Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-oriented Block Ciphers

*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, China.

wp@is.ac.cn

Asiacrypt 2014

Outline

1. **Motivation**
2. **Existing methods for differential characteristic search**
3. **Our method**
4. **Applications and results**

Motivation

- Differential cryptanalysis [Biham, Shamir, 1991] is one of the most powerful attacks on block ciphers

- Finding a good differential characteristic with high probability is the first step in the (related-key) differential attack

Existing methods (I)

- Matsui's Algorithm
  - Mitsuru Matsui, *On correlation between the order of S-boxes and the strength of DES*, Eurocrypt 1994.
  - Branch and Bound approach
  - Original method only applicable in the single-key setting

- Variants of Matsui's Algorithm
  - Alex Biryukov, Ivica Nikolic.: Search for related-key differential characteristics in DES-like ciphers. FSE 2011
  - Brach and Bound approach
  - Applicable in the related-key setting, but only for <u>*linear key schedule algorithm*</u>

- Integer programming based method

Existing methods (II)

- Integer programming based method
  - ✓ applicable both in single-key and related-key settings
  - ✓ can be used to obtain security bounds (bounds of the minimum number of active S-boxes) with respect to differential attack
  - ✓ can not be used to obtain good characteristic directly
  - ✓ not applicable to bit-oriented block ciphers such as PRESENT, SIMON, DES(L), etc.

- Nicky Mouha, Qingju Wang, Dawu Gu, Bart Preneel. *Differential and linear cryptanalysis using mixed-integer linear Programming*. Inscrypt 2011.

- Shengbao Wu, Mingsheng Wang. *Security Evaluation against Differential Cryptanalysis for Block Cipher Structures*, IACR ePrint 2011/551.

Our method: mixed-integer programming based

- Integer programming based method
  - applicable both in single-key and related-key settings
  - can be used to obtain security bounds (bounds of the minimum number of active S-boxes) with respect to differential attack
  - can be used to obtain good characteristics directly
  - applicable to bit-oriented block ciphers such as PRESENT, SIMON, DES(L), etc.

Mixed-integer programming: An example

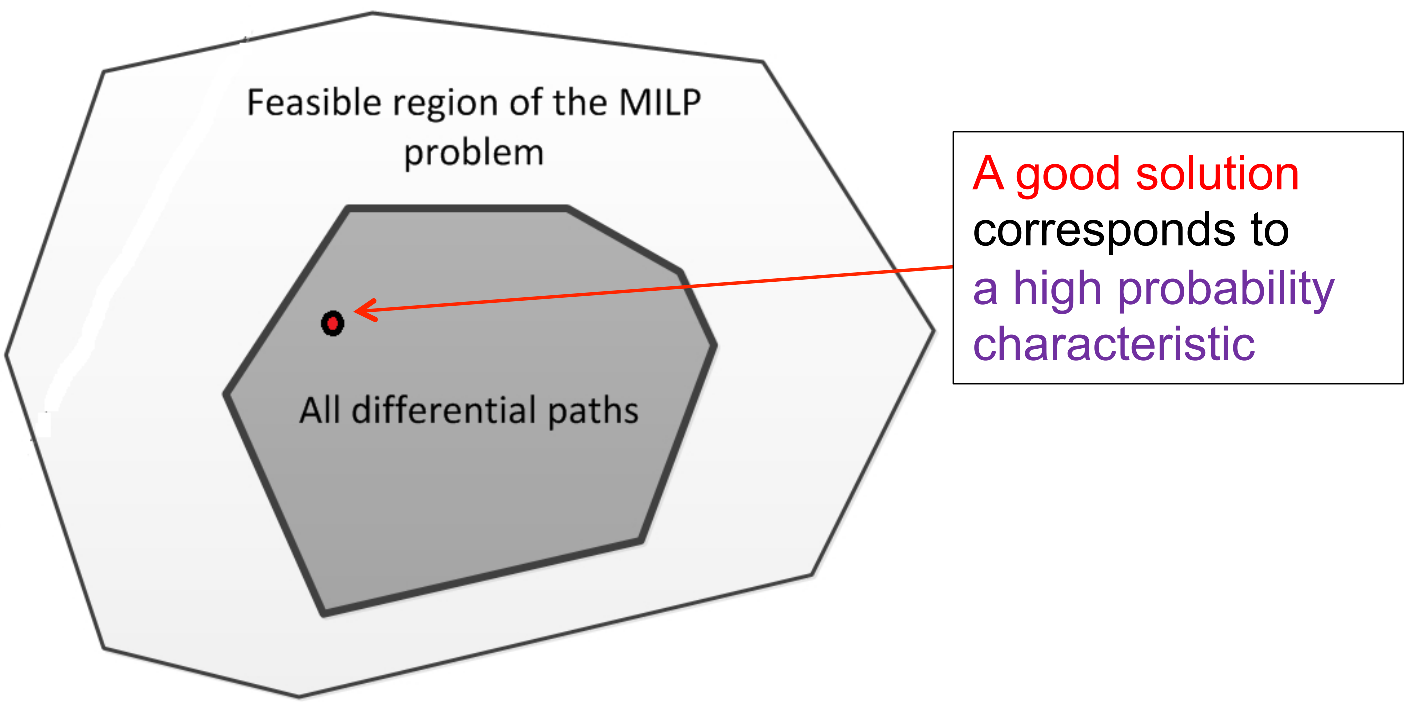
- Mixed-integer linear programming (MILP), an example
  - ✓ Objective function
  - ✓ Feasible region: all solutions satisfy the constraints

$$\min -x_1 + x_2 - 2x_3 + x_4 - x_5$$

$$\begin{aligned} \text{subject to} \quad & \\ x_1 + x_2 &\leq 1 \\ x_1 - 5x_2 + x_3 &\leq 2 \\ 2x_3 + 2x_4 - 4x_5 &\leq 1 \\ x_2 - 2x_4 + x_5 &\leq 0 \\ x &\in \{0, 1\}^5 \end{aligned}$$

Our method: The main idea

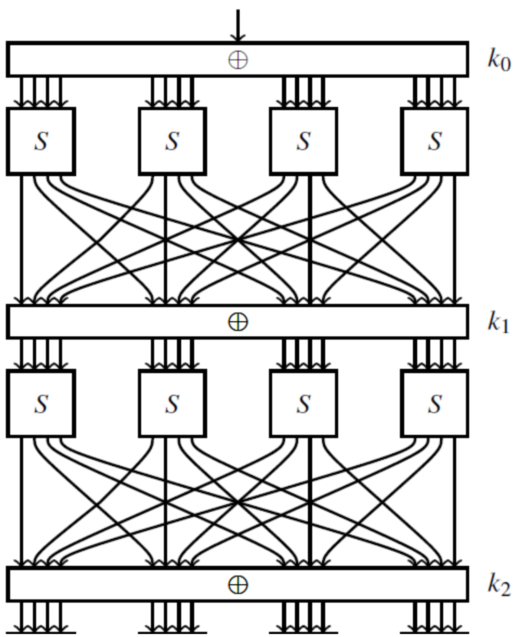
- The main idea of our method
  - ✓ describe the differential behavior of a cipher "at bit-level" by a set of linear inequalities
  - ✓ try to find a characteristic with minimum number of active S-boxes
- Search for high probability characteristic → Extract a good solution from the feasible region of an MILP problem

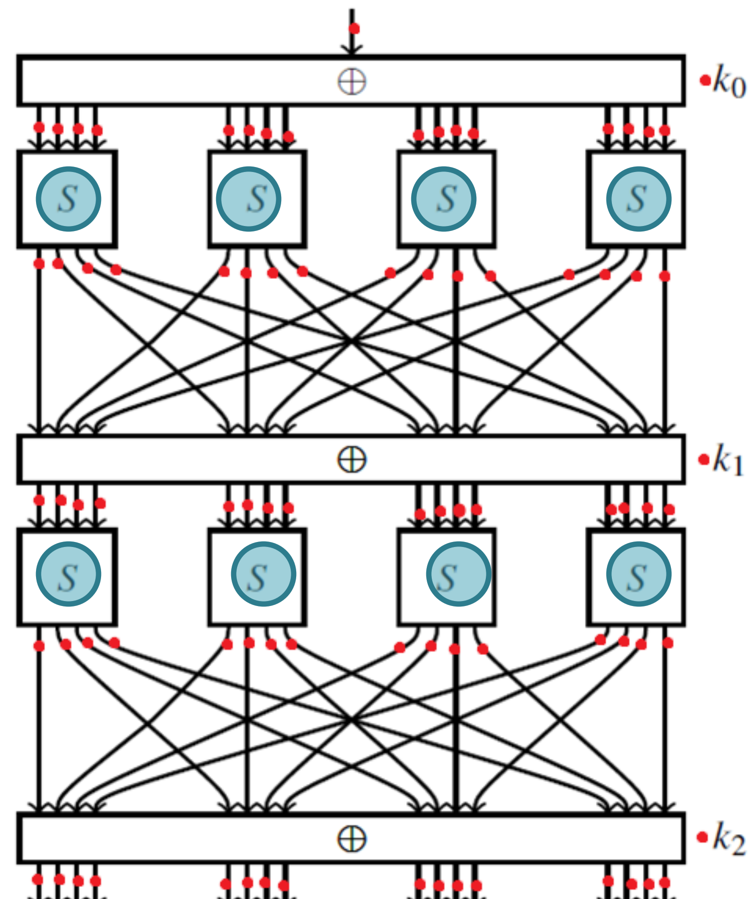


Our method: Modeling technique

Variables involved in our MILP model

- ✓ for every S-box we introduce a new 0-1 variable (represented by a ◯), such that 1 for active and 0 for otherwise
- ✓ for every input and output bit-level difference of every operation we introduce a new 0-1 variable (represeted by a ●)

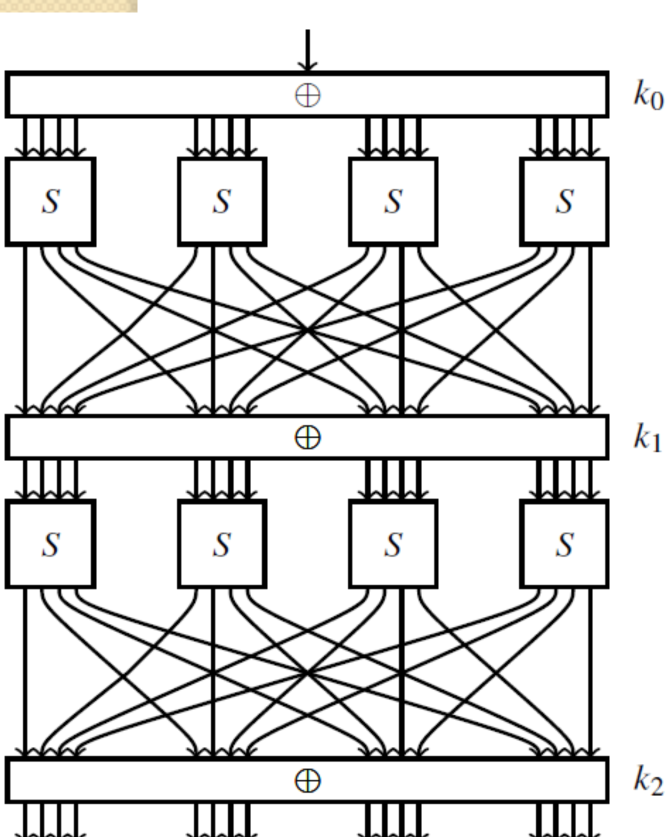


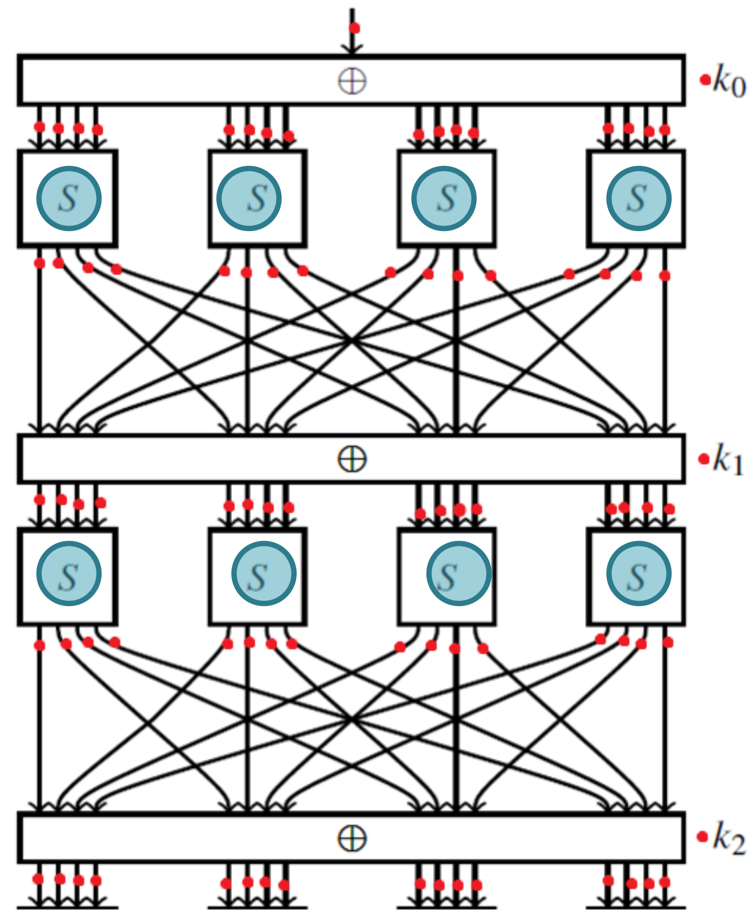


* This toy cipher is taking from Lars R. Knudsen et al's book: *The block cipher The Block Cipher Companion*".

Our method: Modeling technique

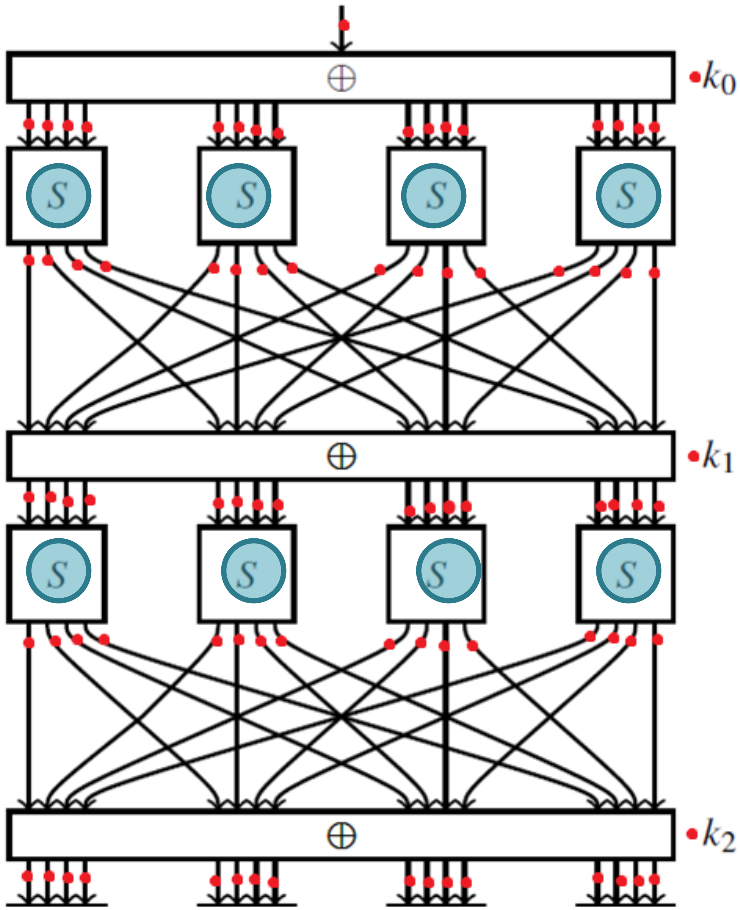
- **Objective function**
  - ✓ Minimize the sum of the varaibles (represented by ◯), that is, minimize the number of active S-boxes.
- **Constraints**
  - ✓ Linear inequalities in the variables represented by ● .





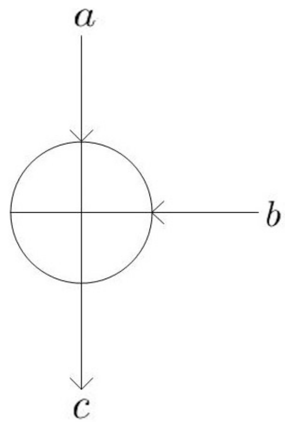
Our method: Constraints generation

- How to describe the constraints imposed (by different operations) on the variables denoted by ● and ◯ with linear inequalities ?



Our method: Constraints generation for XOR

- Constraints imposed on the input and output differences by XOR



$$a = 0, b = 0 \rightarrow c = 0$$
$$a = 0, b = 1 \rightarrow c = 1$$
$$a = 1, b = 0 \rightarrow c = 1$$
$$a = 1, b = 1 \rightarrow c = 0$$

- Constraints (where d is a dummy variable and all variables are 0-1)

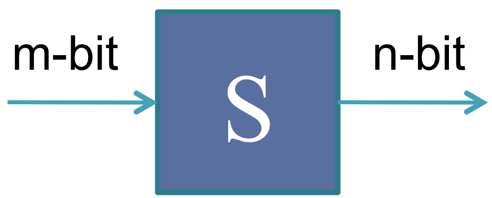
$$\begin{cases} a + b + c \geq 2d \\ d \geq a \\ d \geq b \\ d \geq c \\ a + b + c \leq 2 \end{cases}$$

The first four constraints ($a + b + c \geq 2d$, $d \geq a$, $d \geq b$, $d \geq c$): eliminate the case of one and only one of a, b, and c is 1

The last constraint ($a + b + c \leq 2$): eliminate the case of a=1, b=1 and c=1

Our method: Constraints generation for S-box

- Constraints imposed on the input and output differences by an $m \times n$ S-box (<u>not necessarily invertible</u>)
  - ✓ Let $x_1, x_2, \ldots, x_m$ be the input difference, and $y_1, y_2, \ldots, y_n$ be the output difference
  - ✓ Let A be the variable indicating the activity of the S-box



$$x_1 + \ldots + x_m - A \geq 0$$

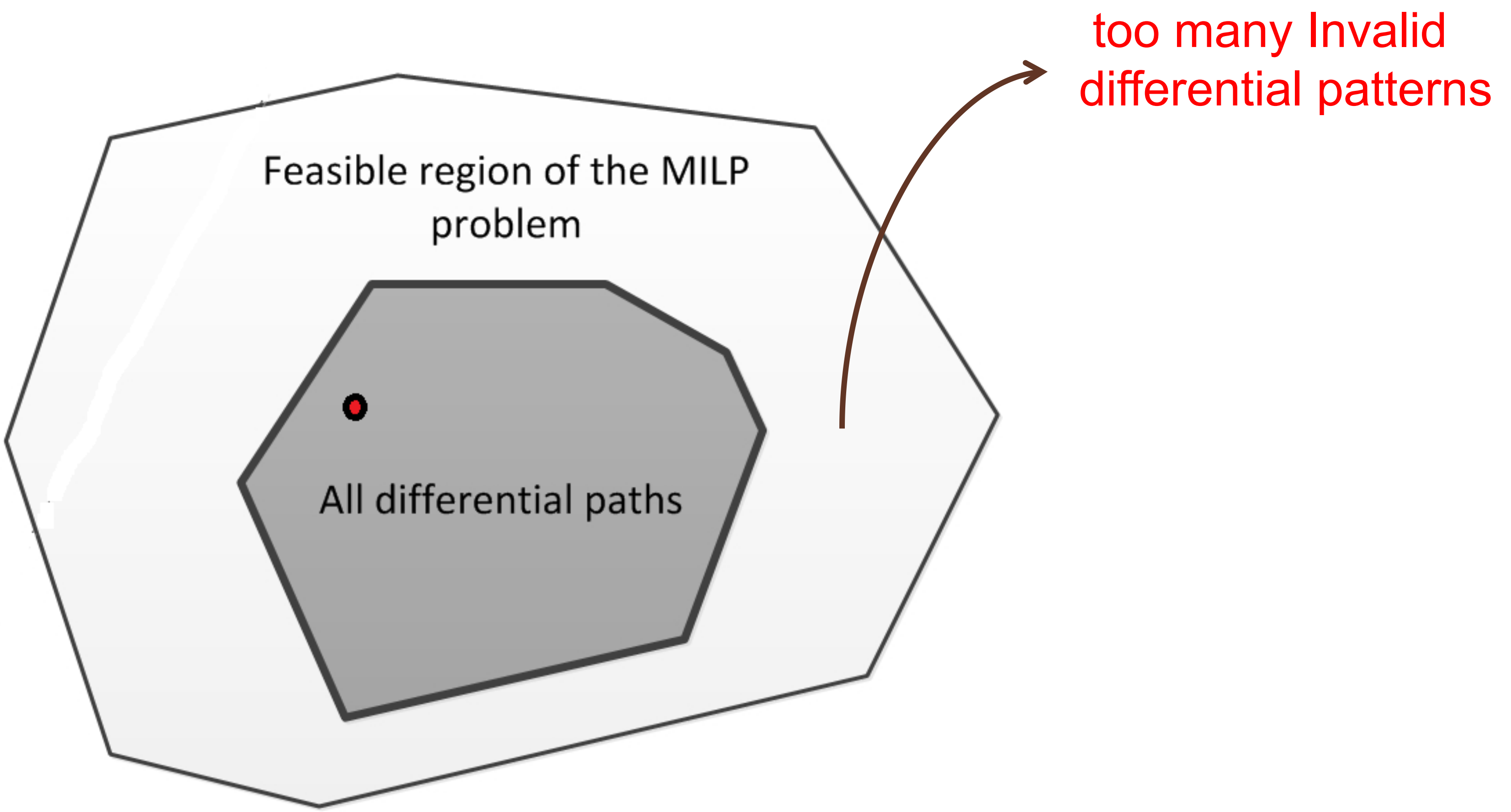
At least one of the input difference bit x_i must be 1 if A = 1.

$$\begin{cases} A - x_1 \geq 0 \\ \ldots \\ A - x_m \geq 0 \end{cases}$$

A must be 1 (active), when anyone of the input difference x_i is 1.

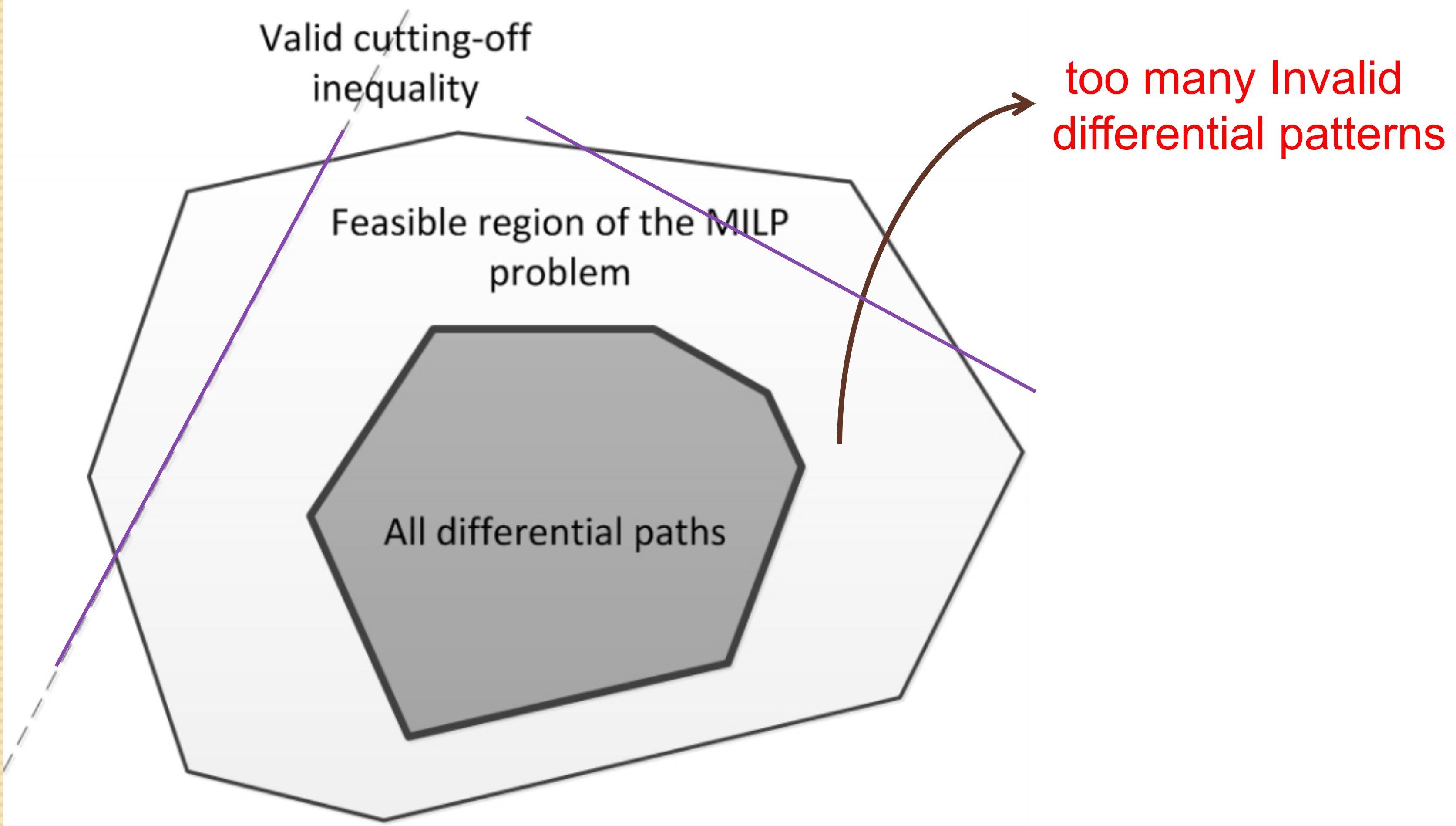
Our method: a more accurate constraints generation

- However, this is too coarse to describe an specific S-box, and result in an feasible region contain many invalid differential patterns



Our method: a more accurate constraints generation

- Hence, we need the so called valid cutting-off inequalities to remove some impossible differential patterns of an specific S-box.



Our method for constraints generation

- Two methods for generating valid cutting-off inequalities for an specific S-box

1. Logical condition modeling

2. Convex hull computation

Method 1

- Logical condition modeling
  - ✓ Assume x, y are 0-1 variables, how to describe the logical condition " x must be 1 when y = 1" ?

$$x - y \geq 0$$

  - ✓ The differentials of some S-boxes has similar properties. For example, the PRESENT S-box.

Fact 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*.*

Method I

- Logical condition modeling
  - ✓ This conditional differential properties can be described by

$$-x_0 + x_1 + x_2 - x_3 - y_3 + 2 \geq 0$$

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

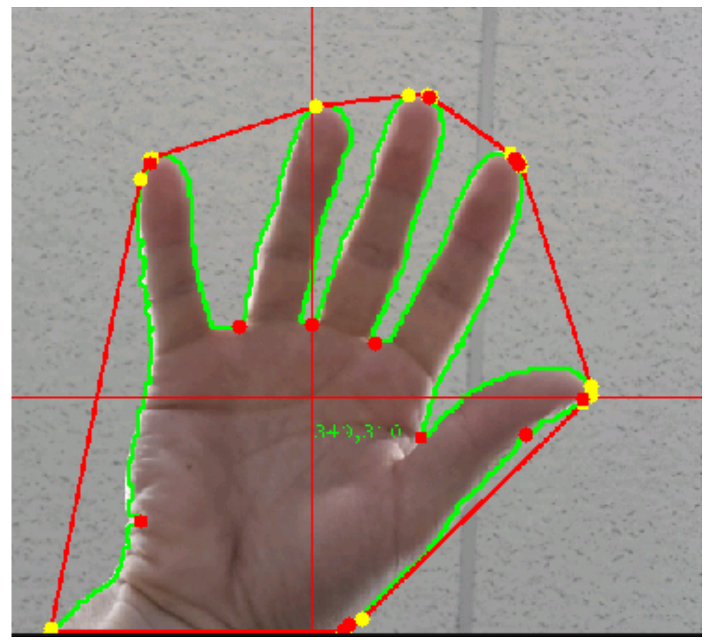
$$\begin{cases} x_3 + y_0 + y_1 + y_2 - y_3 \geq 0 \\ x_3 + y_0 - y_1 + y_2 + y_3 \geq 0 \end{cases}$$

$$-x_3 + y_0 - y_1 + y_2 - y_3 + 2 \geq 0$$

Remove all differential patterns which do not satisfy the differential pattern: 1000→***0, 0001→***1, ***1→0001, ***0→0101

Method II

- Convex hull computation
  - ✓ Convex hull of a set of points in R^n : the smallest convex set that contains these points.



Method II

- Convex hull computation
  - ✓ A convex hull can be represented by a set of linear inequalities

- Treat the set of all possible differential patterns of an S-box as a set of points in R^n. For example, the PRESENT S-box:

{(0, 0, 0, 0, 0, 0, 0, 0), (0, 0, 0, 1, 0, 0, 1, 1), (0, 0, 0, 1, 0, 1, 1, 1),
(0, 0, 0, 1, 1, 0, 0, 1), (0, 0, 0, 1, 1, 1, 0, 1), (0, 0, 1, 0, 0, 0, 1, 1),
(0, 0, 1, 0, 0, 1, 0, 1), (0, 0, 1, 0, 0, 1, 1, 0), (0, 0, 1, 0, 1, 0, 1, 0),
(0, 0, 1, 0, 1, 1, 0, 0), (0, 0, 1, 0, 1, 1, 0, 1), … }

Corresponds to the differential: 0010 → 1101

- Then we can compute the linear inequalities representation of the set of differential patterns

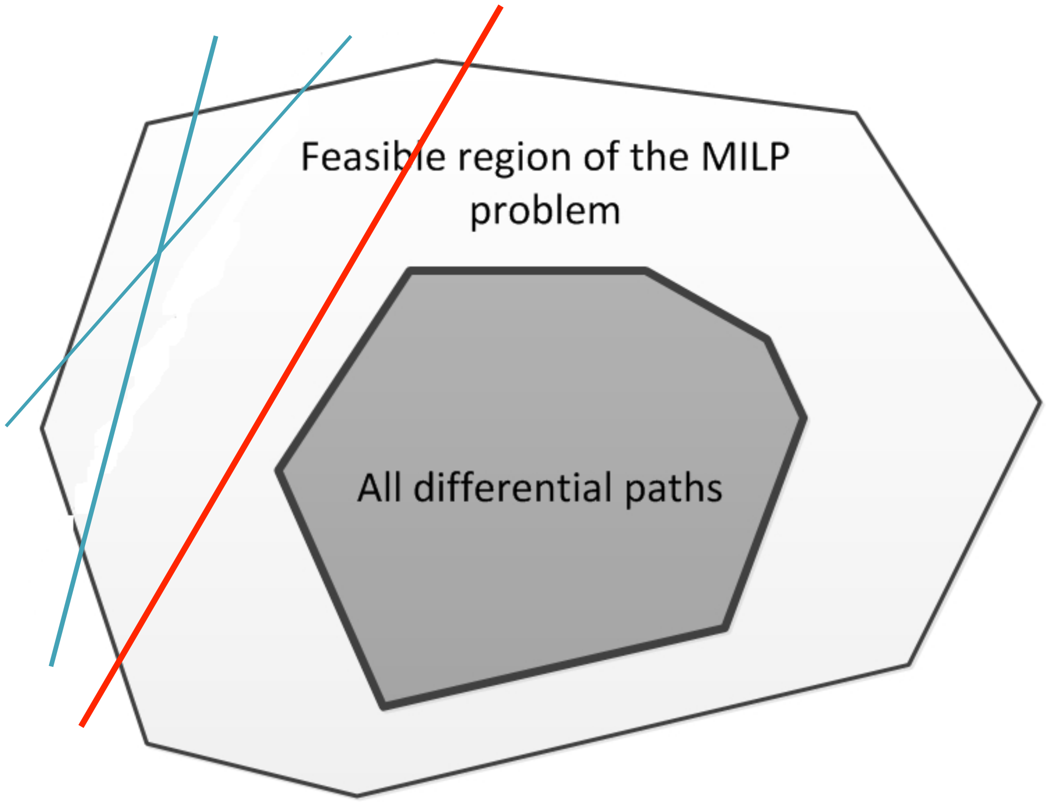
```
( 0,  2, -2,  1, -1, -1, -2, -2,  6)( 0, -2,  2,  1, -2, -1, -1, -2,  6)( 0,  1, -1,  1,  0, -1, -1, -1,  3)(-1,  0,  1, -1, -1, -1,  0, -1,  4)
( 1,  0,  1,  1,  1,  1, -1,  0,  0)( 0,  1,  1,  1, -1,  0,  0,  1,  0)(-1, -1,  0,  1, -1,  0, -1, -1,  4)( 1,  0, -1,  2,  1,  2,  2, -1,  0)
( 1, -1, -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)
```

Linear inequality description of the PRESENT S-box.

Too many inequalities, which will make the MILP problem too difficult to be solved in practical time

Method II

- Convex hull computation
  - ✓ Can we use less inequalities？ Yes！



Method II

- **Convex hull computation**
  - ✓ Can we use less inequalities？ Yes！

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 $\mathcal{H}$

```
l* := None;
X* := X;
H* := H;
O := ∅;
for i ∈ {0, ..., n − 1} do
    l* := The inequality in H* which maximizes the number of removed
          impossible differential patterns from X* ;
    X* := X* − {removed impossible differential patterns by l*};
    H* := H* − {l*};
    O := O ∪ {l*};
end
return O
```

Applications

- Automatic security evaluation with respect to single-key and related-key differential attacks.
  - ✓ obtain the lower bound of the number of active S-boxes of all characteristics
  - ✓ useful in the design of block ciphers

- Automatic search for single-key and related-key differential characteristics
  - ✓ obtain characteristics with high probability
  - ✓ useful in (related-key) differential attack, (related-key) boomerang attack, biclique attack …

Application 1 : Security evaluation

- obtain the lower bound of the number of active S-boxes of all characteristics.
  1. Set the objective function to be the sum of all variables indicating the activities of the S-boxes;
  2. Include the constraints imposed by the operations involved in the cipher;
  3. Require that all variables are 0-1;
  4. Solve the MILP model using the Gurobi optimizer , and the objective value of the optimized solution is a lower bound of the number of active S-boxes.

Gurobi Optimization

http://www.gurobi.com

Application I : Security evaluation

- lower bounds of the number of active S-boxes of the related-key characteristics of PRESENT-80

Rounds	With CDP Constraints		Without CDP Constraints	
	# Active S-boxes	# Time(in seconds)	# Active S-boxes	# Time(in seconds)
1	0	1	0	1
2	0	1	0	1
3	1	1	1	1
4	2	1	2	1
5	3	5	3	3
6	5	16	4	10
7	7	107	6	26
8	9	254	8	111
9	10	522	9	171
10	13	4158	12	1540
11	15	18124	13	8136
12	16	50017	15	18102
13	18	137160*	17	49537*
14	20	1316808*	18	685372*
15	–	$>$ 20days	–	$>$ 20days

There is no related-key characteristic for 12+12=24-round PRESENT-80 with probability higher than $(2^{-2})^{16} \times (2^{-2})^{16} = 2^{-64}$

Warning !

- Such bounds are only valid for *characteristics*, not for *differentials*

Application II : Characteristic search

- obtain characteristics with high probability
  1. Set the objective function to be the sum of all variables indicating the activities of the S-boxes;
  2. Include the constraints imposed by the operations involved in the cipher;
  3. Require that all variables are 0-1;
  4. Solve the MILP model using the Gurobi optimizer, extract a feasible solution when the objective value is small enough;
  5. Check whether the solution is a valid characteristic. <u>If it is invalid, add some valid cutting-off inequalities and go to step 4.</u> If it is valid, we now have a characteristic.

Application II : Characteristic search

- Improved 15-round single-key differential characteristic and differential for SIMON48, a lightweight block cipher designed by NSA.

SIMON48		
Rounds	Left	Right
0	000000001000000000000000	000000100010001000000000
1	000000000010001000000000	000000001000000000000000
2	000000000000100000000000	000000000010001000000000
3	000000000000001000000000	000000000000100000000000
4	000000000000000000000000	000000000000001000000000
5	000000000000001000000000	000000000000000000000000
6	000000100000100000000000	000000000000001000000000
7	000000000010001000000010	000000100000100000000000
8	001000001000001000001000	000000000010001000000010
9	000000000010001000000010	001000001000001000001000
10	000000100000100000000000	000000000010001000000010
11	000000000000001000000000	000000100000100000000000
12	000000000000000000000000	000000000000001000000000
13	000000000000001000000000	000000000000000000000000
14	000000000000100000000000	000000000000001000000000
15	000000000010001000000000	000000000000100000000000

The probability of the above characteristic is 2^{-46}; by considering the differential effect, the probability is $2^{-41.96}$, which is the best result published so far for SIMON48.

Thanks!

Main references:

1. Mouha, Nicky, et al. "*Differential and linear cryptanalysis using mixed-integer linear programming*." *Information Security and Cryptology*. Springer Berlin Heidelberg, 2012.
2. *Sareh Emami, San Ling, Ivica Nikolic, Josef Pieprzyk and Huaxiong Wang*. *The Resistance of PRESENT-80 Against Related-Key Differential Attacks*. Cryptology ePrint Archive, Report 2013/522, 2013.
3. Wu, Shengbao, and Mingsheng Wang. "*Automatic Search of Truncated Impossible Differentials for Word-Oriented Block Ciphers*." *Progress in Cryptology-INDOCRYPT 2012*. Springer Berlin Heidelberg, 2012. 283-302.
4. Bouillaguet, Charles, Patrick Derbez, and Pierre-Alain Fouque. "*Automatic search of attacks on round-reduced AES and applications*." *Advances in Cryptology–CRYPTO 2011*. Springer Berlin Heidelberg, 2011. 169-187.
5. Biryukov, Alex, and Ivica Nikolić. "*Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to AES, camellia, khazad and others*." *Advances in Cryptology–EUROCRYPT 2010*. Springer Berlin Heidelberg, 2010. 322-344.
6. Wu, Shengbao, and Mingsheng Wang. *Security evaluation against differential cryptanalysis for block cipher structures*. Cryptology ePrint Archive, Report 2011/551, 2011.
7. Alex Biryukov, Arnab Roy, Vesselin Velichkov: *Differential analysis of block ciphers SIMON and SPECK*. In: FastSoftware Encryption – FSE 2014
8. Biryukov, Alex, and Ivica Nikolić: *Search for related-key differential characteristics in DES-like ciphers*. In: Fast Software Encryption – FSE 2011. pp. 18–34. Springer (2011)
9. Knudsen, Lars R., and Matthew Robshaw. *The block cipher companion*. Springer, 2011.