2^{-28}	1	13	66	$2^{-28.415}$	140
8	16	1	2^{-32}	17	16	843	2^{-38}	28992
9	18	1	2^{-36}	3	18	29	2^{-42}	2166
10	20	5	2^{-41}	3102	20	643	2^{-48}	8938
11	22	1	2^{-46}	258	22	66	2^{-52}	25917
12	24	1	2^{-52}	243	24	597	2^{-58}	79773
13	26	1	2^{-56}	197	26	110	2^{-62}	1188
14	28	1	2^{-62}	63383	28	1	2^{-68}	8775
15	30	1	2^{-66}	2799	30	9	-	-
16	32	20	2^{-70}	2756	32	1	-	-
17	34	9	2^{-74}	1473	34	15	-	-
18	36	11	2^{-78}	2855	36	41	-	-
19	38	1	2^{-82}	2	38	2	-	-
20	40	1	2^{-86}	9980	40	2	-	-
21	42	1	2^{-90}	61	42	2	-	-
22	44	2	2^{-96}	3549	44	2	-	-
23	46	2	2^{-100}	37980	46	2	-	-
24	48	20	-	-	48	218	-	-
25	50	2	-	-	50	117	-	-
26	52	2	-	-	52	205	-	-
27	54	2	-	-	54	2	-	-
28	56	2	-	-	56	307	-	-
29	58	2	-	-	-	-	-	-
30	60	2	-	-	-	-	-	-
31	62	2	-	-	-	-	-	-
Total time	0.04h		35.88h		1.02h		36.25h	

Table 7: Experimental results of RECTANGLE, LBLOCK and TWINE

Rounds	RECTANGLE				LBLOCK		TWINE	
	#AS	Time(s)	Pr_{max}	Time(s)	Pr_{max}	Time(s)	Pr_{max}	Time(s)
1	1	0	2^{-2}	0	2^{0}	0	2^{0}	0
2	2	0	2^{-4}	0	2^{-2}	1	2^{-2}	2
3	3	0	2^{-7}	9	2^{-4}	2	2^{-4}	5
4	4	4	2^{-10}	15	2^{-6}	1	2^{-6}	2
5	6	12	2^{-14}	193	2^{-8}	4	2^{-8}	7
6	8	14	2^{-18}	5	2^{-12}	1	2^{-12}	1
7	11	18	2^{-25}	18	2^{-16}	6	2^{-16}	8
8	13	17	2^{-31}	60	2^{-22}	3	2^{-22}	4
9	15	22	2^{-36}	51	2^{-28}	10	2^{-28}	13
10	17	63	2^{-41}	191	2^{-36}	10	2^{-38}	63
11	19	80	2^{-46}	310	2^{-44}	30	2^{-46}	70
12	21	218	2^{-51}	1251	2^{-48}	6	2^{-51}	409
13	23	352	2^{-56}	3478	2^{-56}	734	2^{-58}	4602
14	25	1019	2^{-61}	11324	2^{-62}	598	2^{-64}	10447
15	27	1573	2^{-66}	20319	2^{-66}	587	2^{-68}	15204
16	29	3857	-	-	2^{-72}	2712	2^{-74}	19533
17	31	14214	-	-	2^{-76}	3085	2^{-77}	48351
18	-	-	-	-	2^{-82}	6591	-	-
19	-	-	-	-	2^{-86}	15365	-	-
20	-	-	-	-	2^{-92}	25571	-	-
21	-	-	-	-	2^{-96}	51859	-	-
Total time	5.96h		12.49h		29.77h		27.42h	

Table 8: Experimental number of the submodels solved for solving the AS-Problems for the first 8-round PRESENT by using our algorithm

r	r_1							
	1	2	3	4	5	6	7	8
2	0+0	0+0						
3	2+0	482+0	2+0					
4	0+1	0+2	225+1	3+0				
5	64+0	0+0	160+0	384+0	242+0			
6	0+1	0+2	1+2	1+1	0+0	3+0		
7	0+0	0+0	0+0	0+0	0+0	0+0	2+0	
8	0+0	0+0	0+0	0+0	0+0	0+0	0+0	2+0

The item "$*+*$" denotes the number of $\mathcal{M}'$ adding the number of $\mathcal{M}$.

we solved, and the other is comparing the time between our algorithm and the method of solving original models for the same targets.

In our search algorithm, we solve two types of submodels which are built for $r_1 = 1, 2, ..., r$ rounds:

1. The submodels for calculating lower bounds of the minimum number of active S-boxes, denoted as $\mathcal{M}'$ (submodels used in Method 3).
2. The submodels for solving the AS/DC-Problem, denoted as $\mathcal{M}$.

Taking the AS-Problems for the first 8-round PRESENT as an example, we list the number of the submodels we solved in Table 8. The explanation is:

- For $r = 2$, initialized $LbObj[r]$ and $CurObj[r]$ are equal, thus the program terminated and no submodel was solved.
- For $r = 3$, we solved 2, 482, 2 submodels $\mathcal{M}'$ for $r_1 = 1, 2, 3$ rounds respectively. These values are enough for cutting the smaller feasible regions, thus no submodel $\mathcal{M}$ was solved.
- For $r = 4$, the submodels that have been solved when solving the problems for $r = 2, 3$ rounds will not to be solved again. We solved 225, 3 submodels $\mathcal{M}'$ for $r_1 = 3, 4$ rounds respectively, and 2, 1 submodels $\mathcal{M}$ for $r_1 = 2, 3$ rounds respectively.

From Table 6 and 8, we see that the submodels $\mathcal{M}'$ are solved fast and are useful for cutting the smaller feasible regions. The submodels $\mathcal{M}$ for r_1 rounds are time-consuming, but they are easier solved than the original models for r_1 rounds because some variables in $\mathcal{M}$ are fixed.

When solving the same problems by solving original models, the efficiency is not satisfactory from Table 5. We list the comparison between our algorithm and the method of solving original models in Table 1-3. Although more submodels to be solved using our algorithm, the total time of solving the submodels is less than solving an original model form the tables. In conclusion, our algorithm is more efficient than the method of solving original models.

7 Conclusions

In this paper, we propose a new improved MILP-based search algorithm for security evaluation against differential cryptanalysis by incorporating the idea of divide-and-conquer approach. We firstly divide the whole feasible region into numbers of smaller ones; then we search solutions on these smaller regions; we also use several branch-and-bound techniques to eliminate a large number of impossible branches during the search process to improve the efficiency remarkably; finally the solutions to the smaller regions are combined to give a solution to the original model. As a result, we obtain a more efficient new search algorithm.

We only apply our new method to five lightweight block ciphers. We point out that the permutation layers of these five ciphers are all bit permutations. In future work, we will consider applying our method to ciphers with stronger permutation layer, such as AES [8], NOEKEON [7], SERPENT [4], etc.

In Table 6 and 7, although the weight of the best differential characteristics for some reduced rounds is larger than the cipher block size, we argue that it is possibly useful when the differential clustering is taken into consideration [8].

Due to the duality between differential and linear cryptanalysis [10, 8], the method can be also applied to linear cryptanalysis. Moreover, we can extend our method to related-key differential cryptanalysis [3]. We leave these research as our future work.

References

1. sagemath. http://www.sagemath.org/
2. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. pp. 321–345 (2017)
3. Biham, E.: New types of cryptanalytic attacks using related keys (extended abstract). In: Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. pp. 398–409 (1993)
4. Biham, E., Anderson, R.J., Knudsen, L.R.: Serpent: A new block cipher proposal. In: Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings. pp. 222–238 (1998)
5. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. J. Cryptology **4**(1), 3–72 (1991)
6. 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: Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings. pp. 450–466 (2007)
7. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: Nessie proposal: Noekeon (2000)
8. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
9. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. pp. 386–397 (1993)
10. Matsui, M.: On correlation between the order of s-boxes and the strength of DES. In: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. pp. 366–375 (1994)
11. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers. pp. 57–76 (2011)
12. Optimization: Gurobi: Gurobi optimizer reference manual. http://www.gurobi.com (2013)

13. Sasaki, Y., Todo, Y.: New algorithm for modeling s-box in MILP based differential and division trail search. In: Innovative Security Solutions for Information Technology and Communications - 10th International Conference, SecITC 2017, Bucharest, Romania, June 8-9, 2017, Revised Selected Papers. pp. 150–165 (2017)
14. Siwei Sun, Lei Hu, M.W.P.W.K.Q.X.M.D.S.L.S.K.F.: 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), https://eprint.iacr.org/2014/747
15. Sun, S., Hu, L., Qiao, K., Ma, X., Shan, J., Song, L.: Improvement on the method for automatic differential analysis and its application to two lightweight block ciphers DESL and lblock-s. In: Advances in Information and Computer Security - 10th International Workshop on Security, IWSEC 2015, Nara, Japan, August 26-28, 2015, Proceedings. pp. 97–111 (2015)
16. Sun, S., Hu, L., Song, L., Xie, Y., Wang, P.: Automatic security evaluation of block ciphers with s-bp structures against related-key differential attacks. In: Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers. pp. 39–51 (2013)
17. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. pp. 158–178 (2014)
18. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A lightweight block cipher for multiple platforms. In: Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. pp. 339–354 (2012)
19. Wang, M.: Differential cryptanalysis of reduced-round PRESENT. In: Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings. pp. 40–49 (2008)
20. Wu, S., Wang, M.: Security evaluation against differential cryptanalysis for block cipher structures. IACR Cryptology ePrint Archive **2011**, 551 (2011)
21. Wu, W., Zhang, L.: Lblock: A lightweight block cipher. IACR Cryptology ePrint Archive **2011**, 345 (2011), http://eprint.iacr.org/2011/345
22. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. SCIENCE CHINA Information Sciences **58**(12), 1–15 (2015)
23. Zhang, Y., Sun, S., Cai, J., Hu, L.: Speeding up MILP aided differential characteristic search with matsui's strategy. In: Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings. pp. 101–115 (2018)
24. Zhu, B., Dong, X., Yu, H.: Milp-based differential attack on round-reduced GIFT. IACR Cryptology ePrint Archive **2018**, 390 (2018)

A Examples of the Best Differential Characteristics

We provide the best differential characteristics as follows.

Table 9: The best differential characteristic with probability 2^{-100} for 23-round PRESENT.

Round	Input Difference of S-boxes	Output Difference of S-boxes	Probability
1^{th}	$0x$0000000000070007	$0x$0000000000010001	2^{-4}
2^{th}	$0x$0000000000000011	$0x$0000000000000099	2^{-4}
3^{th}	$0x$0003000000000003	$0x$0001000000000001	2^{-6}
4^{th}	$0x$0000000000001001	$0x$0000000000009009	2^{-4}
5^{th}	$0x$0009000000000009	$0x$0004000000000004	2^{-4}
6^{th}	$0x$0000100100000000	$0x$0000900900000000	2^{-4}
7^{th}	$0x$0900000000000900	$0x$0400000000000400	2^{-4}
8^{th}	$0x$0000400400000000	$0x$0000500500000000	2^{-4}
9^{th}	$0x$0000090000000900	$0x$0000040000000400	2^{-4}
10^{th}	$0x$0000040400000000	$0x$0000050500000000	2^{-4}
11^{th}	$0x$0000050000000500	$0x$0000010000000100	2^{-6}
12^{th}	$0x$0000000000000404	$0x$0000000000000505	2^{-4}
13^{th}	$0x$0000000500000005	$0x$0000000100000001	2^{-6}
14^{th}	$0x$0000000000000101	$0x$0000000000000909	2^{-4}
15^{th}	$0x$0005000000000005	$0x$0001000000000001	2^{-6}
16^{th}	$0x$0000000000001001	$0x$0000000000009009	2^{-4}
17^{th}	$0x$0009000000000009	$0x$0004000000000004	2^{-4}
18^{th}	$0x$0000100100000000	$0x$0000900900000000	2^{-4}
19^{th}	$0x$0900000000000900	$0x$0400000000000400	2^{-4}
20^{th}	$0x$0000400400000000	$0x$0000500500000000	2^{-4}
21^{th}	$0x$0000090000000900	$0x$0000040000000400	2^{-4}
22^{th}	$0x$0000040400000000	$0x$0000050500000000	2^{-4}
23^{th}	$0x$0000050000000500	$0x$00000c0000000c00	2^{-4}

Table 10: The best differential characteristic with probability 2^{-58} for 12-round GIFT

Round	Input Difference of S-boxes	Output Difference of S-boxes	Probability
1^{th}	$0x$d0000000c0000000	$0x$4000000040000000	2^{-4}
2^{th}	$0x$4040000000000000	$0x$5050000000000000	2^{-4}
3^{th}	$0x$5000000050000000	$0x$2000000020000000	2^{-6}
4^{th}	$0x$0000202000000000	$0x$0000505000000000	2^{-4}
5^{th}	$0x$0500000005000000	$0x$0200000002000000	2^{-6}
6^{th}	$0x$2020000000000000	$0x$5050000000000000	2^{-4}
7^{th}	$0x$5000000050000000	$0x$2000000020000000	2^{-6}
8^{th}	$0x$0000202000000000	$0x$0000505000000000	2^{-4}
9^{th}	$0x$0500000005000000	$0x$0200000002000000	2^{-6}
10^{th}	$0x$2020000000000000	$0x$5050000000000000	2^{-4}
11^{th}	$0x$5000000050000000	$0x$8000000080000000	2^{-6}
12^{th}	$0x$0000000000008080	$0x$0000000000003030	2^{-4}

Table 11: The best differential characteristic with probability 2^{-62} for 13-round GIFT

Round	Input Difference of S-boxes	Output Difference of S-boxes	Probability
1^{th}	0x0c0000000c000000	0x0200000002000000	2^{-4}
2^{th}	0x2020000000000000	0x5050000000000000	2^{-4}
3^{th}	0x5000000050000000	0x2000000020000000	2^{-6}
4^{th}	0x0000202000000000	0x0000505000000000	2^{-4}
5^{th}	0x0500000005000000	0x0200000002000000	2^{-6}
6^{th}	0x2020000000000000	0x5050000000000000	2^{-4}
7^{th}	0x5000000050000000	0x2000000020000000	2^{-6}
8^{th}	0x0000202000000000	0x0000505000000000	2^{-4}
9^{th}	0x0500000005000000	0x0200000002000000	2^{-6}
10^{th}	0x2020000000000000	0x5050000000000000	2^{-4}
11^{th}	0x5000000050000000	0x2000000020000000	2^{-6}
12^{th}	0x0000202000000000	0x0000505000000000	2^{-4}
13^{th}	0x0500000005000000	0x0f0000000f000000	2^{-4}

Table 12: The best differential characteristic with probability 2^{-68} for 14-round GIFT.

Round	Input Difference of S-boxes	Output Difference of S-boxes	Probability
1^{th}	0x0600000006000000	0x0200000002000000	2^{-4}
2^{th}	0x2020000000000000	0x5050000000000000	2^{-4}
3^{th}	0x5000000050000000	0x2000000020000000	2^{-6}
4^{th}	0x0000202000000000	0x0000505000000000	2^{-4}
5^{th}	0x0500000005000000	0x0200000002000000	2^{-6}
6^{th}	0x2020000000000000	0x5050000000000000	2^{-4}
7^{th}	0x5000000050000000	0x2000000020000000	2^{-6}
8^{th}	0x0000202000000000	0x0000505000000000	2^{-4}
9^{th}	0x0500000005000000	0x0200000002000000	2^{-6}
10^{th}	0x2020000000000000	0x5050000000000000	2^{-4}
11^{th}	0x5000000050000000	0x2000000020000000	2^{-6}
12^{th}	0x0000202000000000	0x0000505000000000	2^{-4}
13^{th}	0x0500000005000000	0x0200000008000000	2^{-6}
14^{th}	0x2000000000800000	0x5000000000300000	2^{-4}

Table 13: The best differential characteristic with probability 2^{-66} for 15-round RECTANGLE.

Round	Input Difference of Left	Input Difference of right	
1^{th}	$0x$000000000d000030	$0x$0000000002000040	2^{-4}
2^{th}	$0x$0000000000000600	$0x$0000000000000100	2^{-2}
3^{th}	$0x$1000000000000000	$0x$8000000000000000	2^{-3}
4^{th}	$0x$8000000000000000	$0x$3000000000000000	2^{-3}
5^{th}	$0x$0001200000000000	$0x$0008600000000000	2^{-5}
6^{th}	$0x$000c000020000000	$0x$0002000060000000	2^{-5}
7^{th}	$0x$0000000600002000	$0x$0000000200006000	2^{-5}
8^{th}	$0x$2000000000060000	$0x$6000000000020000	2^{-5}
9^{th}	$0x$0000200000000006	$0x$0000600000000002	2^{-5}
10^{th}	$0x$0006000020000000	$0x$0002000060000000	2^{-5}
11^{th}	$0x$0000000600002000	$0x$0000000200006000	2^{-5}
12^{th}	$0x$2000000000060000	$0x$6000000000020000	2^{-5}
13^{th}	$0x$0000200000000006	$0x$0000600000000002	2^{-5}
14^{th}	$0x$0006000020000000	$0x$0002000060000000	2^{-5}
15^{th}	$0x$0000000600002000	$0x$0000000100006000	2^{-4}

Table 14: The best differential characteristic with probability 2^{-96} for 21-round LBLOCK.

Round	Input Difference of S-boxes	Output Difference of S-boxes	Probability
1^{th}	$0x$00000000	$0x$00000000	2^{0}
2^{th}	$0x$01000031	$0x$03000011	2^{-7}
3^{th}	$0x$30000101	$0x$10000201	2^{-6}
4^{th}	$0x$00101001	$0x$00103001	2^{-6}
5^{th}	$0x$00000000	$0x$00000000	2^{0}
6^{th}	$0x$10100100	$0x$a0100200	2^{-6}
7^{th}	$0x$00a12000	$0x$00b11000	2^{-6}
8^{th}	$0x$110a0000	$0x$2a010000	2^{-6}
9^{th}	$0x$00000000	$0x$00000000	2^{0}
10^{th}	$0x$0a000011	$0x$020000a1	2^{-6}
11^{th}	$0x$2000010a	$0x$20000a01	2^{-7}
12^{th}	$0x$0020b00a	$0x$0010200a	2^{-7}
13^{th}	$0x$00000000	$0x$00000000	2^{0}
14^{th}	$0x$20b00a00	$0x$20a00c00	2^{-7}
15^{th}	$0x$002ac000	$0x$00912000	2^{-7}
16^{th}	$0x$b1030000	$0x$c20a0000	2^{-6}
17^{th}	$0x$00000000	$0x$00000000	2^{0}
18^{th}	$0x$030000b1	$0x$01000021	2^{-6}
19^{th}	$0x$10000102	$0x$20000a01	2^{-7}
20^{th}	$0x$00201003	$0x$00101002	2^{-6}
21^{th}	$0x$00000000	$0x$00000000	2^{0}

Table 15: The best differential characteristic with probability 2^{-77} for 17-round TWINE.

Round	Input Difference of S-boxes	Output Difference of S-boxes	Probability
1^{th}	$0x00000000$	$0x00000000$	2^{0}
2^{th}	$0x0000903f$	$0x000080f8$	2^{-7}
3^{th}	$0x00080f08$	$0x00030603$	2^{-6}
4^{th}	$0x09003900$	$0x0800f800$	2^{-6}
5^{th}	$0x00000000$	$0x00000000$	2^{0}
6^{th}	$0x90090300$	$0x70080800$	2^{-8}
7^{th}	$0x78008000$	$0x93003000$	2^{-6}
8^{th}	$0x90a00090$	$0x80700080$	2^{-6}
9^{th}	$0x00000000$	$0x00000000$	2^{0}
10^{th}	$0x0000a099$	$0x00007087$	2^{-7}
11^{th}	$0x00070807$	$0x00090309$	2^{-6}
12^{th}	$0x0a009a00$	$0x07008700$	2^{-6}
13^{th}	$0x00000000$	$0x00000000$	2^{0}
14^{th}	$0xa00a0900$	$0x70070800$	2^{-6}
15^{th}	$0x78007000$	$0x93009000$	2^{-6}
16^{th}	$0x909000a0$	$0x80700070$	2^{-7}
17^{th}	$0x00000000$	$0x00000000$	2^{0}