Automatic Security Evaluation and (Related-key) Differential Characteristic Search : Application to PRESENT-80/128, LBlock, DES(L) and Other Bit-oriented Block Ciphers[⋆] (Draft Version, 2014.4.14)

Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Ling Song

State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
Beijing 100093, China
sunsiwei@iie.ac.cn, hu@is.ac.cn

Abstract. In this paper, we propose two systematic methods to describe the differential property of an S-box with linear inequalities based on logical condition modelling and computational geometry. In one method, inequalities are generated according to some conditional differential properties of the S-box; in the other method, inequalities are extracted from the H-representation of the convex hull of all possible differential patterns of the S-box. For the second method, we develop a greedy algorithm for selecting a given number of inequalities from the convex hull. Using these inequalities combined with Mixed-Integer Linear Programming (MILP) technique, we propose an automatic method for evaluating the security of bit-oriented block ciphers against the (related-key) differential attacks, and several techniques for obtaining tighter security bounds. we successfully prove that 24-round PRESENT-80 is secure enough to resist against standard related-key differential attacks, and the probability of the best related-key differential characteristic of full LBlock is upper bounded by 2^{-60}. These are the tightest security bound with respect to related-key differential attack published so far for PRESENT-80 and LBlock.
Also, we present a new tool for finding (related-key) characteristics automatically for bit-oriented block ciphers. Using this tool, we obtain new related-key characteristics for LBlock, DESL and PRESENT-128, which cover larger number of rounds or have larger probability than all previously known results.
The methodology presented in this paper is generic, automatic and applicable to many bit-oriented block ciphers, including but not limited to PRESENT, EPCBC, LBlock, DES(L), MIBS etc.

Keywords: Related-key differential attack, Active S-box, Mixed-integer Linear Programming, Logical condition modelling, Convex hull

⋆ Source code for generating valid cutting-off inequalities and MILP instances will be made freely available online after publication of the paper.

1 Introduction

Differential cryptanalysis is one of the most well-known attacks on modern block ciphers, based on which a whole bunch of techniques has been developed for analyzing the security of block ciphers, such as truncated differential attack [34], impossible differential attack [9], and boomerang attack [56]. Providing a security evaluation with respect to the differential attack has become a basic requirement for a newly designed practical block cipher to be accepted by the cryptographic community.

Contrary to the single-key model, where methodologies for constructing block ciphers provably resistant to differential attacks are readily available, the understanding of the security of block ciphers with regard to related-key differential attacks is relatively limited. This situation can be seen from the fact that even internationally standardized block ciphers such as AES and PRESENT enjoy no security proof at all for related-key differential attacks at the time of their publication. This limited understanding of the security concerning related-key differential attacks has been greatly improved in recent years for AES-like byte- or word-oriented SPN block ciphers. Along this line of research, two representative papers [10, 23] were published in Eurocrypt 2010 and Crypto 2013. In the former paper [10], an efficient search tool for finding differential characteristics both in the state and in the key was presented, and the best differential characteristics were obtained for some byte-oriented block ciphers such as AES, byte-Camellia, and Khazad. In the latter paper [23], Pierre-Alain Fouque *et al.* showed that the full-round AES-128 can not be proven secure against related-key differential attacks unless the exact coefficients of the MDS matrix and the S-Box differential properties are taken into account. Moreover, a variant of Dijkstra's shortest path algorithm to efficiently find the most efficient related-key attacks on SPN ciphers was developed in [23].

For bit-oriented block ciphers such as PRESENT-80 and DES, Sareh Emami proved that no related-key differential characteristic exists with probability higher than 2^{-64} for PRESENT-80, and therefore PRESENT-80 is secure against basic related-key differential attacks [22]. In [52], Sun *et al.* obtained tigher security bounds for PRESENT-80 with respect to related-key differential attacks using the MILP technique. In [11], Alex Biryukov and Ivica Nikolić proposed two methods based on Matsui's tool for finding related-key differential characteristics for DES-like ciphers. For their methods, they stated that

"... our approaches can be used as well to search for high probability related-key differential characteristics in any bit-oriented ciphers *with linear key schedule.*"

Sareh Emami et al. and Sun *et al.*'s method [22, 52] can not be used to search for actual (related-key) differential characteristics, and Alex Biryukov *et al.*'s method is only applicable to ciphers with linear key schedule.

In this paper, we provide a method based on Mixed-Integer Linear Programming which can not only evaluate the security (obtain security bound) of a block cipher with respect to (related-key) differential attacks, but is also able to search for actual (related-key) differential characteristics even if the key schedule algorithm of the block cipher is nonlinear.

Mixed-Integer Linear Programming (MILP). The problem of Mixed Integer Linear Programming (MILP) is a class of optimization problems derived from Linear Programming in which the aim is to optimize an objective function under certain constraints. The field of MILP has received extensive study and achieved great success in both academic and industrial worlds. A Mixed Integer Linear Programming problem can be formally described as follows.

MILP: Find a vector $x \in \mathbb{Z}^k \times \mathbb{R}^{n-k} \subseteq \mathbb{R}^n$ with $Ax \leq b$, such that the linear function $c_1x_1 + c_2x_2 + \cdots + c_nx_n$ is minimized (or maximized), where $(c_1, \ldots, c_n) \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$.

Despite its intimate relationship with discrete optimization problems, such as the set covering problem, 0-1 knapsack problem, and traveling salesman problem, it is only in recent years that MILP has been explicitly applied in cryptographic research.

In [57], Michael Walter *et al.* modelled the problem of finding a set of variables involved in a system of polynomial equations over $\mathbb{F}_2$, such that when assigned to fixed values, the number of known variables in the system can be maximized as an MILP problem. They have applied this idea in guess-and-determine algebraic attack on the EPCBC block cipher [66], and experimental results showed that this strategy resulted in a faster key recovery attack compared to random assignment. In [17], Julia Borghoff employed two methods, standard conversion [4] and adapted standard conversion [36], to convert the problem of solving a system of polynomial equations into an MILP problem. Martin Albrecht *et al.* [1] treated the problem of recovering cryptographic key material from decayed DRAM as a Partial Weighted Max-Polynomial System Solving Problem which can be solved with the MILP techniques. Bulygin *et al.* studied the invariant coset attack on PRINTcipher by establishing a one-to-one correspondence between defining sets of the invariant projected subsets of PRINTcipher and all feasible solutions of a specific 0-1 integer programming problem [18]. Moreover, MILP was employed in error-tolerant side channel algebraic attacks [46].

In this paper, we are mainly concerned with the application of MILP method in (related-key) differential cryptanalysis. Roughly speaking, differential attack [6] is a cryptanalysis technique used to discover non-random behaviour of a cipher by analyzing the input and output difference of a cipher. A practical approach to evaluate the security of a cipher against differential attack is to determine the lower bound of the number of active S-boxes throughout the cipher. This strategy has been employed in many designs [2, 7, 16, 15, 20]. MILP was applied in automatically determining the lower bounds of the numbers of active S-boxes for some word-oriented symmetric-key ciphers, and therefore used to prove their security against differential cryptanalysis [14, 44, 63] . Laura Winnen and Sun *et al.* [52, 62] extended this method by making it applicable to ciphers involving bit-oriented operations. We notice that such MILP tools [14, 44, 63, 52] for counting the minimum number of active S-boxes are also applied or mentioned in the design and analysis of some authenticated encryption schemes [64, 21, 29–31, 8].

Our Contributions. We find that the constraints presented in [52] are too coarse (and some of these constraints are redundant in some specific case) to accurately describe the differential properties of a specific cipher, since there are a large number of invalid differential patterns of the cipher satisfying all these constraints, which yields a feasible region of the MILP problem much larger than the set of all valid differential characteristics.

In this paper, we propose two methods to tighten the feasible region by cutting off some impossible differential patterns of a specific S-box with linear inequalities: one method is based on logical condition modeling, and the other is a more general approach based on convex hull computation — a fundamental algorithmic problem in computational geometry. In the first method, typically less than 15 inequalities are generated according to some conditional differential properties of the S-box; while in the second method, several hundreds of inequalities are extracted from the H-representation of the convex hull of all possible differential patterns of the S-box.

However, the second approach produces too many inequalities so that adding all of them to an MILP problem will make the solving process impractical. Therefore, we develop a greedy algorithm for selecting a given number of linear inequalities from the convex hull.

By adding all or a part of the constraints generated by these methods, we provide MILP based methods for evaluating the security of a block cipher with respect to (related-key) differential attack, and searching for actual (related-key) differential characteristics. Using these methods, we obtain the following results.

1. The probability of the best related-key differential characteristic of 24-round PRESENT-80, a bit-oriented SPN block cipher, is upper bounded by 2^{-64}, which is the tightest security bound obtained so far for PRESENT-80.

2. The probability of the best related-key differential characteristic for full-round LBlock is at most 2^{-60}. This is the first theoretic result concerning the full LBlock's security against differential attack in the related-key model.

3. We obtain a 14-round related-key differential characteristic of LBlock *in no more than 4 hours on a PC*, which is the best related-key differential characteristic for 14-round LBlock published so far.

4. We obtain an 8-round related-key differential characteristic of DESL *in several minutes on a PC*, to the best of our knowledge, there is no related-key differential characteristic covering more than 7 rounds of DESL have been published before.

5. We obtain a 7-round related-key characteristic for PRESENT-128 with 0 active S-box in its key schedule algorithm, based on which an improved related-key boomerang distinguisher for 14-round PRESENT-128 and a key-recovery attack on 17-round PRESENT-128 can constructed.

Note that the above specific results are not the focus of this paper. Instead, the focus is the new method presented in the paper, which is generic, automatic, and applicable to other lightweight ciphers with bit-oriented operations such as EPCBC [66], and MIBS [28].

Limitations of the Methodology. The methodology presented in this paper has some limitations which we would like to make clear, and trying to overcome these limitations is a topic deserving further investigation.

1. It's very hard to solve the MILP models generated in this paper, since in our models, we introduce a new variable for almost every input/output bit-level difference, which makes the sizes of the MILP instances reasonably large with respect to the number of variables and constraints. Also, there seems to be no method to estimate the computational complexity before actual computation.

2. This methodology is only suitable to evaluate the security of constructions with S-boxes, XOR operations and bit permutations, and can not be applied to block ciphers like SIMON and SPECK [3], which involve modulo addition, and bitwise AND and no S-boxes at all. For tools which can be applied to ARX constructions, we refer the reader to [12, 43, 39–41].

3. As in almost all practical security evaluations of block ciphers with respect to (related-key) differential attack, we assume that the expected differential probability (EDP) π of a characteristic over all keys is (almost) the same as the fixed-key differential probability (DP) π_K for almost all keys (the common hypothesis of *stochastic equivalence* presented in Lai *et al.*'s work on Markov Ciphers [35]), and that if the lower bound of the EDP for any characteristic of a block cipher is less than 2^{-s}, where s is bigger than the block size or key size of the cipher, then the block cipher is secure against (related-key) differential attack, which is a common assumption in almost all works on practical analysis of block ciphers. For more in-depth discussion of the essential gap between EDP π and DP π_K, and what the bounds on EDP actually mean for the security of a block cipher once a key is fixed, we refer the reader to [13] for more information.

Organization of the paper. In Section 2, we introduce Mouha *et al.*'s framework and its extension for counting the number of active S-boxes of PRESENT-like ciphers automatically with the MILP technique. In Sections 3, 4 and 5 we introduce the concept of valid cutting-off inequalities for tightening the feasible region of an MILP problem, and explore how to generate and select valid cutting-off inequalities. We add these inequalities to the overall constraints of the MILP problems describing the differential behaviour of the block ciphers PRESENT-80 and LBlock in Section 6, which enables us to obtain the security bounds for PRESENT-80 and LBlock against related-key differential attacks. We propose several techniques enables us to get tighter security bounds in Section 7. In Section 8, we propose a new tool for finding (related-key) differential characteristics automatically, and apply it to LBlock, DESL, and PRESENT-128.Finally, in Section 9 we conclude the paper and propose some research directions for bit-oriented ciphers and the application of the MILP technique in cryptography.

2 Mouha *et al.*'s Framework and Its Extension

In this section, we present Mouha *et al.*'s framework and its extension for counting the number of differentially active S-boxes for word-oriented and bit-oriented block ciphers respectively.

2.1 Mouha *et al.*'s Framework for Word-oriented Block Ciphers

Assume a cipher is composed of the following three word-oriented operations, where m is the word size:

- XOR, $\oplus : \mathbb{F}_2^{\omega} \times \mathbb{F}_2^{\omega} \to \mathbb{F}_2^{\omega}$
- Linear transformation $L : \mathbb{F}_{2^\omega}^m \to \mathbb{F}_{2^\omega}^m$ with branch number

$$\mathcal{B}_L = \min_{a \neq 0}\{\mathrm{WT}(a||L(a)) : a \in \mathbb{F}_{2^\omega}^m\},$$

  where $\mathrm{WT}(a||L(a))$ is the number of non-zero entries of the $2m$-dimensional vector $a||L(a)$ over the finite field $\mathbb{F}_{2^\omega}$
- S-box, $\mathcal{S} : \mathbb{F}_2^{\omega} \to \mathbb{F}_2^{\omega}$

Mouha *et al.*'s framework uses 0-1 variables, which are subjected to certain constraints imposed by the above operations, to denote the word level differences propagating through the cipher (1 for nonzero difference and 0 for otherwise).

Detailed MILP model building process for determining a lower bound of the number of active S-boxes is described as follows. Firstly, we should include the constraints imposed by the operations of the cipher.

Constraints Imposed by XOR Operations:
Suppose $a \oplus b = c$, where $a,\ b,\ c \in \mathbb{F}_2^{\omega}$ are the input and output differences of the XOR operation, the following constraints will make sure that when a, b, and c are not all zero, then there are at least two of them are nonzero:

$$\begin{cases} a + b + c \geq 2d_{\oplus} \\ d_{\oplus} \geq a \\ d_{\oplus} \geq b \\ d_{\oplus} \geq c \end{cases} \tag{1}$$

where $d_{\oplus}$ is a dummy variable taking values from $\{0, 1\}$. If each one of a, b, and c represents one bit, we should also add the inequalitie $a + b + c \leq 2$.

Constraints Imposed by Linear Transformation:
Let x_{i_k} and $y_{j_k}, k \in \{0, 1, \ldots, m-1\}$, be 0-1 variables denoting the word-level input and output differences of the linear transformation L respectively. Since for nonzero input differences, there are totally at least $\mathcal{B}_L$ nonzero m-bit words in the input and output differences, we include the following constraints:

$$\begin{cases} \sum_{k=0}^{m-1} (x_{i_k} + y_{j_k}) \geq \mathcal{B}_L d_L \\ d_L \geq x_{i_k}, \quad k \in \{0, \ldots, m-1\} \\ d_L \geq y_{j_k}, \quad k \in \{0, \ldots, m-1\} \end{cases} \tag{2}$$

where d_L is a dummy variable taking values in $\{0, 1\}$ and $\mathcal{B}_L$ is the branch number of the linear transformation.

Then, we set up the **objective function** to be the sum of all variables representing the input words of the S-boxes. Obviously, this objective function corresponds to the number of active S-boxes, and can be minimized to determine its lower bound.

Following this approach, the minimum numbers of active S-boxes were obtained in [44] for the r-round ($r \leq 96$) Enocoro-128V2 [58, 59] and full AES ciphers under both single-key and related-key models. We refer the reader to [44] for more information.

2.2 Extension of Mouha's Framework for Bit-oriented Ciphers

For bit-oriented ciphers, bit-level representations and additional constraints are needed [52]. For every input and output bit-level difference, a new 0-1 variable x_i is introduced obeying the following rule of variable assignment

$$x_i = \begin{cases} 1, & \text{for nonzero difference at this bit,} \\ 0, & \text{otherwise.} \end{cases}$$

For every S-box in the schematic diagram, including the encryption process and the key schedule algorithm, we introduce a new 0-1 variable A_j such that

$$A_j = \begin{cases} 1, & \text{if the input word of the Sbox is nonzero,} \\ 0, & \text{otherwise.} \end{cases}$$

At this point, it is natural to choose the objective function f, which will be minimized, as $\sum A_j$ for the goal of determining a lower bound of the number of active S-boxes.

For bit-oriented ciphers, we need to include two sets of constraints. The first one is the set of constraints imposed by XOR operations, and the other is due to the S-box operation. After changing the representations to bit-level, the set of constraints imposed by XOR operations for bit-oriented ciphers are the same as that presented in (1). The S-box operation is more tricky.

Constraints Describing the S-box Operation:

Suppose $(x_{i_0}, \ldots, x_{i_{\omega-1}})$ and $(y_{j_0}, \ldots, y_{j_{\nu-1}})$ are the input and output bit-level differences of an $\omega \times \nu$ S-box marked by A_t. Firstly, to ensure that $A_t = 1$ holds if and only if $x_{i_0}, \ldots, x_{i_{\omega-1}}$ are not all zero, we require that:

$$\begin{cases} A_t - x_{i_k} \geq 0, \quad k \in \{0, \ldots, \omega-1\} \\ x_{i_0} + x_{i_1} + \cdots + x_{i_{\omega-1}} - A_t \geq 0 \end{cases} \tag{3}$$

For bijective S-boxes, nonzero input difference must result in nonzero output difference and vice versa:

$$\begin{cases} \omega y_{j_0} + \omega y_{j_1} + \cdots + \omega y_{j_{\omega-1}} - (x_{i_0} + x_{i_1} + \cdots + x_{i_{\omega-1}}) \geq 0 \\ \nu x_{i_0} + \nu x_{i_1} + \cdots + \nu x_{i_{\omega-1}} - (y_{j_0} + y_{j_1} + \cdots + y_{j_{\nu-1}}) \geq 0 \end{cases} \tag{4}$$

Note that the above constraints should not be used for non-bijective S-box such as the S-box of DES.

Finally, the Hamming weight of the $(\omega + \nu)$-bit word $x_{i_0} \cdots x_{i_{\omega-1}} y_{j_0} \cdots y_{j_{\nu-1}}$ is lower bounded by the branch number $\mathcal{B}_{\mathcal{S}}$ of the S-box for nonzero input difference $x_{i_0} \cdots x_{i_{\omega-1}}$, where $d_{\mathcal{S}}$ is a dummy variable:

$$\begin{cases} \sum\limits_{k=0}^{\omega-1} x_{i_k} + \sum\limits_{k=0}^{\nu-1} y_{j_k} \geq \mathcal{B}_{\mathcal{S}} d_{\mathcal{S}} \\ d_{\mathcal{S}} \geq x_{i_k}, \quad k \in \{0, \ldots, \omega - 1\} \\ d_{\mathcal{S}} \geq y_{j_k}, \quad k \in \{0, \ldots, \omega - 1\} \end{cases} \tag{5}$$

where the branch number $\mathcal{B}_{\mathcal{S}}$ of an S-box $\mathcal{S}$, is defined as

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

and $\mathrm{wt}(\cdot)$ is the standard Hamming weight of a 2ω-bit word. We point out that constraint (5) is redundant for an invertible S-box with branch number $\mathcal{B}_{\mathcal{S}} = 2$, since in this particular case, all differential patterns not satisfying (5) violate (4).

0-1 Variables:

The MILP model proposed above is indeed a Pure Integer Programming Problem since all variables appearing are 0-1 variables. However, in practice we only need to explicitly restrict a part of all variables to be 0-1, while all other variables can be allowed to be any real numbers, which leads to a Mixed-integer Linear Programming problem. Following this approach, the MILP solving process may be accelerated as suggested in [17].

3 Tighten the Feasible Region with Valid Cutting-off Inequalities

The feasible region of an MILP problem is defined as the set of all variable assignments satisfying all constraints in the MILP problem. The modelling process presented in the previous sections indicates that every differential path corresponds to a solution in the feasible region of the MILP problem. However, a feasible solution of the MILP model is not guaranteed to be a valid differential path, since our constraints are far from perfect to rule out all invalid differential patterns. For instance, assume x_i and y_i $(0 \leq i \leq 3)$ are the bit-level input and output differences of the PRESENT-80 S-box (see Table 1). According to Section 2.2, x_i, y_i are subjected to the constraints of (3), (4) and (5). Obviously, $(x_0 \cdots, x_3, y_0, \cdots, y_3) = (1, 0, 0, 1, 1, 0, 1, 1)$ satisfies the above constraints, whereas $0x9 = 1001 \rightarrow 0xB = 1011$ is not a valid difference propagation pattern for the PRESENT S-box, which can be seen from the differential distribution table presented in Table 2. Hence, we are actually trying to minimize the number of active S-boxes over a larger region as illustrated in Fig. 1, and the optimum value obtained in this setting must be smaller than or equal to the actual minimum number of active S-boxes. Although the above fact will not invalidate the lower bound we obtained from our MILP model, this prevents the designers from obtaining tighter security bounds and therefore making better security and efficiency trade-offs.

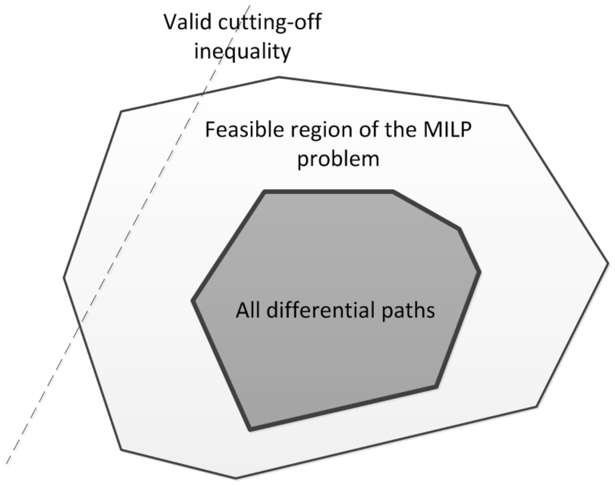
Valid cutting-off inequality
Feasible region of the MILP problem
All differential paths

Fig. 1: The relationship between the set of all differential paths and the feasible region of the MILP problem, and the effect of cutting-off inequality

The situation would be even worse when modelling an invertible S-box with branch number $\mathcal{B}_{\mathcal{S}} = 2$, which is the minimal value of the branch number for an invertible S-box. In the case of invertible S-box with $\mathcal{B}_{\mathcal{S}} = 2$, the constraints of (3), (4) are enough, and (5) is redundant. In this situation, all differential patterns with nonzero input and output differences satisfy the constraints presented in the previous sections, which is obviously too coarse to describe a specific S-box. For instance, all 10 S-boxes of LBlock [65] are invertible and their branch numbers are all 2.

Therefore, we are motivated to look for linear inequalities which can cut off some part of the feasible region of the MILP model while leaving the region of valid differential characteristics intact as illustrated in Fig. 1. For the convenience of discussion, we give the following definition.

Definition 1. *A linear inequality satisfied by all possible valid differential patterns is called a valid cutting-off inequality if it is violated by at least one feasible* *solution corresponding to an impossible differential pattern in the feasible region of the original MILP problem.*

4 Methods for Generating Valid Cutting-off Inequalities

In this section, we present two methods for generating valid cutting-off inequalities by analyzing the differential behaviour of the underlying S-box.

Table 1: The S-box of PRESENT-80

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

Table 2: The Differential Distribution Table of the PRESENT S-box

	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	4	0	0	0	4	0	4	0	0	0	4	0	0
2_x	0	0	0	2	0	4	2	0	0	0	2	0	2	2	2	0
3_x	0	2	0	2	2	0	4	2	0	0	2	2	0	0	0	0
4_x	0	0	0	0	0	4	2	2	0	2	2	0	2	0	2	0
5_x	0	2	0	0	2	0	0	0	0	2	2	2	4	2	0	0
6_x	0	0	2	0	0	0	2	0	2	0	0	4	2	0	0	4
7_x	0	4	2	0	0	0	2	0	2	0	0	0	2	0	0	4
8_x	0	0	0	2	0	0	0	2	0	2	0	4	0	2	0	4
9_x	0	0	2	0	4	0	2	0	2	0	0	0	2	0	4	0
A_x	0	0	2	2	0	4	0	0	2	0	2	0	0	2	2	0
B_x	0	2	0	0	2	0	0	0	4	2	2	2	0	2	0	0
C_x	0	0	2	0	0	4	0	2	2	2	2	0	0	0	2	0
D_x	0	2	4	2	2	0	0	2	0	0	2	2	0	0	0	0
E_x	0	0	2	2	0	0	2	2	2	2	0	0	2	2	0	0
F_x	0	4	0	0	4	0	0	0	0	0	0	0	0	0	4	4

4.1 Modelling Conditional Differential Behaviour with Linear Inequalities

In building integer programming models in practice, sometimes it is possible to model certain logical constraints as linear inequalities. For example, assume x is a continuous variable such that $0 \le x \le M$, where M is a fixed integer, and we know that δ is a 0-1 variable taking value 1 when $x > 0$, that is

$$x > 0 \quad \Rightarrow \quad \delta = 1.$$

It is easy to verify that the above logical condition can be achieved by imposing the following constraint

$$x - M\delta \leq 0.$$

In fact, there is a surprisingly large number of different types of logical conditions can be imposed in a similar way, and we refer the reader to [60, 61] for many other examples.

In this subsection, we take the advantage of this technique to describe the conditional differential behaviour of the PRESENT S-box, which is referred to as undisturbed bits in [54].

Theorem 1. *The S-box of PRESENT-80 has the following properties:*

(i) 1001→???0: If the input difference of the S-box is $0x9 = 1001$*, then the least significant bit of the output difference must be* 0*;*

(ii) 0001→???1 and 1000→???1: If the input difference of the S-box is $0x1 = 0001$ *or* $0x8 = 1000$*, then the least significant bit of the output difference must be* 1*;*

(iii)) ???1→0001 and ???1→0100: If the output difference of the S-box is $0x1 = 0001$ *or* $0x4 = 0100$*, then the least significant bit of the input difference must be* 1*; and*

(iv) ???0→0101: If the output difference of the S-box is $0x5 = 0101$*, then the least significant bit of the input difference must be* 0*.*

Note that similar conditional differential behaviours of other ciphers were also used by other cryptanalysts in different context [24, 32, 38, 33, 18].

Theorem 2. *Let 0-1 variables* (x_0, x_1, x_2, x_3) *and* (y_0, y_1, y_2, y_3) *represent the input and output bit-level differences of the S-box respectively, where* x_3 *and* y_3 *are the least significant bit. Then the logical conditions in Theorem 1 can be described by the following linear inequalities:*

$$-x_0 + x_1 + x_2 - x_3 - y_3 + 2 \geq 0 \tag{6}$$

$$\begin{cases} x_0 + x_1 + x_2 - x_3 - y_3 \geq 0 \\ -x_0 + x_1 + x_2 + x_3 + y_3 \geq 0 \end{cases} \tag{7}$$

$$\begin{cases} x_3 + y_0 + y_1 + y_2 - y_3 \geq 0 \\ x_3 + y_0 - y_1 + y_2 + y_3 \geq 0 \end{cases} \tag{8}$$

$$-x_3 + y_0 - y_1 + y_2 - y_3 + 2 \geq 0 \tag{9}$$

For example, the linear inequality (6) removes all differential patterns of the form $(x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3) = (1, 0, 0, 1, *, *, *, 1)$, where $(x_0, \ldots, x_3)$ and $(y_0, \ldots, y_3)$ are the input and output differences of the PRESENT S-box respectively. We call this group of constraints presented in (6), (7), (8), and (9) the constraints of conditional differential propagation (CDP constraints for short). The CDP constraints obtained from Theorem 1 and the differential patterns removed by these CDP constraints are given in Table 3.

Table 3: Impossible differential patterns removed by the CDP constraints generated according to the differential properties of the PRESENT S-box. Here, a 9-dimensional vector $(\lambda_0, \dots, \lambda_3, \gamma_0, \dots, \gamma_3, \theta)$ in the left column denotes a linear inequality $\lambda_0 x_0 + \cdots + \lambda_3 x_3 + \gamma_0 y_0 + \cdots + \gamma_3 y_3 + \theta \geq 0$, and an 8-dimensional vector in the right column denotes a difference propagation pattern, e.g., $(1, 0, 0, 1, 0, 1, 1, 1)$ denotes $0x9 = 1001 \rightarrow 0x7 = 0111$.

Constraints obtained by logical condition modelling	Impossible differential patterns removed
$(-1, 1, 1, -1, 0, 0, 0, -1, 2)$	(1, 0, 0, 1, 0, 0, 0, 1), (1, 0, 0, 1, 0, 0, 1, 1), (1, 0, 0, 1, 0, 1, 0, 1), (1, 0, 0, 1, 0, 1, 1, 1), (1, 0, 0, 1, 1, 0, 0, 1), (1, 0, 0, 1, 1, 0, 1, 1), (1, 0, 0, 1, 1, 1, 0, 1), (1, 0, 0, 1, 1, 1, 1, 1)
$(1, 1, 1, -1, 0, 0, 0, 1, 0)$	(0, 0, 0, 1, 0, 0, 0, 0), (0, 0, 0, 1, 0, 0, 1, 0), (0, 0, 0, 1, 0, 1, 0, 0), (0, 0, 0, 1, 0, 1, 1, 0), (0, 0, 0, 1, 1, 0, 0, 0), (0, 0, 0, 1, 1, 0, 1, 0), (0, 0, 0, 1, 1, 1, 0, 0), (0, 0, 0, 1, 1, 1, 1, 0)
$(-1, 1, 1, 1, 0, 0, 0, 1, 0)$	(1, 0, 0, 0, 0, 0, 0, 0), (1, 0, 0, 0, 0, 0, 1, 0), (1, 0, 0, 0, 0, 1, 0, 0), (1, 0, 0, 0, 0, 1, 1, 0), (1, 0, 0, 0, 1, 0, 0, 0), (1, 0, 0, 0, 1, 0, 1, 0), (1, 0, 0, 0, 1, 1, 0, 0), (1, 0, 0, 0, 1, 1, 1, 0)
$(0, 0, 0, 1, 1, 1, 1, -1, 0)$	(0, 0, 0, 0, 0, 0, 0, 1), (0, 0, 1, 0, 0, 0, 0, 1), (0, 1, 0, 0, 0, 0, 0, 1), (0, 1, 1, 0, 0, 0, 0, 1), (1, 0, 0, 0, 0, 0, 0, 1), (1, 0, 1, 0, 0, 0, 0, 1), (1, 1, 0, 0, 0, 0, 0, 1), (1, 1, 1, 0, 0, 0, 0, 1)
$(0, 0, 0, 1, 1, -1, 1, 1, 0)$	(0, 0, 0, 0, 0, 1, 0, 0), (0, 0, 1, 0, 0, 1, 0, 0), (0, 1, 0, 0, 0, 1, 0, 0), (0, 1, 1, 0, 0, 1, 0, 0), (1, 0, 0, 0, 0, 1, 0, 0), (1, 0, 1, 0, 0, 1, 0, 0), (1, 1, 0, 0, 0, 1, 0, 0), (1, 1, 1, 0, 0, 1, 0, 0)
$(0, 0, 0, -1, 1, -1, 1, -1, 2)$	(0, 0, 0, 1, 0, 1, 0, 1), (0, 0, 1, 1, 0, 1, 0, 1), (0, 1, 0, 1, 0, 1, 0, 1), (0, 1, 1, 1, 0, 1, 0, 1), (1, 0, 0, 1, 0, 1, 0, 1), (1, 0, 1, 1, 0, 1, 0, 1), (1, 1, 0, 1, 0, 1, 0, 1), (1, 1, 1, 1, 0, 1, 0, 1)

However, there are cases where no such conditional differential property exists. For example, two out of the eight S-boxes of Serpent [5] exhibit no such property. Even when the S-box under consideration can be described with this logical condition modelling technique, the inequalities generated may be not enough to produce a satisfied result. The number of valid cutting-off inequalities can be obtained in this way is given in Table 4 for typical 4×4 S-boxes.

Table 4: The Numbers of valid cutting-off inequalities obtained using different methods. Notations: the "# CDP" columns record the numbers of constraints obtained using logical condition modelling approach, and the "#CH" columns record the numbers of constraints in the H-representation of the convex hulls.

S-box	#CDP	#CH	S-box	#CDP	#CH
Klein [25]	0	312	LBlock S6	12	205
Piccolo [50]	12	202	LBlock S7	12	205
TWINE [53]	0	324	LBlock S8	12	205
PRINCE [16]	0	300	LBlock S9	12	205
MIBS	0	378	Serpent S0	6	327
PRESENT/LED [27]	6	327	Serpent S1	6	327
LBlock S0	12	205	Serpent S2	6	325
LBlock S1	12	205	Serpent S3	0	368
LBlock S2	12	205	Serpent S4	3	321
LBlock S3	12	205	Serpent S5	3	321
LBlock S4	12	205	Serpent S6	3	327
LBlock S5	12	205	Serpent S7	6	368

In the next subsection, a more general approach for generating valid cutting-off inequalities is proposed.

4.2 Convex Hull of All Possible Differentials for an S-box

The convex hull of a set Q of discrete points in $\mathbb{R}^n$ is the smallest convex set that contains Q. A convex hull in $\mathbb{R}^n$ can be described as the common solutions of a set of finitely many linear equations and inequalities as follows:

$$\begin{cases} \lambda_{0,0}x_0 + \cdots + \lambda_{0,n-1}x_{n-1} + \lambda_{0,n} \geq 0 \\ \quad\cdots \\ \gamma_{0,0}x_0 + \cdots + \gamma_{0,n-1}x_{n-1} + \gamma_{0,n} = 0 \\ \quad\cdots \end{cases} \tag{10}$$

This is called the H-Representation of a convex hull. Computing the H-representation of the convex hull of a set of finitely many points is a fundamental algorithm in computation geometry with many applications [26, 49, 47].

If we treat a possible differential of an $\omega \times \nu$ S-box as a point in $\mathbb{R}^{\omega+\nu}$, then we can obtain a set of finitely many discrete points which includes all possible differential patterns of this S-box . For example, one possible differential pattern of PRESENT S-box is $0x9 = 1001 \rightarrow 0xE = 1110$ which is identified with $(1, 0, 0, 1, 1, 1, 1, 0)$, and one possible differential pattern of the DESL S-box is $0x3E = 111110 \rightarrow 0xB = 1011$ which is identified with $(1, 1, 1, 1, 1, 0, 1, 0, 1, 1)$. The set of all possible differential patterns for the S-boxes of PRESENT and DESL are given at `http://paste.ubuntu.com/7123001/` and `http://paste.ubuntu.com/7123005/` respectively. These are essentially sets of finitely many discrete points in high dimensional space, hence we can compute their convex hulls by standard method in computational geometry.

We now define the convex hull of a specific $\omega \times \nu$ S-box to be the set of all linear inequalities in the H-Representation of the convex hull $\mathcal{V}_{\mathcal{S}} \subseteq \mathbb{R}^{\omega+\nu}$ of all possible differential patterns of the S-box. For instance, the Convex Hull of PRESENT S-box can be found in Appendix B. This result is obtained by using the inequality_generator() function in the sage.geometry.polyhedron class of the SAGE computer algebra system [51]. The convex hull of the PRESENT S-box contains 327 linear inequalities. Any one of these inequalities can be taken as a valid cutting-off inequality. The numbers of linear inequalities in the convex hulls of typical 4×4 S-boxes are given in Table 4.

5 Selecting Valid Cutting-off Inequalities from the Convex Hull: A Greedy Approach

The number of equations and inequalities in the H-Representation of a convex hull computed from a set of discrete points in n dimensional space is very large in general. For instance, the convex hull $\mathcal{V}_{\mathcal{S}} \subseteq \mathbb{R}^8$ of a 4×4 S-box typically involves several hundreds of linear inequalities. Adding all of them to an MILP problem will quickly make the MILP problem insolvable in practical time. Hence, it is necessary to select a small number, say n, of "best" inequalities from the convex hull. Here by "best" we mean that, among all possible selections of n inequalities, the selected ones maximize the number of removed impossible differentials. Obviously, this is a hard combinatorial optimization problem. Therefore, we design a greedy algorithm, listed in Algorithm 1, to approximate the optimum selection.

Algorithm 1: Selecting n inequalities from the convex hull $\mathcal{H}$ of an S-box

```
   Input:
   $\mathcal{H}$: the set of all inequalities in the H-representation of the convex hull of
   an S-box;
   $\mathcal{X}$: the set of all possible differential patterns of an S-box;
   $n$: a positive integer.
   Output: $\mathcal{O}$: a set of $n$ inequalities selected from $\mathcal{H}$
 1 $l^* :=$ None;
 2 $\mathcal{X}^* := \mathcal{X}$;
 3 $\mathcal{H}^* := \mathcal{H}$;
 4 $\mathcal{O} := \emptyset$;
 5 for $i \in \{0, \dots, n-1\}$ do
 6     $l^* :=$ The inequality in $\mathcal{H}^*$ which maximizes the number of removed
       impossible differential patterns from $\mathcal{X}^*$ ;
 7     $\mathcal{X}^* := \mathcal{X}^* - \{$removed impossible differential patterns by $l^*\}$;
 8     $\mathcal{H}^* := \mathcal{H}^* - \{l^*\}$;
 9     $\mathcal{O} := \mathcal{O} \cup \{l^*\}$;
10 end
11 return $\mathcal{O}$
```

The algorithm builds up a set of valid cutting-off inequalities by selecting at each step an inequality from the convex hull which maximizes the number of removed impossible differential patterns from the current feasible region.

We select 6 valid cutting-off inequalities from the convex hull of the PRESENT S-box using Algorithm 1. These inequalities and the impossible differential patterns removed are listed in Table 5. Compared with the 6 valid cutting-off inequalities obtained in Theorem 1 (see Table 3), they cut off $66 - 42 = 24$ more impossible differential patterns, which leads to a relatively tighter feasible region.

6 Application to PRESENT-80, LBlock, and DES

In this section, we apply our method to two block ciphers with different structures. One is the bit-oriented SPN block cipher PRESENT-80, and the other is the bit-oriented Feistel block cipher LBlock.

6.1 The 24-round Reduced PRESENT-80 is Secure Against Basic Related-key Differential Attack

We have applied the method presented in previous sections to the block cipher PRESENT-80 to determine its security bound with respect to the related-key differential attack. A Python module [55] is developed to generate the MILP instances in "lp" format [19]. In each of these MILP models, we include one more constraint to ensure that the difference of the initial key register is nonzero, since the case where the difference of the initial key register is zero can by analyzed

Table 5: Impossible differential patterns removed by the constraints selected from the convex hull of the PRESENT S-box

Constraints selected from the convex hull by the greedy algorithm	Impossible differential patterns removed
$(-2, 1, 1, 3, 1, -1, 1, 2, 0)$	(1, 0, 1, 0, 0, 1, 0, 0) (1, 0, 0, 0, 1, 1, 0, 0) (1, 0, 0, 0, 1, 0, 0, 0) (1, 0, 1, 0, 0, 1, 1, 0) (1, 0, 0, 0, 1, 1, 1, 0) (1, 1, 0, 0, 0, 1, 0, 0) (1, 1, 0, 0, 0, 1, 1, 0) (1, 0, 0, 0, 0, 1, 1, 0) (1, 0, 1, 0, 1, 1, 0, 0) (1, 0, 0, 0, 0, 1, 0, 0) (1, 0, 0, 0, 0, 1, 0, 1) (1, 0, 0, 0, 0, 0, 1, 0) (1, 1, 0, 0, 1, 1, 0, 0) (1, 1, 1, 0, 0, 1, 0, 0)
$(1, -2, -3, -2, 1, -4, 3, -3, 10)$	(0, 1, 1, 0, 1, 1, 0, 1) (1, 1, 1, 0, 0, 1, 0, 1) (0, 1, 1, 1, 0, 1, 1, 1) (1, 0, 1, 1, 0, 1, 0, 1) (0, 1, 1, 0, 0, 1, 0, 1) (0, 1, 1, 1, 0, 1, 0, 0) (0, 1, 1, 1, 0, 1, 0, 1) (1, 1, 1, 1, 1, 1, 0, 1) (0, 0, 1, 1, 0, 1, 0, 1) (0, 1, 1, 1, 1, 1, 0, 1) (1, 1, 1, 1, 0, 1, 0, 1) (0, 1, 0, 1, 0, 1, 0, 1) (0, 0, 1, 1, 1, 1, 0, 1)
$(2, -2, 3, -4, -1, -4, -4, 1, 11)$	(0, 1, 0, 1, 0, 1, 1, 0) (1, 1, 0, 1, 0, 1, 1, 0) (0, 0, 0, 1, 1, 1, 1, 0) (0, 1, 0, 1, 0, 1, 1, 1) (0, 0, 0, 1, 1, 1, 1, 1) (0, 1, 0, 1, 1, 1, 1, 1) (0, 1, 0, 1, 1, 1, 1, 0) (0, 0, 0, 1, 0, 1, 1, 0) (1, 1, 0, 1, 1, 1, 1, 0) (0, 1, 1, 1, 1, 1, 1, 0) (1, 1, 0, 1, 1, 1, 1, 1)
$(-1, -2, -2, -1, -1, 2, -1, 0, 6)$	(1, 1, 1, 0, 1, 0, 1, 1) (1, 1, 1, 0, 1, 0, 1, 0) (1, 1, 1, 1, 1, 0, 0, 1) (1, 1, 1, 1, 1, 0, 0, 0) (0, 1, 1, 1, 1, 0, 1, 1) (1, 1, 1, 1, 1, 0, 1, 0) (0, 1, 1, 1, 1, 0, 1, 0) (1, 1, 1, 1, 0, 0, 1, 1) (1, 1, 1, 1, 1, 0, 1, 1) (1, 1, 1, 1, 0, 0, 1, 0)
(-2, 1, -2, -1, 1, -1, -2, 0, 6)	(1, 1, 1, 1, 0, 1, 1, 0) (1, 1, 1, 1, 0, 1, 1, 1) (1, 0, 1, 1, 0, 0, 1, 0) (1, 0, 1, 0, 0, 1, 1, 1) (1, 0, 1, 1, 0, 0, 1, 1) (1, 0, 1, 1, 1, 1, 1, 0) (1, 0, 1, 1, 1, 1, 1, 1) (1, 0, 1, 1, 0, 1, 1, 1) (1, 0, 1, 1, 0, 1, 1, 0)
$(2, 1, 1, -3, 1, 2, 1, 2, 0)$	(0, 0, 0, 1, 1, 0, 0, 0) (0, 0, 1, 1, 0, 0, 1, 0) (0, 0, 0, 1, 0, 0, 0, 1) (0, 1, 0, 1, 1, 0, 0, 0) (0, 0, 0, 1, 0, 1, 0, 0) (0, 0, 0, 1, 0, 0, 1, 0) (0, 0, 1, 1, 1, 0, 0, 0) (0, 1, 0, 1, 0, 0, 1, 0) (0, 0, 0, 1, 1, 0, 1, 0)

in the single-key model. Then we employ the Gurobi 5.5 optimizer [45] to solve the MILP instances.

By default the computations are performed on a PC using 4 threads with Intel(R) Core(TM) Quad CPU (2.83GHz, 3.25GB RAM, Windows XP), and a star "*" is appended on a timing data to mark that the corresponding computation is taken on a workstation equipped with two Intel(R) Xeon(R) E5620 CPU(2.4GHz, 8GB RAM, 8 cores). Despite there are only 2 CPUs and 8 physical cores in total on the workstation, we fire up 16 threads in Gurobi5.5 to solve the corresponding MILP instances to exploit Intel's Hyper-Threading Technology [42], where for each physical core, the operating system simulates two virtual or logical cores, and shares the workload between them.

We have computed the number of active S-boxes for PRESENT-80 in the related-key model up to 14 rounds, and the results are summarized in Table 6 and Table 7, where the "#Constraints" columns record the number of constraints imposed and the "#Variables" columns show the number of 0-1 variables and continuous variables in the underlying MILP instance. For example, according to the first row of Table 6, there are 97 0-1 variables and 277 continuous variables in the MILP instance corresponding to 1-round PRESENT-80, and the Gurobi optimizer find the minimum number of active S-boxes is 0 in no more than 1 second.

Table 6: MILP models for PRESENT-80 with CDP constraints added

Rounds	#Variables	#Constraints	#Active S-boxes	Time (in seconds)
1	$97 + 277$	632	0	1
2	$130 + 474$	1262	0	1
3	$163 + 671$	1892	1	1
4	$196 + 868$	2522	2	1
5	$229 + 1065$	3152	3	5
6	$262 + 1262$	3782	5	16
7	$295 + 1459$	4412	7	107
8	$328 + 1656$	5042	9	254
9	$361 + 1853$	5672	10	522
10	$394 + 2050$	6302	13	4158
11	$427 + 2247$	6932	15	18124
12	$460 + 2444$	7562	16	50017
13	$493 + 2641$	8192	18	137160*
14	$526 + 2838$	8822	20	1316808*
15	$559 + 3035$	9452	–	$>$ 20days

Note that there are $(17 \times 6)r$ more constraints of the rth row(corresponding to r rounds) of Table 6 than that of Table 7. This is due to the fact that for every S-box there are 6 more constraints (see Theorem 1) in the MILP instance with CDP constraints included, and for every round there are 17 S-boxes: 16 in

Table 7: MILP models for PRESENT-80 without CDP constraints

Rounds	#Variables	#Constraints	#Active S-boxes	Time (in seconds)
1	$97 + 277$	530	0	1
2	$130 + 474$	1058	0	1
3	$163 + 671$	1586	1	1
4	$196 + 868$	2114	2	1
5	$229 + 1065$	2642	3	3
6	$262 + 1262$	3170	4	10
7	$295 + 1459$	3698	6	26
8	$328 + 1656$	4226	8	111
9	$361 + 1853$	4754	9	171
10	$394 + 2050$	5282	12	1540
11	$427 + 2247$	5810	13	8136
12	$460 + 2444$	6338	15	18102
13	$493 + 2641$	6866	17	49537*
14	$526 + 2838$	7394	18	685372*
15	$559 + 3035$	7922	–	$>$ 20days

the encryption process (Appendix A, Fig. 2) and 1 in the key schedule algorithm (Appendix A, Fig. 3).

These results clearly demonstrate that the MILP models with CDP constraints lead to tighter security bounds. In particular, we have proved that there are at least 16 active S-boxes in the best related-key differential characteristic for any consecutive 12-rounds of PRESENT-80. Therefore, the probability of the best related-key differential characteristic of 24-round PRESENT-80 is $(2^{-2})^{16} \times (2^{-2})^{16} = 2^{-64}$, leading to the result that the 24-round PRESENT-80 is resistant to basic related-key differential attack based on related-key differential characteristic (rather than differential).

For round reduced variants of PRESENT-80 with round $r \geq 15$, we are unable to accomplish the computation within 20 days.

It is possible to get tighter security bounds by adding more constraints: experimental result shows that, by adding 6 more valid cutting-off inequalities listed in Table 5 to the MILP problems for each S-box appearing in the schematic representation of PRESENT-80, we are able to prove that the guaranteed number of active S-boxes in related-key model for 7-round PRESENT-80 is at least 8, which is the tightest bound obtained so far (see Table 6 and Table 7 for comparison).

6.2 Results on LBlock

Up to now, there is no concrete result concerning the security of full-round LBlock [65] against differential attack in the related-key model due to a lack of proper tools for bit-oriented designs.

Since the encryption process of LBlock is nibble-oriented, the security of LBlock against single-key differential attack can be evaluated by those word-oriented techniques. However, the "$\lll 29$" operations in the key schedule algorithm of LBlock destroy its overall nibble-oriented structure, and make those word-oriented approaches infeasible in evaluating the security of LBlock against related-key differential attacks.

In this subsection, we apply the method proposed in this paper to LBlock, some results concerning its security against related-key differential attacks are obtained. The valid cutting-off inequalities used to obtain these results are listed in Appendix C. Note that the type of constraints given in (5) are removed in our MILP models for LBlock according to the explanations presented in previous sections.

From Table 8, we can deduce that the probability of the best differential characteristic for full LBlock (totally $32 = 11+11+10$ rounds) is upper bounded by $(2^{-2})^{10}\times(2^{-2})^{10}\times(2^{-2})^{8} = 2^{-56}$, where 2^{-2} is the best differential probability for a single S-box of LBlock. To the best of our knowledge, this is the first result concerning the security of the full-round LBlock against related-key differential attacks.

In fact, here we have an implicit trade-off between the number of constraints we use and the number of rounds we analyze. For example, we can use less constraints for every S-box and try to analyze more rounds, or we can use more constraints and focus on less rounds (but stronger bounds). However, it is not a simple task to find the best trade-off due to our limited computational power, and preliminary experiments show that we cannot get more interesting results for full LBlock by pursuing this direction. In section 7, we will propose other techniques for getting tighter security bounds.

Table 8: Results for related-key differential analysis on LBlock.

Rounds	#Variables	#Constraints	#Active S-boxes	Time (in seconds)
1	218+104	660	0	1
2	292+208	1319	0	1
3	366+312	1978	0	1
4	440+416	2637	0	1
5	514+520	3296	1	2
6	588+624	3955	2	12
7	662+728	4614	3	38
8	736+832	5273	5	128
9	810+936	5932	6	386
10	884+1040	6591	8	19932
11	958+1144	7250	10	43793

7 Techniques for Getting Tighter Security Bounds

In our analysis, if we can show that any t-round characteristic has at least N_t active S-boxes by employing the tools presented in this paper, then we argue that the probability of the best t-round characteristic is upper bounded by ϵ^{N_t}, where ϵ is the maximum differential probability of the underlying S-boxes. However, it is unlikely that all the active S-boxes take the maximum differential probability ϵ. Therefore, we have the following strategy for obtaining tighter security bound for a t-round characteristic.

Firstly, compute the set $\mathcal{E}$ of all the differential patterns of an S-box with probabilities greater or equal to the S-box's maximum differential probability ϵ.

Secondly, compute the H-representation $H_{\mathcal{E}}$ of the convex hull of $\mathcal{E}$, and then use the inequalities selected from $H_{\mathcal{E}}$ by algorithm 1 to generate a t-round model according to section 2 and 3. Note that the feasible region of this model is smaller than that of a t-round model generated in standard way, since the differential patterns allowed to take in this model is more restrictive. Hence, we are hopefully to get a larger objective value than N_t.

Finally, Solve the model using a software optimizer. If the objective value is greater than N_t, we know that there is no differential characteristic with N_t active S-boxes such that all these S-boxes take differential patterns with probability ϵ. And hence, we can conclude that there is at least one active S-box taking a differential pattern with probability greater than ϵ in a t-round characteristic with only N_t active S-boxes. Using the above strategy, we have proved the following theorem/observation/fact.

Theorem 3. *there are at least 13 active S-boxes in a* 13*-round related-key differential characteristic of LBlock, and there is at least one active S-box taking a differential pattern with probability* 2^{-3} *in any* 13*-round related-key differential characteristic of LBlock with only 13 active S-boxes. Therefore, the probability of a* 13*-round related-key differential characteristic of LBlock is upper bounded by* $(2^{-2})^{12} \times (2^{-3}) = 2^{-27}$.

Yet another technique for obtaining tighter security bound is inspired by Alex Biryukov and Sareh Emami *et al.*'s (extended) split approach [11, 22]. In Sun *et al.*'s work, the strategy for proving an n-round iterative cipher's security against related-key differential attacks is as follows. By employing the MILP technique, compute the minimum number N_t of differentially active S-boxes for any consecutive t-round ($1 \leq t \leq n$) related-key differential characteristic. Then the lower bound of the number of active S-boxes for the full cipher (n-round) can be obtained by computing

$$\sum_{j \in I \subseteq \{1,2,\dots\}} N_{t_j}, \text{ where } \sum_{j \in I} t_j = n.$$

Note that the computational cost is too high to compute N_n directly.

For example, to evaluate the security of full LBlock (32-round) with respect to related-key differential attacks, We use the lower bounds $N_{11} = 10$ and $N_{10} = 8$ to deduce that there are at least $N_{11} + N_{11} + N_{10} = 10 + 10 + 8 = 28$ active S-boxes for any related-key differential characteristic of full LBlock.

We point out that this simple "split strategy" can be improved to obtain tighter security bound by exploiting more information of a differential characteristic. The main idea is that the characteristic of round 1 to round m and the characteristic of round $m+1$ to round $2m$ should not be treated equal although they have the same number of rounds, since the starting difference of a characteristic of round $m+1$ to m is not as free as that of a characteristic of round 1 to round m.

By adding the constraint that the number of active S-boxes of any characteristic covering round 22 to round 26 (5 rounds in total) has at least 1 active S-box (see table 8), and at most 12 active S-boxes to a 11-round (round 22 to round 32) MILP model, we can show that there are at least 3 active S-boxes in a characteristic covering round 27 to round 32 (6 rounds in total). Combined with Theorem X, we have that the probability of the best related-key differential characteristic for full LBlock is upper bounded by $2^{-27} \times 2^{-27} \times (2^{-2})^3 = 2^{-60}$, which is a tighter security bound for the full LBlock.

8 A Heuristic Method for Finding (Related-key) Differential Characteristics Automatically based on MILP Technique

To find a (related-key) differential characteristic with relatively high probability covering r rounds of a cipher is the most important step in (related-key) differential cryptanalysis. Most of the tools for searching differential characteristics are essentially based on Matsui's algorithm. In this section, we propose an MILP based heuristic method for finding (related-key) differential characteristics. Compared to other methods, our method is easier to implement, and more flexible.

Thanks to the valid cutting-off inequalities which can describe the property of an S-box according to its differential distribution table, our method can output a good (related-key) differential characteristic directly by employing the MILP technique. The procedure of our method is outlined as follows.

Step 1. For every S-box $\mathcal{S}$, select n inequalities from the convex hull of the set of all possible differential patterns of $\mathcal{S}$ using Algorithm 1, and generate an r-round MILP problem in which we require that *all variables involved are 0-1.*

Step 2. Extract a feasible solution of the MILP model by using the Gurobi [45] optimizer.

Step 3. Check whether the feasible solution is a valid (related-key) differential characteristic. If it is a valid characteristic, we are done. Otherwise, go to step 1, increase the number of selected inequalities from the convex hulls, and repeat the whole process.

We have developed a software by employing the python interface provided by the Gurobi optimizer, which automates the whole process of the above method. To demonstrate the practicability of our method, we apply it to the the block cipher LBlock and DESL [37].

8.1 Related-key Characteristics for LBlock, DESL, and PRESENT-128

We find a 14-round related key differential characteristic with only 20 active S-boxes for LBlock in 4 hours, and related-key characteristics for 8- and 9-round DESL. To the best of our knowledge, these are the best related-key differential characteristics have been obtained so far. We give the concrete results for 14-round LBlock, 8-round DESL and 7-round PRESENT-128 in Appendix.

9 Conclusion and Directions for Future Work

In this paper, we bring new constraints into Mouha *et al.*'s framework to describe the differential properties of a specific S-box, and therefore obtain a more accurate mixed integer programming model for the differential behavior of a block cipher.

Based these constraints. we propose an automatic method for evaluating the security of bit-oriented block ciphers with respect to (related-key) differential attack. We also present a new tool for finding (related-key) characteristics automatically. By using these methods, we obtained tighter security bounds for some bit-oriented block ciphers, and related-key characteristics which cover larger number of rounds or have larger probabilities than all previously known results.

At this point, several open problems emerged. Firstly, we have observed that the MILP instances derived from such cryptographic problems are very hard to solve compared with general MILP problems with the same scale with respect to the numbers of variables and constraints. Hence, it is interesting to develop specific methods to accelerate the solving process of such problems and therefore increase the number of rounds of the cipher under consideration can be dealt with. Secondly, the method presented in this paper is very general, is it possible to develop a compiler which can convert a standard description, say a description using hardware description language [48], of a cipher into an MILP instance to automate the entire security evaluation cycle with respect to (related-key) differential attack?

References

1. Albrecht, M., Cid, C.: Cold boot key recovery by solving polynomial systems with noise. In: Applied Cryptography and Network Security. pp. 57–72. Springer (2011)
2. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platformsdesign andanalysis. In: Selected Areas in Cryptography. pp. 39–56. Springer (2001)
3. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), `http://eprint.iacr.org/2013/404`
4. Beigel, R.: The polynomial method in circuit complexity. In: Structure in Complexity Theory Conference, 1993., Proceedings of the Eighth Annual. pp. 82–95. IEEE (1993)

5. Biham, E., Anderson, R., Knudsen, L.: Serpent: A new block cipher proposal. In: Fast Software Encryption. pp. 222–238. Springer (1998)
6. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of CRYPTOLOGY 4(1), 3–72 (1991)
7. Bilgin, B., Bogdanov, A., Knežević, M., Mendel, F., Wang, Q.: Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In: Cryptographic Hardware and Embedded Systems-CHES 2013, pp. 142–158. Springer (2013)
8. Bilgin, B., Bogdanov, A., Knežević, M., Mendel, F., Wang, Q.: Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In: Cryptographic Hardware and Embedded Systems-CHES 2013, pp. 142–158. Springer (2013)
9. Biryukov, A.: Impossible differential attack. In: Encyclopedia of Cryptography and Security, pp. 597–597. Springer (2011)
10. Biryukov, A., Nikolić, I.: Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others. In: Advances in Cryptology–EUROCRYPT 2010, pp. 322–344. Springer (2010)
11. Biryukov, A., Nikolić, I.: Search for related-key differential characteristics in des-like ciphers. In: Fast Software Encryption. pp. 18–34. Springer (2011)
12. Biryukov, A., Velichkov, V.: Automatic search for differential trails in arx ciphers. In: Topics in Cryptology–CT-RSA 2014, pp. 227–250. Springer (2014)
13. Blondeau, C., Bogdanov, A., Leander, G.: Bounds in shallows and in miseries. In: Advances in Cryptology–CRYPTO 2013, pp. 204–221. Springer (2013)
14. Bogdanov, A.: On unbalanced feistel networks with contracting mds diffusion. Designs, Codes and Cryptography 59(1-3), 35–58 (2011)
15. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems-CHES 2007, pp. 450–466. Springer (2007)
16. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al.: Prince–a low-latency block cipher for pervasive computing applications. In: Advances in Cryptology–ASIACRYPT 2012, pp. 208–225. Springer (2012)
17. Borghoff, J., Knudsen, L.R., Stolpe, M.: Bivium as a mixed-integer linear programming problem. In: Cryptography and Coding, pp. 133–152. Springer (2009)
18. Bulygin, S., Walter, M.: Study of the invariant coset attack on printcipher: more weak keys with practical key recovery. Tech. rep., Cryptology eprint Archive, Report 2012/85 (2012)
19. CPLEX, I.I.: Ibm software group. User-Manual CPLEX 12 (2011)
20. Daemen, J., Rijmen, V., Proposal, A.: Rijndael. In: Proceedings from the First Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology (NIST) (1998)
21. Elif Bilge Kavun, Martin M. Lauridsen, G.L.C.R.P.S.T.Y.: Pr∅st v1. CAESAR submission (2014), `http://competitions.cr.yp.to/round1/proestv1.pdf`
22. Emami, S., Ling, S., Nikolic, I., Pieprzyk, J., Wang, H.: The resistance of PRESENT-80 against related-key differential attacks. Cryptology ePrint Archive, Report 2013/522 (2013), `http://eprint.iacr.org/`
23. Fouque, P.A., Jean, J., Peyrin, T.: Structural evaluation of AES and chosen-key distinguisher of 9-round aes-128. In: Canetti, R., Garay, J. (eds.) Advances

in Cryptology CRYPTO 2013, Lecture Notes in Computer Science, vol. 8042, pp. 183–203. Springer Berlin Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-40041-4_11

24. Fuhr, T.: Finding second preimages of short messages for hamsi-256. In: Advances in Cryptology-ASIACRYPT 2010, pp. 20–37. Springer (2010)
25. Gong, Z., Nikova, S., Law, Y.W.: Klein: a new family of lightweight block ciphers. In: RFID. Security and Privacy, pp. 1–18. Springer (2012)
26. Goodman, J.E., O'Rourke, J.: Handbook of discrete and computational geometry. CRC press (2010)
27. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The led block cipher. In: Cryptographic Hardware and Embedded Systems–CHES 2011, pp. 326–341. Springer (2011)
28. Izadi, M., Sadeghiyan, B., Sadeghian, S.S., Khanooki, H.A.: Mibs: a new lightweight block cipher. In: Cryptology and Network Security, pp. 334–348. Springer (2009)
29. Jérémy Jean, Ivica Nikolić, T.P.: Deoxys v1. CAESAR submission (2014), http://competitions.cr.yp.to/round1/deoxysv1.pdf
30. Jérémy Jean, Ivica Nikolić, T.P.: Joltik v1. CAESAR submission (2014), http://competitions.cr.yp.to/round1/joltikv1.pdf
31. Jérémy Jean, Ivica Nikolić, T.P.: Kiasu v1. CAESAR submission (2014), http://competitions.cr.yp.to/round1/kiasuv1.pdf
32. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of nlfsr-based cryptosystems. In: Advances in Cryptology-ASIACRYPT 2010, pp. 130–145. Springer (2010)
33. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of trivium and katan. In: Selected Areas in Cryptography. pp. 200–212. Springer (2012)
34. Knudsen, L.R.: Truncated and higher order differentials. In: Fast Software Encryption. pp. 196–211. Springer (1995)
35. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Advances in CryptologyEUROCRYPT91. pp. 17–38. Springer (1991)
36. Lamberger, M., Nad, T., Rijmen, V.: Numerical solvers and cryptanalysis. Journal of mathematical cryptology 3(3), 249–263 (2009)
37. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New lightweight des variants. In: Fast Software Encryption. pp. 196–210. Springer (2007)
38. Lehmann, M., Meier, W.: Conditional differential cryptanalysis of grain-128a. In: Cryptology and Network Security, pp. 1–11. Springer (2012)
39. Leurent, G.: Construction of differential characteristics in arx designs application to skein. In: Advances in Cryptology–CRYPTO 2013, pp. 241–258. Springer (2013)
40. Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Fast Software Encryption. pp. 336–350. Springer (2002)
41. Lipmaa, H., Wallén, J., Dumas, P.: On the additive differential probability of exclusive-or. In: Fast Software Encryption. pp. 317–331. Springer (2004)
42. Marr, D.T., Binns, F., Hill, D.L., Hinton, G., Koufaty, D.A., Miller, J.A., Upton, M.: Hyper-threading technology architecture and microarchitecture. Intel Technology Journal 6(1) (2002)
43. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for arx: Application to salsa20. Cryptology ePrint Archive, Report 2013/328 (2013), http://eprint.iacr.org/2013/328

44. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Information Security and Cryptology. pp. 57–76. Springer (2012)
45. Optimization, G.: Gurobi optimizer reference manual. URL: http://www. gurobi. com (2013)
46. Oren, Y., Kirschbaum, M., Popp, T., Wool, A.: Algebraic side-channel analysis in the presence of errors. In: Cryptographic Hardware and Embedded Systems, CHES 2010, pp. 428–442. Springer (2010)
47. o'Rourke, J.: Computational geometry in C. Cambridge university press (1998)
48. Pedroni, V.A.: Circuit design with VHDL. The MIT Press (2004)
49. Preparata, F.P., Shamos, M.I.: Computational geometry: An introduction (monographs in computer science). Monographs in Computer Science (Springer-Verlag, New York, 1985), ISBN 3540961313 (1993)
50. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: an ultra-lightweight blockcipher. In: Cryptographic Hardware and Embedded Systems–CHES 2011, pp. 342–357. Springer (2011)
51. Stein, W., et al.: Sage: Open source mathematical software (2008)
52. 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. Cryptology ePrint Archive, Report 2013/547 (2013), `http://eprint.iacr.org/`
53. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: Twine: A lightweight, versatile block cipher. In: ECRYPT Workshop on Lightweight Cryptography. pp. 146–169 (2011)
54. Tezcan, C.: Improbable differential attack on present using undisturbed bits. In: International Conference on Applied and Computational Mathematics (2013)
55. Van Rossum, G., et al.: Python programming language. In: USENIX Annual Technical Conference (2007)
56. Wagner, D.: The boomerang attack. In: Fast Software Encryption. pp. 156–170. Springer (1999)
57. Walter, M., Bulygin, S., Buchmann, J.: Optimizing guessing strategies for algebraic cryptanalysis with applications to EPCBC. In: The 8th China International Conference on Information Security and Cryptology (Inscrypt 2012). Springer (2012)
58. Watanabe, D., Okamoto, K., Kaneko, T.: A hardware-oriented light weight pseudo-random number generator enocoro-128v2. In: The Symposium on Cryptography and Information Security. pp. 3D1–3 (2010)
59. Watanabe, D., Owada, T., Okamoto, K., Igarashi, Y., Kaneko, T.: Update on enocoro stream cipher. In: Information Theory and its Applications (ISITA), 2010 International Symposium on. pp. 778–783. IEEE (2010)
60. Williams, H.P.: Logical problems and integer programming. Bulletin of the Institute of Mathematics and its Applications 13, 18–20 (1977)
61. Williams, H.P.: Model building in mathematical programming (1999)
62. Winnen, L.: Sage s-box milp toolkit, `http://www.ecrypt.eu.org/tools/sage-s-box-milp-toolkit`
63. Wu, S., Wang, M.: Security evaluation against differential cryptanalysis for block cipher structures. Tech. rep., Cryptology ePrint Archive, Report 2011/551 (2011)
64. Wu, S., Wu, H., Huang, T., Wang, M., Wu, W.: Leaked-state-forgery attack against the authenticated encryption algorithm ale. In: Advances in Cryptology-ASIACRYPT 2013, pp. 377–404. Springer (2013)
65. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Applied Cryptography and Network Security. pp. 327–344. Springer (2011)

66. Yap, H., Khoo, K., Poschmann, A., Henricksen, M.: EPCBC-a block cipher suitable for electronic product code encryption. In: Cryptology and Network Security, pp. 76–97. Springer (2011)

A The PRESENT-80 Lightweight Block Cipher

PRESENT-80 is a 31-round SPN block cipher with 64-bit block size and 80-bit secret key. The substitution and diffusion layers of PRESENT-80 are constructed with 4×4 S-boxes and bit-wise permutation to make its hardware implementation suitable for extremely constrained devices.

The schematic description of PRESENT-80's encryption process and key schedule algorithm are given in Fig.2 and Fig.3. These two schematic descriptions are enough to understand the contents of the paper, and for more information on PRESENT, we refer the reader to [15].

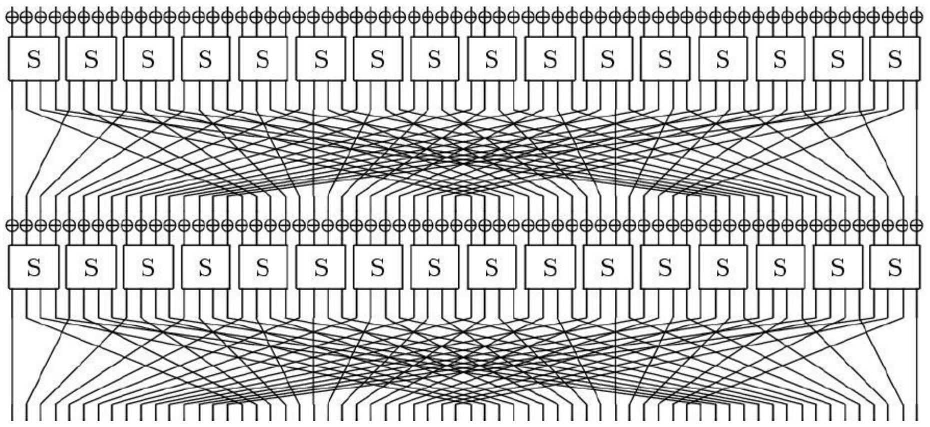

Fig. 2: Two consecutive rounds of PRESENT-80 encryption process

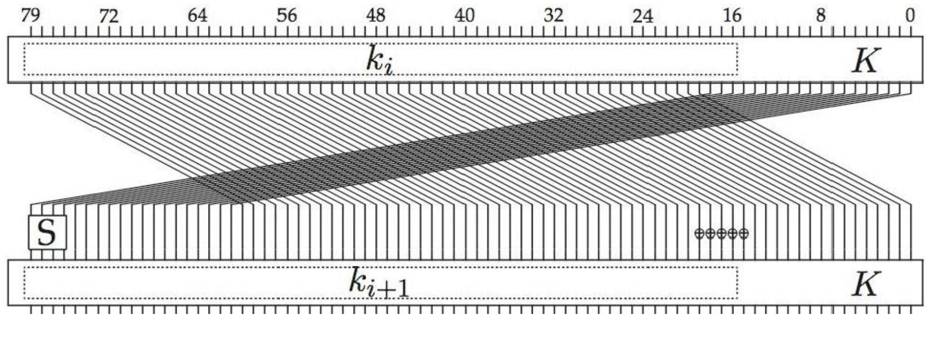

Fig. 3: The key schedule algorithm of PRESENT-80: for each round the most significant 64 bits of the 80-bit key register K are extracted as the subkey k_i

B The Convex Hull of the PRESENT S-box

```
( 0,  2, -2,  1, -1, -1, -2, -2,  6)( 0, -2,  2,  1, -2, -1, -1, -2,  6)( 0,  1, -1,  1,  0, -1, -1, -1,  3)(-1,  0,  1, -1, -1, -1,  0, -1,  4)
( 1,  0,  1,  1,  1,  1, -1,  0,  0)( 0,  1,  1,  1, -1,  0,  0,  1,  0)(-1, -1,  0,  1, -1,  0, -1, -1,  4)( 1,  0, -1,  2,  1,  2,  2, -1,  0)
( 1, -1, -1,  0,  0, -1,  1, -1,  3)(-1,  0, -1, -1,  1,  0, -1,  0,  3)( 0,  1,  1,  0, -1,  1, -1,  1,  1)( 1,  1,  1,  0,  0,  0, -1,  1,  0)
( 2,  1,  2,  2,  0,  1, -1, -1,  0)(-1,  0,  0,  1,  1,  1,  1,  0,  0)(-1,  0,  0,  1,  1,  0,  1,  1,  0)( 0,  1,  1, -1, -1,  1, -1,  0,  2)
( 0,  0, -1,  0,  0,  0,  0,  0,  1)( 1,  2,  0, -1, -2, -2, -2, -1,  6)( 0,  1, -1, -1, -1, -1,  0,  1,  3)( 0,  0, -1,  0,  1,  1,  1,  1,  0)
( 0,  1,  1, -2, -2, -1, -2, -2,  7)( 2,  2, -1,  2, -1,  3,  2, -1,  0)( 1,  0,  1, -1,  0,  1,  1,  1,  0)( 0,  1, -1,  2,  1,  2,  2, -1,  0)
(-2,  1,  1,  3,  1, -1,  1,  2,  0)( 1, -1, -1,  1,  1,  0,  0, -1,  2)( 0,  1,  1, -1,  1,  1,  1,  0,  0)(-2,  1,  1,  2,  1,  0,  1,  1,  0)
( 0, -1, -1, -1, -1,  1,  0, -1,  4)(-1,  1, -2, -1, -2, -2,  1,  2,  6)( 2, -2,  3, -4, -1, -4, -4,  1, 11)( 1, -1,  1,  2,  2,  2,  0, -1,  0)
( 0, -1,  1, -1,  0, -1, -1,  1,  3)( 0,  1, -1,  1,  0,  1,  1, -1,  1)( 1, -1,  2, -1,  0,  2,  2,  2,  0)(-1,  0, -1,  1, -1,  0, -1, -1,  4)
( 0, -1, -1,  0, -1,  1, -1,  1,  3)( 2,  1,  1,  0, -2, -1, -2, -1,  4)( 1,  0, -1,  1,  0,  1,  1, -1,  1)( 1,  2,  2,  1,  0,  0, -2,  1,  0)
( 0,  0,  0,  0,  0, -1,  0,  0,  1)( 1,  1,  1,  1,  1,  1,  1, -2,  0)( 1,  0,  1, -1,  1,  1,  0,  1,  0)( 0, -1,  1, -1, -1, -1, -1,  0,  4)
( 2,  1,  1, -2,  1,  1,  0,  2,  0)(-2,  1, -1, -1,  1, -1, -1, -1,  5)( 1, -1,  0,  2,  2,  2,  1, -1,  0)( 3,  2,  2, -1, -4, -2, -4, -1,  8)
( 0,  0,  0,  1,  1, -1,  1,  1,  0)( 0, -1, -1, -1,  1,  0, -1, -1,  4)( 2,  2,  1,  2, -1,  1,  0, -1,  0)( 1, -1, -1,  0,  1, -1,  0, -1,  3)
( 1, -1, -1,  0, -1,  0,  1, -1,  3)(-2,  0,  0,  1,  2,  1,  2,  1,  0)( 1, -1,  1,  2, -2, -2,  1, -2,  5)( 1, -1,  0,  0,  1,  1, -1, -1,  2)
( 3,  2,  3,  3, -1, -1,  0, -1,  0)( 1,  1, -1,  1, -1,  2,  1, -1,  1)( 1, -1, -1,  1,  2,  0,  2,  1,  0)( 1, -1, -1,  1,  0,  0,  1, -1,  2)
( 1,  1,  1,  1,  1,  1, -2,  1,  0)( 2,  1,  2, -1,  1,  0, -1,  2,  0)( 0, -1, -1, -1, -1,  0,  1, -1,  4)(-2,  1, -2, -1,  1, -1, -2,  0,  6)
( 0, -1,  2, -1, -2, -2, -2, -1,  7)(-1,  0, -1, -1,  1, -1,  0, -1,  4)( 3, -3, -2,  1, -1, -1,  3, -3,  7)( 1,  1, -1,  1,  0,  1,  1,  0,  0)
( 1,  1, -1, -1, -1, -1,  0,  0,  3)( 1, -2, -2,  0,  1, -2,  1, -1,  5)(-1, -1, -1, -1,  3,  4,  3,  4,  0)( 2,  2,  2,  0, -1,  0, -1,  1,  0)
( 2,  0,  2, -1,  2,  1, -1,  2,  0)( 1, -1, -2, -1, -2,  0,  2, -2,  6)( 1, -1,  0, -1,  1,  0, -1, -1,  3)( 1,  0, -1, -1,  0, -1,  1, -1,  3)
( 1,  1,  1,  1,  0,  0, -1,  0,  0)( 1,  0,  1,  1,  1,  1,  0, -1,  0)( 0,  0,  0, -1,  0,  0,  0,  0,  1)( 1, -2, -2, -1,  2, -3,  1, -2,  7)
( 1,  0,  1,  0, -1,  1,  1,  1,  0)(-1,  1,  1, -2, -1, -1, -1, -2,  6)( 1,  2, -1, -2, -2, -1,  1,  1,  4)( 2,  2, -1,  1, -1,  2,  2,  0,  0)
( 1,  1,  0,  1, -1,  1,  0, -1,  1)( 0,  0,  0, -1,  1,  1,  1,  1,  0)( 1,  1, -1,  0,  0,  1,  1,  1,  0)( 1, -1,  2, -2,  1, -1, -2,  1,  4)
( 1,  1, -1,  0, -1,  1,  1,  0,  1)(-3,  2, -2, -1,  1, -2, -2, -1,  8)( 0,  0,  1, -1, -1, -1, -1, -1,  4)( 0,  0,  0,  1,  0,  0,  0,  0,  0)
(-1,  0, -1, -1,  3,  3,  2,  3,  0)( 0,  1, -1, -1, -1, -1, -1,  0,  4)(-1,  0, -1, -1, -1, -1,  1,  1,  4)(-1,  1, -1,  0, -1, -1,  1,  1,  3)
( 1, -1,  1,  0,  0,  1,  1,  1,  0)( 1, -1,  1,  0,  1,  1, -1,  0,  1)(-4,  1,  1,  2,  3,  2,  3,  1,  0)( 2,  1,  1, -3,  1,  2,  1,  2,  0)
( 0,  0, -1,  1,  1,  1,  1,  0,  0)(-1,  0, -1, -1,  1,  1,  0,  1,  2)( 0,  0, -1,  1,  1,  0,  1,  1,  0)( 0,  1, -1,  1, -1,  0, -1, -1,  3)
( 1, -1,  0, -1,  1, -1,  0, -1,  3)(-1,  1,  0, -1,  0, -1, -1, -1,  4)( 0, -1,  0,  0,  0,  0,  0,  0,  1)( 0, -1, -1, -1, -1,  1, -1,  0,  4)
( 1,  1, -1,  2,  1, -2, -2, -2,  5)( 1,  1, -1, -1, -1,  0,  1,  0,  2)(-1,  1,  0, -1,  1, -1,  0, -1,  3)( 0, -1,  0,  0,  1,  1,  1,  1,  0)
( 0,  1,  1, -1, -1,  0, -1, -1,  3)( 0,  1, -1,  0, -1, -1, -1, -1,  4)( 2, -3, -1, -1,  2,  1, -3, -3,  8)( 0,  0,  0,  0,  0,  0,  0, -1,  1)
( 1,  2, -1, -2, -2, -2, -1,  0,  6)(-1,  1,  1,  2,  0, -1,  1,  2,  0)( 1,  1, -2,  1,  1,  1,  1,  1,  0)( 1, -1,  1,  1,  1,  1,  0,  0,  0)
( 0, -1, -1, -1,  1, -2,  1, -2,  5)( 1,  3, -2, -2,  3,  4,  1,  4,  0)(-1, -1, -1, -1,  0,  1,  0,  1,  3)(-2,  0,  2, -2, -1, -1,  1, -2,  6)
( 1, -2,  1,  0,  1,  2,  1,  2,  0)( 1, -1, -2, -2,  1, -3,  2, -2,  7)( 0, -2, -2,  3,  4,  1,  4,  1,  0)( 0,  0,  0, -1,  1, -1,  1, -1,  2)
(-1,  1,  1,  1,  0,  0,  0,  1,  0)(-1,  0,  1,  1,  0, -1,  1,  1,  1)( 0,  2,  2,  1, -1,  1, -1,  2,  0)( 1,  2, -1, -2, -2, -2,  0,  1,  5)
(-1,  1,  1, -1,  1, -1,  1, -2,  3)( 1,  1,  0,  1, -1,  1,  1,  0,  0)( 2,  2,  0, -1, -1,  1,  2,  2,  0)( 2,  1, -2, -2, -1, -1,  1, -1,  5)
( 4,  1,  3, -2,  3,  1, -2,  4,  0)( 1,  0,  0, -1, -1, -1, -1,  1,  3)( 0,  0, -2,  1,  2,  1,  2,  1,  0)( 0,  0,  0,  0, -1,  0,  0,  0,  1)
( 2,  2,  2,  1,  1, -3,  1,  1,  0)( 1,  1,  0, -2,  1,  2,  1,  2,  0)( 0, -1,  1,  1,  1,  1,  0, -1,  1)(-1,  1,  0,  1,  0, -1,  1,  1,  1)
( 1, -1,  1, -1,  1,  0, -1,  0,  2)( 1, -2, -1, -2,  2, -3,  1, -2,  7)( 1,  1,  0, -1, -1,  0,  1,  1,  1)( 1,  1,  1, -2,  1,  1,  1,  1,  0)
(-1, -1,  1,  0,  1, -1, -1,  1,  3)( 1,  1,  1, -1,  1,  0,  1,  0,  0)( 1,  1,  0, -1,  0,  1,  1,  1,  0)(-2, -1, -1,  2, -2,  0, -2, -1,  7)
( 1,  1,  0, -1, -1, -1, -1,  0,  3)( 1, -1, -1,  0,  1,  0,  1,  1,  1)( 0, -1,  1,  1,  0,  1, -1, -1,  2)( 3,  1,  1, -2, -2, -2, -2,  1,  5)
(-2, -1,  2, -1, -2, -2,  0, -1,  7)( 1,  1,  1,  0, -1,  0, -1,  0,  1)( 2,  0,  2,  1, -2, -1, -1, -2,  4)( 1,  0,  1,  0, -1, -1, -1, -1,  3)
(-2,  2,  1,  4,  1, -2,  2,  3,  0)(-1, -1,  1,  0, -1, -1,  0, -1,  4)( 1,  1,  0,  1,  0, -1, -1, -1,  2)( 1, -1, -1,  0,  1, -1,  1,  0,  2)
( 0, -1, -1,  1,  1,  0,  1,  0,  1)(-1, -1, -1, -1, -1,  1,  0,  0,  4)( 1,  0, -1, -1, -1,  0,  1, -1,  3)( 1, -2,  3, -2,  1,  4,  3,  4,  0)
( 1,  1,  0, -1,  1,  1,  0,  1,  0)( 1,  4, -1, -2, -4, -4, -3, -2, 12)( 0,  0,  0,  0,  0,  0,  0,  1,  0)( 0, -1,  1,  1, -1,  0, -1, -1,  3)
( 0,  1,  0,  0,  0,  0,  0,  0,  0)( 3, -1,  3, -1,  3,  2, -1,  2,  0)( 1,  1,  1,  1, -2,  1,  1,  1,  0)( 1,  0,  1,  1, -1, -1,  0, -1,  2)
( 0, -1, -1, -1,  1, -1,  0, -1,  4)( 2, -2,  1, -2,  1, -1, -1, -1,  5)( 1, -1,  3, -1, -2, -3, -3, -2,  9)( 1,  0,  1, -1, -1, -1, -1,  0,  3)
( 0, -1, -1,  1, -1,  0, -1,  1,  3)(-1,  0,  1,  2,  1, -1,  1,  2,  0)( 1,  0,  1, -1,  0, -1, -1,  1,  2)( 1,  0,  2, -1, -2, -2, -2, -1,  6)
( 1,  0,  1, -2,  1,  2,  1,  2,  0)( 0, -1,  0,  1,  1,  1,  1,  0,  0)( 1, -1, -1,  0,  1,  0, -1, -1,  3)( 0, -1,  0,  1,  1,  0,  1,  1,  0)
(-1,  1, -1, -1,  0, -1, -1,  0,  4)(-1,  0,  1, -1,  0, -1,  1, -1,  3)( 2,  1,  1, -1, -2, -1, -2,  0,  4)( 2, -1,  2,  1,  2,  2, -1,  0,  0)
( 1,  2,  2,  1,  0, -2,  0,  1,  0)( 1,  0,  0,  0,  0,  0,  0,  0,  0)( 1,  0,  1,  1,  0,  1, -1, -1,  1)( 2, -1, -3, -1, -3,  1,  2, -3,  8)
( 1, -1,  2, -2,  0, -2, -2,  1,  5)(-1, -1,  0, -1,  2,  3,  3,  3,  0)( 0,  1,  1,  1,  0, -1,  0,  1,  0)( 1,  1,  1,  0, -1,  0,  0,  1,  0)
( 1,  0,  1, -1,  1,  0, -1,  1,  1)(-2,  1,  2,  4,  1, -2,  2,  3,  0)( 1, -1,  4, -2, -3, -4, -4, -2, 12)( 1,  1,  0,  0,  1,  1, -1,  1,  0)
(-1, -1, -1,  0, -1,  1, -1,  0,  4)(-1, -3,  2,  1, -3, -2, -1, -3, 10)( 3,  3,  2,  3,  0, -1, -1, -1,  0)( 1,  1,  0,  0, -1,  1,  1,  1,  0)
( 0,  0,  0,  0,  0,  1,  0,  0,  0)( 0, -1, -1,  2,  2,  1,  2,  0,  0)( 4,  3,  1, -2, -2,  1,  3,  4,  0)( 0,  2, -1, -1, -2, -2, -2, -1,  7)
( 0,  1,  1,  2,  1, -2,  1,  1,  0)( 1,  1,  1,  1,  0, -1,  0,  0,  0)( 0, -1, -1,  2,  2,  0,  2,  1,  0)(-1, -1,  0, -1, -1,  0,  1,  0,  3)
( 1,  1, -1,  2,  0,  2,  2, -1,  0)( 1,  3, -1, -1, -3, -3, -2, -2,  9)( 1, -1, -1,  2,  1,  1,  1, -1,  1)( 0, -1, -1, -1,  0, -1,  1, -1,  4)
( 2,  2,  1, -1, -1,  0,  1,  2,  0)( 1, -2,  2, -3, -1, -3, -3,  1,  9)( 3,  4,  4,  1, -2,  0, -2,  1,  0)( 0,  1,  1,  0,  1,  0,  1, -1,  0)
( 2,  1,  1,  0, -2,  1,  1,  2,  0)( 0, -2,  0,  1,  2,  1,  2,  1,  0)( 1, -2, -1, -1,  2, -2,  0, -2,  6)(-2,  2,  0, -2,  1, -1, -1, -2,  6)
( 0, -1, -1, -1,  0,  1, -1, -1,  4)(-2, -1, -1,  1, -2,  1, -2, -1,  7)(-1,  2, -3,  1, -1, -2, -3, -3, 10)( 1,  2, -1, -1,  2,  2,  0,  2,  0)
(-3, -2,  2, -1, -2, -2,  1, -1,  8)( 0, -1,  1,  0, -1, -1, -1, -1,  4)( 1, -1, -1,  3,  2,  2,  2, -1,  0)( 1, -1,  0,  1,  1,  1,  0, -1,  1)
( 2,  3, -2, -4, -4, -4, -1,  1, 11)( 2,  0,  1, -2, -1, -2, -2,  1,  5)(-1,  1, -1,  0,  1, -1, -1,  0,  3)( 2,  3,  3,  2,  1, -4,  1,  1,  0)
(-1,  1,  0, -1,  1,  0, -1, -1,  3)(-1,  1,  1,  3,  1, -2,  1,  2,  0)( 0,  0,  0,  0,  0,  0, -1,  0,  1)(-2,  2, -1, -1,  0, -2, -2, -1,  7)
( 1,  1,  1,  0,  0, -1,  0,  1,  0)( 1,  1,  0,  0, -1, -1, -1, -1,  3)( 1,  2, -2, -3, -3, -3, -1,  1,  9)( 1, -2,  1,  1,  1,  1,  1,  1,  0)
(-3,  1,  1,  1,  2,  2,  2,  1,  0)( 1,  2,  2,  0, -1,  1, -1,  1,  0)( 1, -2, -3, -2,  1, -4,  3, -3, 10)(-1,  0,  1, -1, -1,  0,  1, -1,  3)
( 0, -1,  1,  2,  2,  2,  1, -1,  0)( 0,  1,  1,  1,  0,  0, -1,  1,  0)( 3,  2,  2,  2, -1,  0, -1, -1,  0)(-1, -1, -1, -1,  0,  1, -1,  0,  4)
( 2,  2,  2, -1,  3, -1,  3, -1,  0)(-1,  1, -1,  0,  0, -1, -1, -1,  4)( 2,  1,  1,  1, -1,  0, -1, -1,  1)( 2,  2,  0,  1, -1, -1, -2, -2,  4)
( 0,  0,  0,  1,  1,  1,  1, -1,  0)(-1,  0,  0,  0,  0,  0,  0,  0,  1)( 1, -1,  1,  1,  1,  2, -1, -1,  1)( 0,  0,  0,  0,  0,  0,  1,  0,  0)
(-1, -2, -2, -1, -1,  2, -1,  0,  6)( 1,  0, -1,  0, -1,  1,  1, -1,  2)( 0, -1,  1, -1,  0,  1,  1,  1,  1)( 0, -1,  1, -1,  1,  2,  2,  2,  0)
( 4,  3,  3, -1, -1, -1, -1,  3,  0)(-1, -1, -1,  1, -1,  0, -1,  0,  4)( 1,  1, -1,  0,  1,  1,  0,  1,  0)( 1,  1,  1,  1,  0,  0,  0, -1,  0)
(-1, -1,  0, -1,  0,  1,  1,  1,  2)( 1,  1,  0, -1, -1, -1,  0,  1,  2)(-2,  1,  2,  4,  2, -2,  1,  3,  0)(-2, -2,  1, -1, -2, -1,  1,  0,  6)
( 0, -1,  1,  1, -1, -1,  0, -1,  3)( 1,  1,  1, -1,  0,  0,  0,  1,  0)( 1,  2,  2,  1, -2,  0,  0,  1,  0)(-1, -1,  1,  0, -1, -1,  1,  0,  3)
( 3,  3,  2, -1, -1, -1,  0,  3,  0)(-1,  0,  0,  0,  1,  1,  1,  1,  0)( 3,  3, -1, -1, -1,  2,  3,  2,  0)( 2, -1,  2,  0,  2,  2, -1,  1,  0)
(-1, -1,  0, -1,  0, -1,  1, -1,  4)(-2, -1,  1, -1, -1, -1,  1, -1,  5)( 1,  1, -2,  0,  1,  2,  1,  2,  0)( 1, -1, -1,  0, -1,  0, -1,  1,  3)
( 0,  1,  0, -1, -1, -1, -1, -1,  4)( 2,  1,  1,  0,  1,  1, -2,  2,  0)(-2,  2,  1,  4,  2, -2,  1,  3,  0)( 1, -2, -2, -1,  1, -3,  2, -2,  7)
( 1,  0,  1,  0,  1,  1, -1,  1,  0)( 1, -2, -1, -1,  2,  0, -2, -2,  6)(-1,  0,  1,  1,  1, -1,  0,  1,  1)( 2,  1,  0, -2, -2, -2, -1,  1,  5)
( 1, -1, -2, -1,  0, -2,  2, -2,  6)(-1,  1,  1, -1,  0,  0,  0, -1,  2)( 1,  1,  1,  1, -1,  0,  0,  0,  0)( 1, -1, -1, -1,  1, -2,  1, -1,  4)
( 2,  1,  1, -2,  0,  1,  1,  2,  0)(-1, -1,  1, -1, -1, -1,  0,  0,  4)( 3,  2,  3, -1,  0, -1, -1,  3,  0)( 1, -1,  1,  0,  1,  1,  0,  1,  0)
( 1, -2, -2,  2,  1,  0,  1, -1,  3)( 1, -1,  0,  1, -1, -1,  1, -1,  3)(-1,  3,  3, -1,  2,  2,  2, -1,  0)(-1,  1,  0,  1,  1, -1,  0,  1,  1)
(-1, -2,  1, -1,  1, -2, -2,  2,  6)( 0,  1, -1,  1, -1,  1,  0, -1,  2)( 0,  1, -1, -1,  2,  2,  1,  2,  0)(-1, -1,  0, -1,  1, -1, -1,  1,  4)
( 0,  0,  0,  0,  1,  0,  0,  0,  0)( 1, -3, -2, -2,  3, -4,  1, -3, 10)( 2,  2, -1,  0, -1,  2,  2,  1,  0)( 1, -1,  2, -2, -1, -2, -2,  0,  6)
( 2, -1,  2,  2,  2,  3, -1, -1,  0)( 0, -2, -2, -2, -1,  2, -1, -1,  7)( 3, -2, -3,  1,  3, -1, -1, -3,  7)(-1,  1,  0,  2,  1, -1,  1,  2,  0)
( 0,  1, -1, -1,  1,  1,  0,  1,  1)( 1,  1,  0,  1,  0,  1,  1, -1,  0)( 1, -1,  1, -1,  0, -1, -1,  0,  3)( 1, -1, -1,  0,  2,  1,  2,  2,  0)
( 0,  0,  1,  0,  0,  0,  0,  0,  0)( 1,  0, -1,  1,  1, -1, -1, -1,  3)(-1,  1,  1,  2,  1, -1,  0,  2,  0)
```

C Valid Cutting-off Inequalities Used in Analyzing LBlock

S-box	Valid cutting-off inequalities
S0	(-1, 2, -2, -1, 0, 0, -2, -1, 5), (0, 1, 0, 0, 1, -1, 1, 0, 0), (-1, -1, 1, -3, 3, -1, -2, 2, 5), (3, -1, -1, -1, 0, 3, 2, 1, 0), (-1, 1, 2, 0, -1, -1, 2, -2, 3), (0, -1, 0, 1, -1, 0, -1, 1, 2), (0, -1, 0, 0, 1, 1, 1, 0, 0), (-1, -1, -1, 0, -1, -1, 0, -1, 5), (1, 2, -2, 1, 0, 0, 1, 2, 0), (1, 2, 3, -2, 1, 0, -1, 3, 0), (-1, 0, 0, 0, 1, 0, 1, 1, 0), (1, 1, -2, -2, 0, -1, -1, -2, 6), (-1, -1, 1, 0, -1, 1, -1, -1, 4), (1, 0, 1, 1, 0, -1, -1, -1, 2), (1, -1, 1, -1, 2, 2, 0, 1, 0), (-1, 2, 2, 1, 0, 0, 2, -1, 0), (0, -1, -1, 1, 1, 1, 0, -1, 2), (-1, 1, 0, 0, -1, 1, 1, -1, 2), (2, 3, 1, 1, 0, -3, 1, 1, 0), (1, -1, -1, 0, 1, -1, -1, 1, 3), (2, -1, -1, 0, -1, 1, 1, 1, 1), (3, 1, 2, 1, -3, -1, 1, 3, 0), (2, -1, -1, 1, -2, 1, 0, 1, 2), (1, -1, 1, -1, 0, 1, 0, -1, 2), (1, 1, 2, 2, 0, 1, 1, -2, 0), (-1, -1, -1, -2, 2, 1, 0, 1, 3)
S1	(2, 1, 1, 1, -3, 0, 2, 1, 0), (-1, 2, -2, -1, 0, 0, -1, -2, 5), (0, 1, 0, 0, 1, -1, 0, 1, 0), (-1, -1, 1, -3, 3, -1, 2, -2, 5), (3, -1, -1, -1, 0, 3, 1, 2, 0), (-1, 4, 5, 3, -1, -1, -2, 5, 0), (0, -1, 0, 1, -1, 0, 1, -1, 2), (0, -1, 0, 0, 1, 1, 0, 1, 0), (1, 2, -2, 1, 0, 0, 2, 1, 0), (-1, -1, -1, 0, -1, -1, -1, 0, 5), (0, 1, 2, -2, 1, 0, 2, -1, 1), (-1, 0, 0, 0, 1, 0, 1, 1, 0), (1, 1, -2, -2, 0, -1, -2, -1, 6), (2, 0, 1, 1, -2, -1, 1, -1, 2), (-1, -1, 1, 0, -1, 1, -1, -1, 4), (-1, 1, -1, 0, -1, 1, -1, 0, 3), (1, -1, 1, -1, 0, 1, -1, 0, 2), (-1, -1, -1, 0, 1, 1, -1, 0, 3), (0, -1, 1, 1, 1, -1, -1, -1, 3), (1, -1, -1, 1, 0, 1, -1, 0, 2), (2, 3, 1, 1, 0, -3, 1, 1, 0), (2, -1, 2, 3, -1, 2, 3, -1, 0), (-1, 1, 1, -1, 2, 0, 1, 1, 0), (-1, -1, 0, 0, -1, -1, -1, 1, 4), (1, -1, 0, 0, 0, 1, 0, 1, 0), (3, -1, -1, 0, -1, 2, 2, 2, 0), (1, 0, -1, 1, 0, -1, 1, -1, 2)
S2	(2, 1, 1, 1, 1, -3, 2, 0, 0), (-1, 2, -2, -1, -2, 0, -1, 0, 5), (0, 1, 0, 0, 1, 1, 0, -1, 0), (-1, -1, 1, -3, -2, 3, 2, -1, 5), (3, -1, -1, -1, 2, 0, 1, 3, 0), (-1, 4, 5, 3, 5, -1, -2, -1, 0), (0, -1, 0, 1, -1, -1, 1, 0, 2), (0, -1, 0, 0, 1, 1, 0, 1, 0), (-1, -1, -1, 0, 0, -1, -1, -1, 5), (1, 2, -2, 1, 1, 0, 2, 0, 0), (1, -2, 1, -2, 1, 3, 2, 4, 0), (1, 1, -2, -2, -1, 0, -2, -1, 6), (-1, 0, 0, 0, 1, 1, 1, 0, 0), (2, 0, 1, 1, -1, -2, 1, -1, 2), (0, -1, 1, 1, -1, 1, -1, -1, 3), (0, 1, 1, -1, -1, 0, 1, 0, 1), (-1, -1, -1, 0, 0, 1, -1, 1, 3), (0, -1, 1, -1, -1, -1, -1, 1, 4), (1, -1, -1, 1, 0, 0, -1, 1, 2), (-1, 1, 0, 0, 1, -1, -1, 1, 2), (3, 2, -1, 3, -1, 0, 3, -1, 0), (1, 2, 1, 1, 1, 0, 0, -2, 0), (-1, 2, 1, -2, 1, 3, 2, 0, 0), (-1, 1, 2, 0, 2, -1, -2, -1, 3), (3, 1, 2, 2, 1, -4, 2, 1, 0), (-1, -1, 1, 1, -1, -1, 0, 1, 3), (3, -1, -1, 0, 2, -1, 2, 2, 0)
S3	(2, 1, 1, 1, 0, 1, -3, 2, 0), (-1, 2, -2, -1, 0, -2, 0, -1, 5), (0, 1, 0, 0, -1, 1, 1, 0, 0), (-1, -1, 1, -3, -1, -2, 3, 2, 5), (3, -1, -1, -1, 3, 2, 0, 1, 0), (-1, 1, 2, 0, -1, 2, -1, -2, 3), (0, -1, 0, 1, 0, -1, -1, 1, 2), (0, -1, 0, 0, 1, 1, 1, 0, 0), (-1, -1, -1, 0, -1, 0, -1, -1, 5), (1, 2, -2, 1, 0, 1, 0, 2, 0), (1, 2, 2, -1, 0, -1, 0, 2, 0), (1, 1, -2, -2, -1, -1, 0, -2, 6), (-1, 0, 0, 0, 0, 1, 1, 1, 0), (1, 0, 1, 1, -1, -1, 0, -1, 2), (1, -1, -1, 1, 1, 0, 0, -1, 2), (1, -1, 1, -1, 2, 0, 2, 1, 0), (-1, 0, 1, 0, 1, 1, 1, 0, 0), (-1, 0, -1, 0, 1, -1, 1, -1, 3), (-1, 1, -1, 0, 1, 0, -1, -1, 3), (0, -1, 1, -1, 1, -1, -1, -1, 4), (3, 2, -1, 3, -1, -1, 0, 3, 0), (1, 2, -1, 1, -1, 1, 0, 1, 0), (3, 1, 2, 0, -1, 1, -2, 2, 0), (-1, -1, 1, 1, 1, -1, -1, 0, 3), (1, 1, 2, 2, 1, 1, 0, -2, 0), (2, -1, -1, 0, 1, 1, -1, 1, 1), (0, -1, 1, 1, -1, -1, 1, -1, 3)
S4	(2, 1, 1, 1, 1, -3, 0, 2, 0), (-1, 2, -2, -1, -2, 0, 0, -1, 5), (0, 1, 0, 0, 1, 1, -1, 0, 0), (-1, -1, 1, -3, -2, 3, -1, 2, 5), (3, -1, -1, -1, 2, 0, 3, 1, 0), (-1, 4, 5, 3, 5, -1, -1, -2, 0), (0, -1, 0, 1, -1, -1, 0, 1, 2), (0, -1, 0, 0, 1, 1, 1, 0, 0), (-1, -1, -1, 0, 0, -1, -1, -1, 5), (1, 2, -2, 1, 1, 0, 0, 2, 0), (1, -1, 1, -1, -1, 2, 1, 1, 1), (1, 1, -2, -2, -1, 0, -1, -2, 6), (-1, 0, 0, 0, 1, 1, 0, 1, 0), (-1, -1, -1, 0, 0, 1, 1, -1, 3), (1, 0, 1, 1, -1, 0, -1, -1, 2), (1, -1, -1, 1, 0, 0, 1, -1, 2), (-1, 1, 0, 0, 1, -1, 1, -1, 2), (2, 2, 3, -1, -1, 0, -1, 3, 0), (-1, -1, 1, 0, -1, -1, 1, -1, 4), (3, -1, -1, 0, 2, -1, 2, 2, 0), (-1, 1, 1, -1, 1, 2, 0, 1, 0), (1, -1, 0, 0, 1, 0, 1, 0, 0), (1, -1, 1, -1, 0, 0, 1, -1, 2), (-1, -1, 1, 0, -1, 1, -1, -1, 4), (2, 3, 1, 1, 1, 0, -3, 1, 0), (1, 0, -1, 1, -1, 0, -1, 1, 2), (0, 1, 2, -2, -1, 1, 0, 2, 1)
S5	(2, 1, 1, 1, -3, 1, 0, 2, 0), (-1, 2, -2, -1, 0, -2, 0, -1, 5), (0, 1, 0, 0, 1, 1, -1, 0, 0), (-1, -1, 1, -3, 3, -2, -1, 2, 5), (3, -1, -1, -1, 0, 2, 3, 1, 0), (-1, 1, 2, 0, -1, 2, -1, -2, 3), (0, -1, 0, 1, -1, -1, 0, 1, 2), (0, -1, 0, 0, 1, 1, 1, 0, 0), (-1, -1, -1, 0, -1, 0, -1, -1, 5), (1, 2, -2, 1, 0, 1, 0, 2, 0), (1, -1, 1, -1, 2, -1, 1, 1, 1), (-1, 0, 0, 0, 1, 1, 0, 1, 0), (1, 1, -2, -2, 0, -1, -1, -2, 6), (0, 1, 1, -1, 0, -1, 0, 1, 1), (-1, 0, 1, 0, 1, 1, 1, 0, 0), (1, 0, 1, 1, 0, -1, -1, -1, 2), (1, -1, -1, 1, 0, 0, 1, -1, 2), (-1, 1, -1, 0, -1, 0, 1, -1, 3), (0, -1, 1, -1, -1, -1, 1, -1, 4), (-1, -1, -1, 0, 1, 0, 1, -1, 3), (2, -1, -1, 0, -1, 1, 1, 1, 1), (1, -1, -1, 0, 1, -1, -1, 1, 3), (1, 1, 0, 0, 0, 1, -1, 0, 0), (5, 2, 4, 1, -4, 1, -2, 4, 0), (1, 1, 2, 2, 0, 1, 1, -2, 0), (-1, -1, 1, 1, -1, -1, 1, 0, 3), (0, -1, 1, 1, 1, -1, -1, -1, 3)
S6	(2, 1, 1, 1, -3, 0, 2, 1, 0), (-1, 2, -2, -1, 0, 0, -1, -2, 5), (0, 1, 0, 0, 1, -1, 0, 1, 0), (-1, -1, 1, -3, 3, -1, 2, -2, 5), (3, -1, -1, -1, 0, 3, 1, 2, 0), (-1, 4, 5, 3, -1, -1, -2, 5, 0), (0, -1, 0, 1, -1, 0, 1, -1, 2), (0, -1, 0, 0, 1, 1, 0, 1, 0), (1, 2, -2, 1, 0, 0, 2, 1, 0), (-1, -1, -1, 0, -1, -1, -1, 0, 5), (0, 1, 2, -2, 1, 0, 2, -1, 1), (-1, 0, 0, 0, 1, 0, 1, 1, 0), (1, 1, -2, -2, 0, -1, -2, -1, 6), (2, 0, 1, 1, -2, -1, 1, -1, 2), (-1, -1, 1, 0, -1, 1, -1, -1, 4), (-1, 1, -1, 0, -1, 1, -1, 0, 3), (1, -1, 1, -1, 0, 1, -1, 0, 2), (-1, -1, -1, 0, 1, 1, -1, 0, 3), (0, -1, 1, 1, 1, -1, -1, -1, 3), (1, -1, -1, 1, 0, 1, -1, 0, 2), (2, 3, 1, 1, 0, -3, 1, 1, 0), (2, -1, 2, 3, -1, 2, 3, -1, 0), (-1, 1, 1, -1, 2, 0, 1, 1, 0), (-1, -1, 0, 0, -1, -1, -1, 1, 4), (1, -1, 0, 0, 0, 1, 0, 1, 0), (3, -1, -1, 0, -1, 2, 2, 2, 0), (1, 0, -1, 1, 0, -1, 1, -1, 2)
S7	(2, 1, 1, 1, -3, 0, 2, 1, 0), (-1, 2, -2, -1, 0, 0, -1, -2, 5), (0, 1, 0, 0, 1, -1, 0, 1, 0), (-1, -1, 1, -3, 3, -1, 2, -2, 5), (3, -1, -1, -1, 0, 3, 1, 2, 0), (-1, 4, 5, 3, -1, -1, -2, 5, 0), (0, -1, 0, 1, -1, 0, 1, -1, 2), (0, -1, 0, 0, 1, 1, 0, 1, 0), (1, 2, -2, 1, 0, 0, 2, 1, 0), (-1, -1, -1, 0, -1, -1, -1, 0, 5), (0, 1, 2, -2, 1, 0, 2, -1, 1), (-1, 0, 0, 0, 1, 0, 1, 1, 0), (1, 1, -2, -2, 0, -1, -2, -1, 6), (2, 0, 1, 1, -2, -1, 1, -1, 2), (-1, -1, 1, 0, -1, 1, -1, -1, 4), (-1, 1, -1, 0, -1, 1, -1, 0, 3), (1, -1, 1, -1, 0, 1, -1, 0, 2), (-1, -1, -1, 0, 1, 1, -1, 0, 3), (0, -1, 1, 1, 1, -1, -1, -1, 3), (1, -1, -1, 1, 0, 1, -1, 0, 2), (2, 3, 1, 1, 0, -3, 1, 1, 0), (2, -1, 2, 3, -1, 2, 3, -1, 0), (-1, 1, 1, -1, 2, 0, 1, 1, 0), (-1, -1, 0, 0, -1, -1, -1, 1, 4), (1, -1, 0, 0, 0, 1, 0, 1, 0), (3, -1, -1, 0, -1, 2, 2, 2, 0), (1, 0, -1, 1, 0, -1, 1, -1, 2)
S8	(2, 1, 1, 1, 0, -3, 2, 1, 0), (-1, 2, -2, -1, 0, 0, -1, -2, 5), (0, 1, 0, 0, -1, 1, 0, 1, 0), (-1, -1, 1, -3, -1, 3, 2, -2, 5), (3, -1, -1, -1, 3, 0, 1, 2, 0), (-1, 4, 5, 3, -1, -1, -2, 5, 0), (0, -1, 0, 1, 0, -1, 1, -1, 2), (0, -1, 0, 0, 1, 1, 0, 1, 0), (1, 2, -2, 1, 0, 0, 2, 1, 0), (-1, -1, -1, 0, -1, -1, -1, 0, 5), (1, 1, 2, -1, -1, 0, 2, -1, 1), (1, 1, -2, -2, -1, 0, -2, -1, 6), (-1, 0, 0, 0, 0, 1, 1, 1, 0), (3, 2, 3, 3, -1, 0, -1, -1, 0), (1, -1, -1, 1, 1, 0, -1, 0, 2), (1, -1, 1, -1, 1, 2, 1, -1, 1), (-1, -1, -1, 0, 1, 1, -1, 0, 3), (-1, 1, 0, 0, 1, -1, -1, 1, 2), (0, -1, 1, -1, 1, -1, -1, -1, 4), (-1, 2, 1, -2, 0, 3, 2, 1, 0), (3, 2, -1, 3, -1, 0, 3, -1, 0), (-1, -1, 0, 0, -1, -1, -1, 1, 4), (-1, -1, 1, 0, -1, 1, -1, -1, 4), (2, 3, 1, 1, -3, 0, 1, 1, 0), (3, 1, 2, 2, 1, -4, 2, 1, 0), (-1, -1, 1, 1, 1, -1, 0, -1, 3), (-1, 1, 0, -1, 0, 0, 1, -1, 2)
S9	(2, 1, 1, 1, 1, 2, 0, -3, 0), (-1, 2, -2, -1, -2, -1, 0, 0, 5), (0, 1, 0, 0, 1, 0, -1, 1, 0), (-1, -1, 1, -3, -2, 2, -1, 3, 5), (3, -1, -1, -1, 2, 1, 3, 0, 0), (-1, 4, 5, 3, 5, -2, -1, -1, 0), (0, -1, 0, 1, -1, 1, 0, -1, 2), (0, -1, 0, 0, 1, 0, 1, 1, 0), (-1, -1, -1, 0, 0, -1, -1, -1, 5), (1, 2, -2, 1, 1, 2, 0, 0, 0), (1, -1, 1, -1, -1, 1, 1, 2, 1), (1, 1, -2, -2, -1, -2, -1, 0, 6), (-1, 0, 0, 0, 1, 1, 0, 1, 0), (-1, -1, -1, 0, 0, -1, 1, 1, 3), (-1, 1, 0, 0, 1, -1, 1, -1, 2), (1, -1, -1, 1, 0, -1, 1, 0, 2), (-1, 1, 0, -1, -1, 1, 0, 0, 2), (6, 2, 3, 3, -1, 3, -1, -4, 0), (0, -1, 1, 1, -1, -1, -1, 1, 3), (0, -1, 1, -1, -1, -1, 1, -1, 4), (1, -1, 0, 0, 1, 0, 1, 0, 0), (2, 3, 1, 1, 1, 1, -3, 0, 0), (1, 0, -1, 1, -1, 1, -1, 0, 2), (3, -1, -1, -1, 2, 1, 2, -1, 1), (-1, 0, 1, -1, 1, 1, 1, 2, 0), (-1, 1, 2, 0, 2, -2, -1, -1, 3), (-1, -1, 1, 1, -1, 0, 1, -1, 3)

D Related-key Characteristics for DESL, LBlock and PRESENT-128

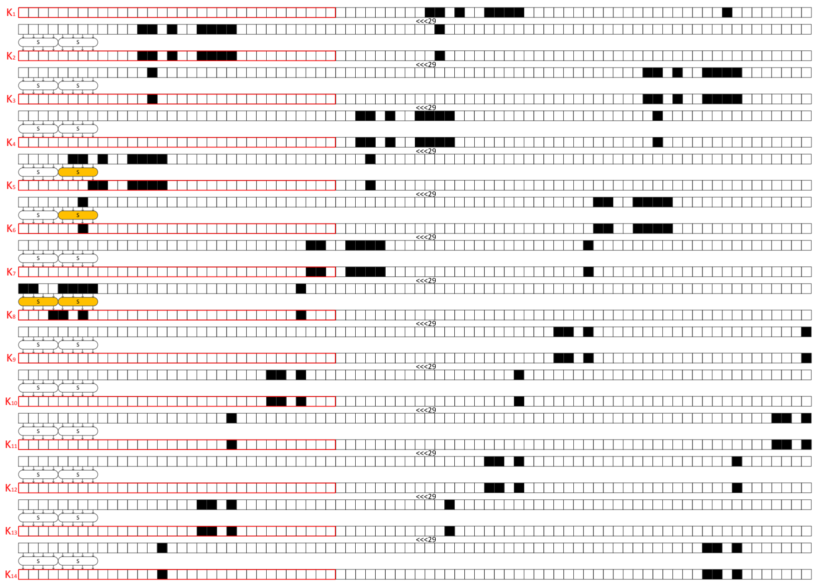

Fig. 4: The key schedule of LBlock

```
PRESENT-128
The differences of the key registers
Round 0:0000000000000000000001000000000000000000000000000000000000000000
        0000000000000100000000000000000000000000000000010000000000000000000

Round 1:0000000000000000100000000000000000000000000000001000000000000000
        0000000000000000000000001000000000000000000000000000000000000000

Round 2:0000000000000000000000000001000000000000000000000000000000000000
        0000000000000000000100000000000000000000000000000001000000000000

Round 3:0000000000000000000000100000000000000000000000000000001000000000
        0000000000000000000000000000001000000000000000000000000000000000
```

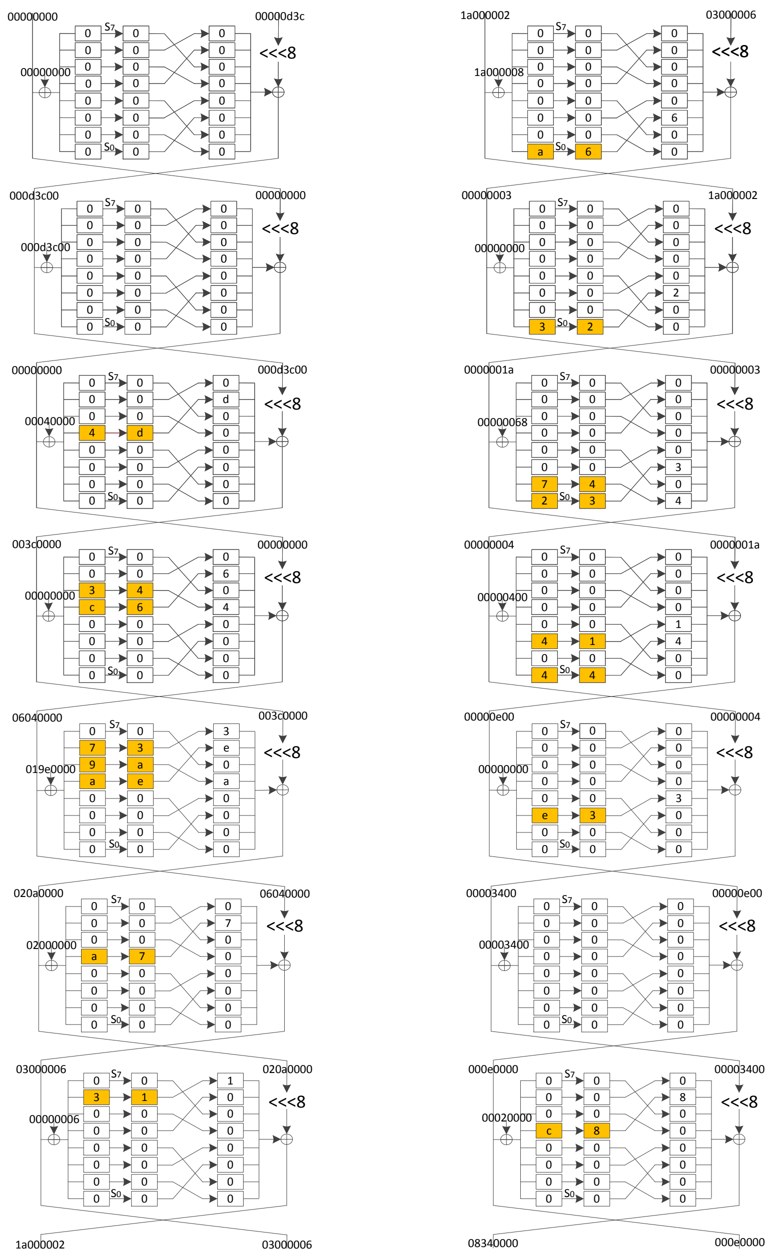
00000000
00000d3c
00000000
<<<8
S7
S0
000d3c00
00000000
000d3c00
00000000
000d3c00
00040000
003c0000
00000000
00000000
06040000
003c0000
019e0000
020a0000
06040000
02000000
03000006
020a0000
00000006
1a000002
03000006
1a000002
03000006
1a000008
00000003
1a000002
00000000
0000001a
00000003
00000068
00000004
0000001a
00000400
00000e00
00000004
00000000
00003400
00000e00
00003400
000e0000
00003400
00020000
08340000
000e0000

Fig. 5: LBlock

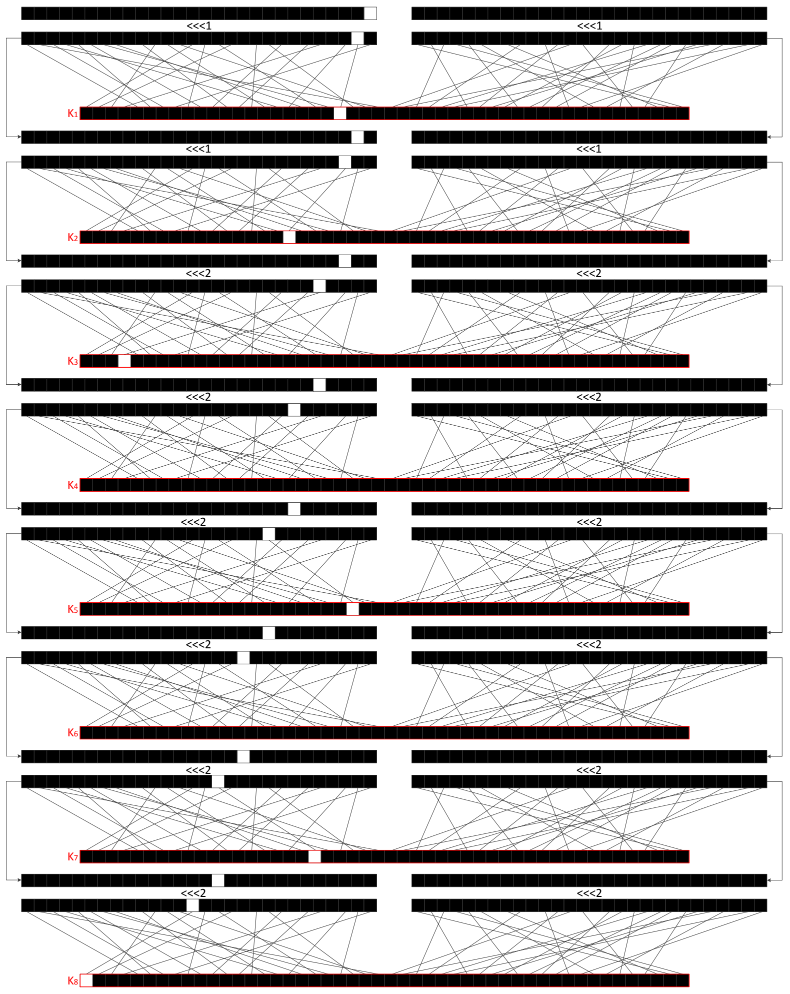

Fig. 6: The key schedule algorithm of DESL

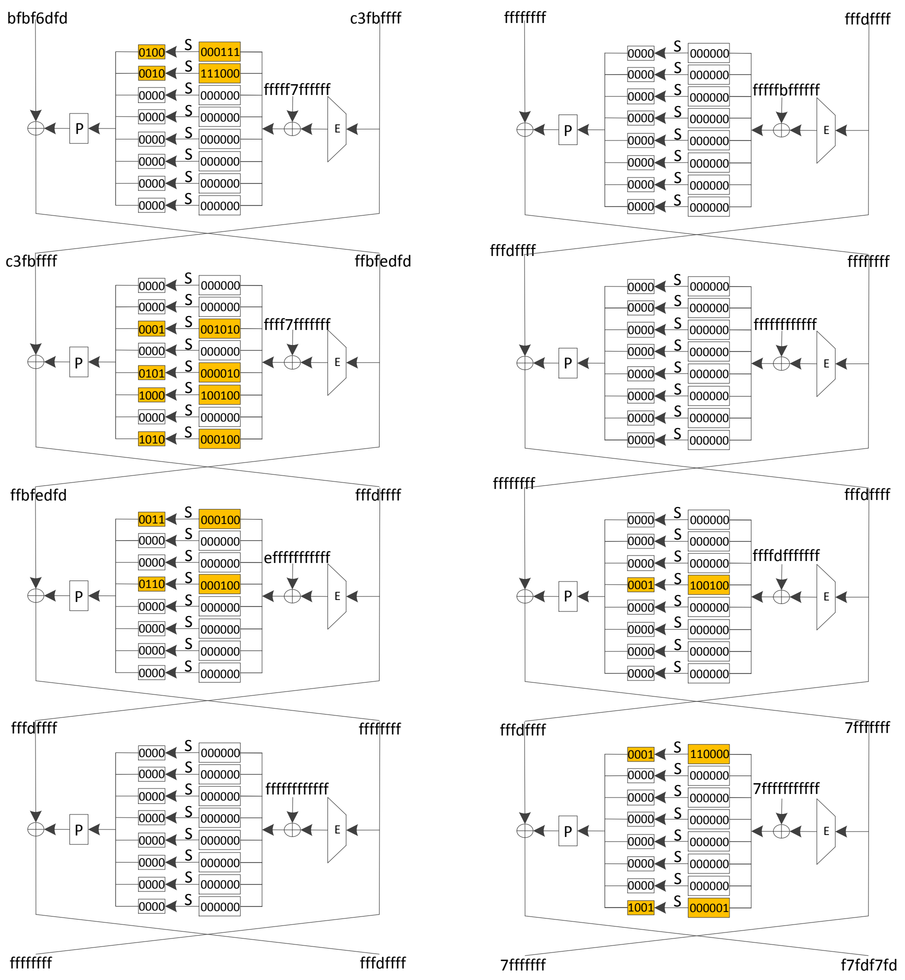
bfbf6dfd
c3fbffff
fffff7ffffff
c3fbffff
ffbfedfd
ffff7fffffff
ffbfedfd
fffdffff
efffffffffff
fffdffff
ffffffff
ffffffffffff
ffffffff
fffdffff
ffffffff
fffdffff
fffffbffffff
fffdffff
ffffffff
ffffffffffff
ffffffff
fffdffff
ffffdfffffff
fffdffff
7fffffff
7fffffffffff
7fffffff
f7fdf7fd
P
E
S

Fig. 7: DESL

```
Round 4:0000000000000000000000000000000001000000000000000000000000000000
        0000000000000000000000000100000000000000000000000000000001000000

Round 5:0000000000000000000000000000100000000000000000000000000000001000
        0000000000000000000000000000000000001000000000000000000000000000

Round 6:0000000000000000000000000000000000000001000000000000000000000000
        0000000000000000000000000000000100000000000000000000000000000001

Round 7:0000000000000000000000000000000000100000000000000000000000000000
        0010000000000000000000000000000000000000001000000000000000000000

Round 8:0000010100000000000000000000000000000000000001000000000000000000
        0000000000000000000000000000000000000100000000000000000000000000

The input and output differences of the S-box layer:
Round 1:
In: 1010000000000000000000000000000000000000000000000000000000000000
Out 0101000000000000000000000000000000000000000000000000000000000000

Round 2:
In: 0000000000000000000000000000000000000000000000000000000000000000
Out:0000000000000000000000000000000000000000000000000000000000000000

Round 3:
In: 0000000000000000000000000001000000000000000000000000000000000000
Out:0000000000000000000000001101000000000000000000000000000000000000

Round 4:
In: 0000001000000000000000000000000000000000000000000000000000000000
Out:0000001100000000000000000000000000000000000000000000000000000000

Round 5:
In: 0000000000000000000000000000000000000000000000000100000000000000
Out:0000000000000000000000000000000000000000000000000101000000000000

Round 6:
In: 0000000000000000000000000000000000000000000000000000000000000000
Out:0000000000000000000000000000000000000000000000000000000000000000

Round 7:
In: 0000000000000000000000000000000000000001000000000000000000000000
Out:0000000000000000000000000000000000000111000000000000000000000000


Input difference before the initial subkey xor:
1010000000000000000001000000000000000000000000000000000000000000
```