Usage of Mixed Integer Linear Programming in Cryptanalysis of Block Ciphers

Halil İbrahim Kaplan[1][0009-0006-0120-6512]

Cryptanalysis Laboratory, TÜBİTAK BİLGEM, Kocaeli, Turkey
halil.kaplan@tubitak.gov.tr

Abstract. This paper presents a comprehensive approach to the cryptanalysis of block ciphers using Mixed Integer Linear Programming (MILP). By formulating the cipher's components including substitution boxes, linear layers, and key schedules as systems of linear inequalities, MILP enables the automated discovery of optimal cryptanalytic characteristics. Our methodology is demonstrate the MILP modelling through the analysis of the ITUbee cipher under three attack models: differential, linear, and related-key differential cryptanalysis. This work highlights the potential of MILP as a robust, generalizable tool for evaluating cipher security and guiding the design of resilient cryptographic primitives.

Keywords: Cryptanalysis · MILP · Differential Cryptanalysis · Block Ciphers

1 Introduction

Cryptanalysis of block ciphers has long been a cornerstone of cryptographic research, driven by the need to ensure the security of data across various platforms. Block ciphers are essential in securing communication in environments ranging from the Internet of Things (IoT) and RFID systems to Smart Cards and sensor networks. The security requirements for these platforms differ significantly, often necessitating the use of lightweight cryptographic algorithms that can operate efficiently under constraints like limited area, energy, and code size. As a result, traditional ciphers such as the Advanced Encryption Standard (AES) may not be suitable for all applications, highlighting the importance of designing and thoroughly analyzing alternative cryptographic algorithms.

Differential, linear and related key are among the most fundamental techniques in the cryptanalyst's toolkit. These methods have been playing important role in evaluating of the security of block ciphers. However, the complexity and time-consuming nature of these analyses pose significant challenges.

The improvements in the area of cryptanalysis have led to the development of automated and semi-automated tools designed to enhance the efficiency of cryptanalysis. Among these tools, Mixed-Integer Linear Programming (MILP) has emerged as a particularly powerful method for cryptanalysis. MILP offers a flexible and effective approach to modeling and analyzing the security of block ciphers, making it easier to evaluate their resistance to cryptographic attacks.

Organization of the paper. The rest of the paper is organized as follows. Section 2 introduces Linear and mixed-integer linear programming with example. Section 3 defines ITUbee algorithm. Section 4 studies how to model block cipher components and. Section 6 introduces our MILP models for differantial, linear, relatekey differential cryptanalysis

2 LP and MILP

2.1 Linear Programming

Linear Programming (LP) is a fundamental mathematical optimization technique that was first introduced in the 1940s by George Dantzig [2], although its development continued into the 1950s. LP has become an essential tool across various fields, including economics, operations research, engineering, logistics, and the natural sciences. The primary goal of LP is to optimize (maximize or minimize) a linear objective function subject to a set of linear constraints.

Mathematical Formulation The standard form of a linear programming problem can be expressed mathematically as follows:

Maximize (or Minimize)

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

Subject to:

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\ x_1, x_2, \ldots, x_n \geq 0 \end{gathered}$$

Here, $x_1, x_2, \ldots, x_n$ represent the decision variables, $c_1, c_2, \ldots, c_n$ are the coefficients of the objective function, a_{ij} are the coefficients of the constraints, and $b_1, b_2, \ldots, b_m$ are the bounds of the constraints. The decision variables are typically constrained to be non-negative, although this condition can be relaxed or modified depending on the problem context.

The objective function is linear, meaning that the relationship between the decision variables and the function value is additive and proportional. The constraints are also linear, implying that each constraint defines a half-space in the decision variable space. The feasible region, which is the set of all points that satisfy the constraints, is a convex polytope. The optimal solution, if it exists, lies on a vertex of this polytope.

2.2 Mixed-Integer Linear Programming

Mixed-Integer Linear Programming (MILP) extends the framework of Linear Programming by incorporating integer constraints on some or all of the decision variables.

Mathematical Formulation The mathematical formulation of an MILP problem is similar to that of an LP problem, but with the addition of integer constraints on certain decision variables:

Maximize (or Minimize)

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

Subject to:

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\ x_1, x_2, \ldots, x_p \in Z \\ x_{p+1}, x_{p+2}, \ldots, x_n \geq 0 \end{gathered}$$

In this formulation, the first p decision variables $x_1, x_2, \ldots, x_p$ are constrained to take integer values, while the remaining $n-p$ variables $x_{p+1}, x_{p+2}, \ldots, x_n$ are real-valued.

Computational Complexity and Solution Methods MILP problems are generally NP-hard, meaning that there is no known algorithm that can solve all instances of MILP problems in polynomial time. The presence of integer variables introduces combinatorial complexity, making MILP significantly more challenging than LP.

Despite this complexity, several algorithms have been developed to solve MILP problems efficiently in practice. Some example of theese algorithms are Branch and Bound [6], Branch and Cut [10], Branch and Price [1]. These algorithms are implemented in various commercial and open-source MILP solvers, such as CPLEX [12], Gurobi [4], and GLPK [7], which are widely used in industry and academia.

Toy Example of MILP Modeling A company produces two types of products, **Product A** and **Product B**. The goal is to maximize profit while considering production capacity, resource availability, and a minimum production requirement. The constraints involve available labor hours, machine hours, and minimum production levels.

Variables Let:

- x be the number of units produced of **Product A**.
- y be the number of units produced of **Product B**.

Both x and y are integer variables.

Objective Function Maximize the total profit: Maximize $Z = 40x + 30y$ where:

- 40 is the profit per unit of **Product A**,
- 30 is the profit per unit of **Product B**.

Constraints

1. **Labor Constraint**: Each unit of **Product A** requires 2 hours of labor, and each unit of **Product B** requires 1 hour of labor. The total available labor hours are 100. $2x + y \leq 100$
2. **Machine Hours Constraint**: Each unit of **Product A** requires 1 hour of machine time, and each unit of **Product B** requires 3 hours of machine time. The total available machine hours are 90. $x + 3y \leq 90$
3. **Minimum Production Constraint**: At least 20 units of **Product A** must be produced to meet market demand. $x \geq 20$
4. **Non-Negativity Constraint**: The number of units produced cannot be negative. $x \geq 0, \quad y \geq 0$

Formulation The entire problem can be formulated as follows:
Maximize

$$Z = 40x + 30y$$

subject to:

$$\begin{gathered} 2x + y \leq 100 \\ x + 3y \leq 90 \\ x \geq 20 \\ y \geq 0 \end{gathered}$$

Modelling and Result We will use Sagemath and GLPK solver for modeling Toy example. After running code in Listing 1.1 . We found that Maximum profit 2160 can be achieved by producing 42 A product and 16 B product.

Listing 1.1. Sagemath MILP modelling of toy example

```
1
2  p.<x,y> = MixedIntegerLinearProgram
3  (maximization=True, solver = "GLPK")
4
5  p.set_objective(40*x[0]+ 30*y[0])
6
7  p.add_constraint(2*x[0] + y[0] == 100)
8  p.add_constraint(x[0] + 3*y[0] == 90)
9  p.add_constraint(x[0] >= 20)
10 p.add_constraint(y[0] >= 0)
11
12 print(p.solve())
13 print(p.get_values(x,y))
```

3 ITUbee Algorithm

ITUbee [5] is a lightweight software-oriented block cipher specifically designed for resource-constrained devices which include a microcontroller and have a limited battery power such as sensor nodes in wireless sensor networks.

The cipher has Fiestel structure which consists of 20 rounds. İt has 80 bits block size and 80 bits key size. The round structure of the cipher is given in Figure 1. Cipher also has a key whitening layer in the beginning and end of the 20 rounds. The encryption process of the cipher is given in Algorithm 1. Round constants of the cipher is given in Table 1

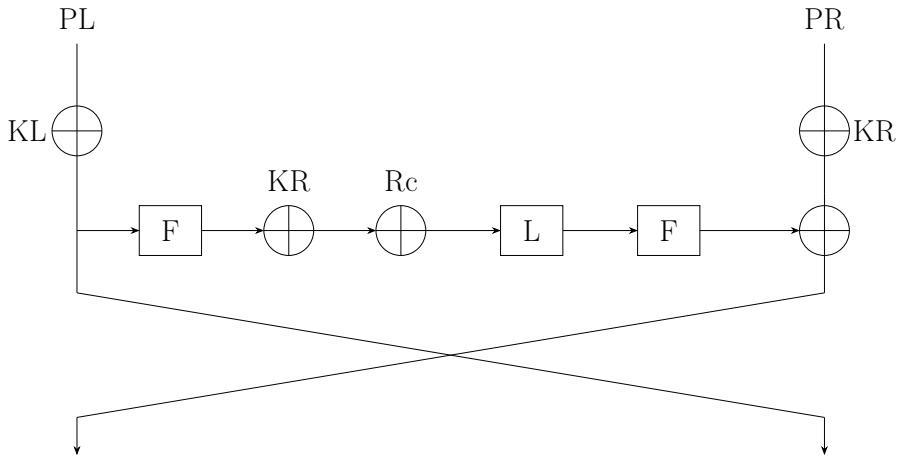


Fig. 1. ITUbee round structure

Algorithm 1 Pseudo code for ITUbee encryption

$X_1 \leftarrow P_L \oplus K_L \quad X_0 \leftarrow P_R \oplus K_R$ **for** $i = 1$ *to* 20 **do**
  if $i \in \{1, 3, \ldots, 19\}$ **then**
    $RK \leftarrow K_R$
  else
    $RK \leftarrow K_L$
  $X_{i+1} \leftarrow X_{i-1} \oplus F(L(RK \oplus RC_i \oplus F(X_i)))$ `// Round constant is added to the rightmost 16 bits`
$C_L \leftarrow X_{20} \oplus K_R \quad C_R \leftarrow X_{21} \oplus K_L$

Round	Round Constant (Hex)	Round	Round Constant (Hex)
1	0x1428	11	0x0a1e
2	0x1327	12	0x091d
3	0x1226	13	0x081c
4	0x1125	14	0x071b
5	0x1024	15	0x061a
6	0x0f23	16	0x0519
7	0x0e22	17	0x0418
8	0x0d21	18	0x0317
9	0x0c20	19	0x0216
10	0x0b1f	20	0x0115

Table 1. Round Constants of ITUbee algorithm

3.1 L Function

The L funtion which has branch number 4 is computed as,
$L(a \parallel b \parallel c \parallel d \parallel e) = (e \oplus a \oplus b) \parallel (a \oplus b \oplus c) \parallel (b \oplus c \oplus d) \parallel (c \oplus d \oplus e) \parallel (d \oplus e \oplus a)$
This function can also be represented as matrix multiplication using rotational matrix:

$$\begin{pmatrix} 1&1&0&0&1 \\ 1&1&1&0&0 \\ 0&1&1&1&0 \\ 0&0&1&1&1 \\ 1&0&0&1&1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} e \oplus a \oplus b \\ a \oplus b \oplus c \\ b \oplus c \oplus d \\ c \oplus d \oplus e \\ d \oplus e \oplus a \end{pmatrix}$$

3.2 F Function

The F function is computed as,

$$F(X) = S(L(S(X)))$$

S is the S-box used in AES [3]. S is the only non-linear structure in the ITUbee algorithm. 40 bit input of S is splitted into 8 bits then enters the s-box.

$$S(a \parallel b \parallel c \parallel d \parallel e) = s[a] \parallel s[b] \parallel s[c] \parallel s[d] \parallel s[e]$$

3.3 Key Schedule

ITUbee has no key schedule. The algorithm splits the 80 bit key into two 40 bit parts K_L and K_R and uses the right part of the key in odd rounds and the left part of the keys in even rounds.

4 Construction of MILP Modeling

4.1 Branch Number

Branch number is a critical parameter used to evaluate the diffusion properties of cipher's components. The concept of branch number originates from the study of linear codes and is particularly relevant when assessing the security and strength of cryptographic primitives.

Definition and Calculation Formally, if we consider a transformation $\mathcal{T}$ applied to an n-element vector, the branch number is given by:

$$B = \min_{\mathbf{x} \neq \mathbf{0}} \{\mathrm{wt}(\mathbf{x}) + \mathrm{wt}(\mathcal{T}(\mathbf{x}))\},$$

where $\mathrm{wt}(\mathbf{x})$ denotes the Hamming weight of the vector $\mathbf{x}$, i.e., the number of non-zero elements in $\mathbf{x}$.

The branch number is significant in cryptanalysis because it provides a lower bound on the diffusion introduced by a transformation. A higher branch number indicates better diffusion, which implies that any single bit/byte difference in the input will affect a larger portion of the output thereby increasing the complexity of differential and linear attacks.

Example: Branch Number of AES's MDS Matrix For AES [3], the MDS matrix used in the MixColumns operation has a branch number 5. This means that minimum number of total active byte of input and output of the MDS is 5. Table 2 shows possible active byte patterns of the AES MDS matrix.

Input Active Bytes	Output Active Bytes
1	4
2	3 / 4
3	2 / 3 / 4
4	1 / 2 / 3 / 4

Table 2. Possible active byte patterns of the AES MDS matrix

Role of Branch Number in MILP Modeling Branch number is encoded directly into the MILP model, ensuring that any feasible solution must respect the minimum branch number criteria.In this way, the MILP model we obtain will give us the correct characteristic for the algorithm.

4.2 H-representation

The H-representation, also known as the half-space representation, describes a polyhedron as the intersection of a finite number of half-spaces, each defined by a linear inequality of the form $\mathbf{a}^T\mathbf{x} \leq b$, where $\mathbf{a}$ is a vector of coefficients, $\mathbf{x}$ is the vector of variables, and b is a scalar. This representation is integral to the MILP modeling of block ciphers as it encapsulates the constraints from the cipher's operations, including the behavior of S-boxes, linear layers, and key additions.

In the context of cryptanalysis, using the H-representation allows complex relationships between variables to be expressed linearly, enabling MILP solvers to explore the solution space efficiently. This approach is especially crucial when identifying optimal characteristics for differential or linear attacks.

Example of H-representation Consider linear function L which takes 6 bit input, 6 bit output and computed as:

$$L(a, b, c) = (a \oplus b, a \oplus c, b \oplus c) \text{ where a,b,c are 2 bit values}$$

H-representation of this function can be found in 2 steps with using Sagemath polyhedra library. [11]

1. We scan the outputs for all possible inputs and give a value of 0 if the output value is 0 and 1 if it is different from zero. Possible input and output values of the L function can be seen in Table 3.
2. These points are given as input to the sage polyhedra library and the inequalities given in table 4 are obtained.

In table 4 , $(1, 1, 1, -1, -1, -1)x + 1 \geq 0$ means that $x_0 + x_1 + x_2 - x_3 - x_4 - x_5 + 1 \geq 0$ where x_0 , x_1 and x_2 are input differences, x_3 , x_4 and x_5 are output differences.

4.3 Milp Modeling of XOR Operation

We will use modeling defined in [9]. Lets x_{in1} and x_{in2} be input and x_{out} be output difference of xor operation. Branch number is defined in Section 4.1. Branch number of xor operation is 2. We use a dummy variable d to add the branch number to the equation system. The dummy variable is equal to 0 if x_{in1}, x_{in2} and x_{out} are equal to 0, otherwise it is equal to 1. Using this information, the xor operation can be expressed with the following equations.

$$\begin{gathered} x_{in1} + x_{in2} + x_{out} \geq 2d \\ d \geq x_{in1} \\ d \geq x_{in2} \\ d \geq x_{out} \end{gathered}$$

Input Differences	Output Differences
0 0 0	0 0 0
0 0 1	0 1 1
0 1 0	1 0 1
0 1 1	1 1 0
0 1 1	1 1 1
1 0 0	1 1 0
1 0 1	1 0 1
1 0 1	1 1 1
1 1 0	0 1 1
1 1 1	0 0 0
1 1 1	0 1 1
1 1 0	1 1 1
1 1 1	1 0 1
1 1 1	1 1 0
1 1 1	1 1 1

Table 3. Input and Output Differences of L function

4.4 Milp Modeling of Linear Transformation

We will also use modeling defined in [9]. Lets $x_{in1}, x_{in2}, ..., x_{inM}$, be input differences and $x_{out1}, x_{out2}, ..., x_{outM}$, be output difference of linear transformation. We will use dummy variable d like in XOR modelling and we have branch number B of Linear transformation. Then we can define linear transformation with following equations.

$$x_{in1} + x_{in2} + x_{inM} + ... + x_{out1} + x_{out2} + ... + x_{outM} \geq B * d$$
$$d \geq x_{in1}$$
$$d \geq x_{in2}$$
$$...$$
$$d \geq x_{inM}$$
$$d \geq x_{out1}$$
$$d \geq x_{out2}$$
$$...$$
$$d \geq x_{outM}$$

In block ciphers, any nonlinear structure can be considered as a linear transformation. For example, xor operation or matrix multiplication are linear transformations. When the branch number value is given as 2 in the equation above, the equations given for XOR operation are obtained.

4.5 Milp Modeling of S-box

Since the main purpose of this article is to show that the algorithm is secure by calculating the minimum number of active s-boxes, s-boxes are not modeled in the MILP models of cryptographic attacks. The reason for this is that if the input

No.	Inequality
1	$(0,0,-1,0,0,0)x+1\geq 0$
2	$(0,-1,0,0,0,0)x+1\geq 0$
3	$(-1,0,0,0,0,0)x+1\geq 0$
4	$(0,0,0,-1,0,0)x+1\geq 0$
5	$(0,0,0,0,-1,0)x+1\geq 0$
6	$(0,0,0,0,0,-1)x+1\geq 0$
7	$(0,-1,1,0,0,1)x+0\geq 0$
8	$(0,0,0,-1,1,1)x+0\geq 0$
9	$(0,1,-1,0,0,1)x+0\geq 0$
10	$(0,0,0,1,-1,1)x+0\geq 0$
11	$(1,0,-1,0,1,0)x+0\geq 0$
12	$(0,0,0,1,1,-1)x+0\geq 0$
13	$(1,0,1,0,-1,0)x+0\geq 0$
14	$(0,1,1,0,0,-1)x+0\geq 0$
15	$(1,1,1,-1,-1,-1)x+1\geq 0$
16	$(1,1,0,-1,0,0)x+0\geq 0$
17	$(-1,0,1,0,1,0)x+0\geq 0$
18	$(-1,1,0,1,0,0)x+0\geq 0$
19	$(1,-1,-1,1,1,-1)x+1\geq 0$
20	$(1,-1,0,1,0,0)x+0\geq 0$
21	$(-1,1,-1,1,-1,1)x+1\geq 0$
22	$(-1,-1,1,-1,1,1)x+1\geq 0$

Table 4. H-representation of L function

of the s-box is active, its output is also active, and similarly, passive input gives passive output. Therefore, the model of the s-box has no effect on the minimum number of active s-boxes. However, if the purpose of the modeling is not to find the minimum active s-box but to find the best characteristic, the s-box must be modeled. Therefore, the s-box model defined in [13] is given below as a reference for the reader.

Let S be a 4x4 bijective S-box whichs inputs are $x_{in1}, x_{in2}, x_{in3}, x_{in4}$ and outputs are $x_{out1}, x_{out2}, x_{out3}, x_{out4}$. A is equal to 1 if and of the input bits are active. Otherwise A is 0. Then S can be defined by following equations.

$$
\begin{gathered}
x_{in1} - A \leq 0 \\
x_{in2} - A \leq 0 \\
x_{in3} - A \leq 0 \\
x_{in4} - A \leq 0 \\
x_{in1} + x_{in2} + x_{in3} + x_{in4} - A \geq 0 \\
4 * (x_{in1} + x_{in2} + x_{in3} + x_{in4}) - (x_{out1} + x_{out2} + x_{out3} + x_{out4}) \geq 0 \\
4 * (x_{out1} + x_{out2} + x_{out3} + x_{out4}) - (x_{in1} + x_{in2} + x_{in3} + x_{in4}) \geq 0
\end{gathered}
$$

4.6 Milp Modeling of three-forked branch

We will use modeling defined in [9]. This model will be used in linear cryptanalysis. Lets y_{in} be input and x_{out1} and x_{out2} be output mask of three-forked branch. Modeling is similar to modeling xor operation. We use a dummy variable d which is equal to 0 if x_{in}, x_{out1} and x_{out2} are equal to 0, otherwise it is equal to 1. Using these information, the three-forked branch can be expressed with the following equations.

$$\begin{gathered} x_{in} + x_{out1} + x_{out2} \geq 2d \\ d \geq y_{in} \\ d \geq y_{out1} \\ d \geq y_{out2} \end{gathered}$$

5 Differential Cryptanalysis of ITUbee

ITUbee algorithm has Fiestel structure so we divide input as a two variables RO and $R1$. In model, we define them as a binary variables. After each operation in the cipher, if active byte can be passive or passive byte can be active we define new binary. In differential cryptanalysis, S-box does not change activity of the byte because S-box takes byte and if input byte is active, output byte is also active. It is same for inactive bytes. Therefore we don't define new variables after s-box operations. L operations are changes the active bytes so after them we define new variable. XOR operation changes active byte if it XOR two binary variables. In differential cryptanalysis we do not give active values to the keys therefore key xor's does not change the active byte. Skech of the 1 round modelling can be seen in the Figure 2.

In MILP model we are trying to find best characteristic that makes minimum number of active s-box so we minimize binary values before and after s-box operation in the MILP model. ITUbee algorithm uses AES's [3] s-box. AES s-box have best input-output difference 2^{-6}. In our model we found that best characteristic in 3 round have minimum 16 active s-box. Therefore best probability for differential characteristic will be $(2^{-6})^{16} = 2^{-96}$ which is not usable for differential attack. With this result we can say that 3 round of the algorithm is secure against the differential cryptanalysis. Our result is same for differential with [5]. Best characteristic found with the model can be seen in Table 2

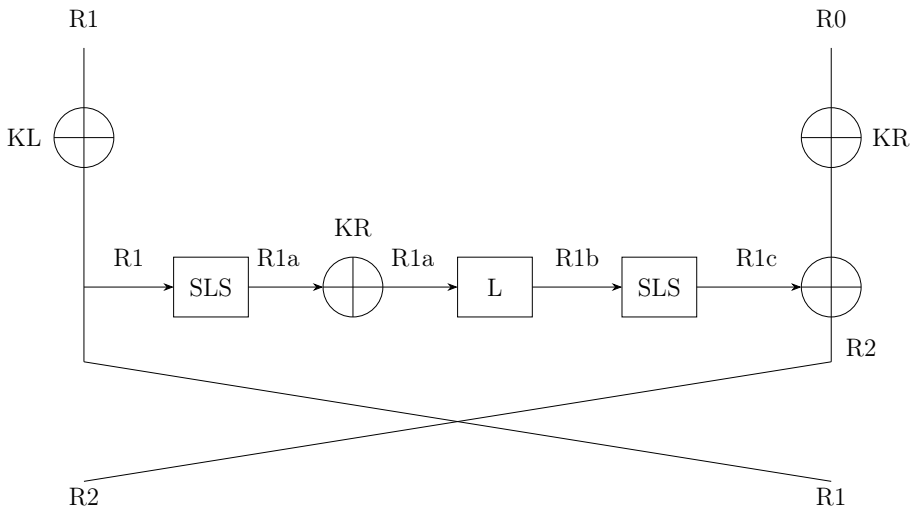


Fig. 2. ITUbee MILP differential cryptanalysis sketch

Stage	Binary Index	Characteristic
1. round	R1	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R0	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
1. round	R1a	{0: 1.0, 1: 0.0, 2: 0.0, 3: 1.0, 4: 1.0}
middle values	R1b	{0: 0.0, 1: 1.0, 2: 1.0, 3: 0.0, 4: 1.0}
	R1c	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
2. round	R2	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R1	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
2. round	R2a	{0: 1.0, 1: 0.0, 2: 0.0, 3: 1.0, 4: 1.0}
middle values	R2b	{0: 0.0, 1: 1.0, 2: 1.0, 3: 0.0, 4: 1.0}
	R2c	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
3. round	R3	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
input	R2	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
3. round	R3a	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
middle values	R3b	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
	R3c	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
4. round	R4	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R3	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}

Table 5. Best 3 round differential characteristic of ITUbee algorithm

6 Linear Cryptanalysis of ITUbee

Linear cryptanalysis model is very similar to the differential model. Only difference is XOR and three-forked branch changes their role in the model. In linear model we add constraints for three-forked branch and pass the XOR operation. Since middle of the fiestel structure is symmetric and XOR and three-forked

branch have same constraints, same model with differential cryptanalysis can be also used for linear cryptanalysis. Sketch of the one round MILP model of the linear cryptanalysis is given in Figure 3

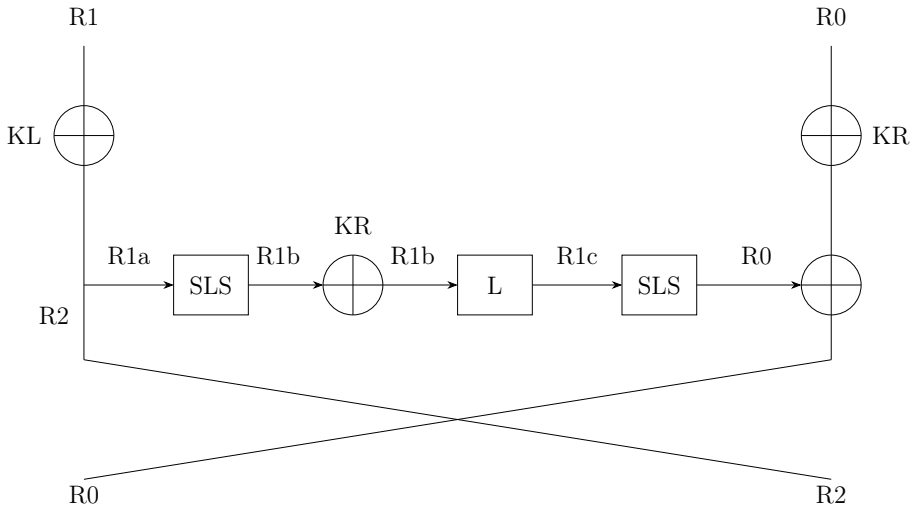


Fig. 3. ITUbee MILP linear cryptanalysis sketch

Best linear bias for ITUbee S-box input-output mask is 2^{-4}. In our MILP model we found that minimum number of active s-box for 3 consecutive round is 16. We use piling up lemma in [8] for calculating 3 round probability. 3 round probability is $2^{15} * (2^{-4})^{16} = 2^{-49}$. Linear cryptanalysis is known-plaintext attack. Number of required plaintexts for the three round attack is $(2^{-49})^{-2} = 2^{96}$ which is more than all possible plaintexts for the cipher. As a conclusion, we can say that 3 round of the ITUbee algorithm is secure against linear cryptanalysis.

7 Related-key Differential Cryptanalysis of ITUbee

For related key model, unlike the differential model, we must also take into account the keys. We also define binary variables for the keys and add constraints for activity of at least 1 key bytes.(Otherwise model will never give active bytes for key and it will be same as differential cryptanalysis) In this model we have more middle values than differential and linear model because we also need to chance the variable after the key xor in the middle. Sketch of the one round MILP model of the linear cryptanalysis is given in Figure 4.

We found that 8 round of the algorithm has minimum 16 active s-box. ITUbee algorithm uses AES's [3] s-box. AES s-box have best input-output difference 2^{-6}. Therefore best probability for differential characteristic will be $(2^{-6})^{16} = 2^{-96}$ which is not usable for related key differential attack. Authors of the ITUbee

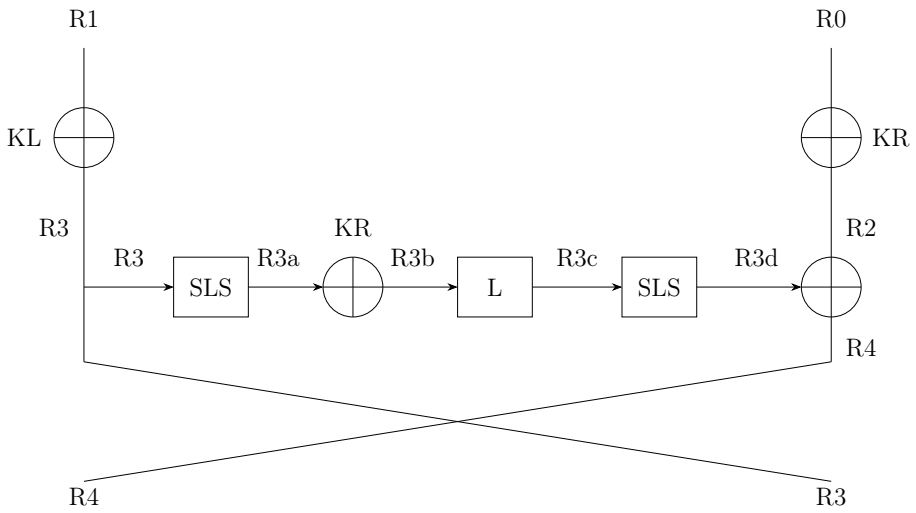


Fig. 4. ITUbee MILP Related-Key Differential Cryptanalysis Sketch

algorithm claims in [5] that algorithm is secure against related key differential cryptanalysis after 10 round. In our model we found that 8 round is enough for security against related key differential cryptanalysis. Best 8 round characteristic found with the model can be seen in Table 6

8 Conclusion

As a conclusion, our results demonstrate that ITUbee exhibits strong resistance to classical differential and linear cryptanalysis for more than three rounds, in alignment with the theoretical claims of the original authors. No effective differential or linear trails with sufficiently high probability or correlation could be found beyond this threshold, affirming the cipher's robustness in this context.

In the case of related-key differential cryptanalysis, our MILP-based analysis reveals a better security margin compared to what was originally proposed. While the designers asserted security against related-key differential attack after 10 rounds, our models shows that algorithm is secure against related-key differential cryptanalysis after 8 rounds. This observation provides valuable insight into the actual security margin of the cipher under related-key scenarios and highlights the importance of model-based cryptanalysis during the cipher evaluation process.

The MILP models developed in this work serve as a reusable framework for analyzing other block ciphers with similar structures. Moreover, the results presented here emphasize the utility of MILP in confirming or challenging security claims and in revealing potential structural vulnerabilities that may not be immediately apparent through manual analysis alone.

Overall, this study reinforces the value of MILP as a precise, automatable, and interpretable method for evaluating security of cryptographic algorithms. All MILP models we developed can be found at `https://github.com/KaplanHalil/Milp/tree/main/ITUBEE` and discovered characteristics are included within the thesis to support further research and reproducibility.

References

[1] Cynthia Barnhart et al. "Branch-and-price: Column generation for solving huge integer programs". In: *Operations Research* 46.3 (1998), pp. 316–329. DOI: 10.1287/opre.46.3.316.

[2] George B Dantzig. "Linear programming". In: *Operations research* 50.1 (2002), pp. 42–47.

[3] Morris J Dworkin et al. "Advanced encryption standard (AES)". In: (2001).

[4] LLC Gurobi Optimization. *Gurobi optimizer reference manual*. 2021.

[5] Ferhat Karakoç, Hüseyin Demirci, and A Emre Harmancı. "ITUbee: a software oriented lightweight block cipher". In: *Lightweight Cryptography for Security and Privacy: Second International Workshop, LightSec 2013, Gebze, Turkey, May 6-7, 2013, Revised Selected Papers 2*. Springer. 2013, pp. 16–27.

[6] A. H. Land and A. G. Doig. "An automatic method of solving discrete programming problems". In: *Econometrica: Journal of the Econometric Society* 28.3 (1960), pp. 497–520. DOI: 10.2307/1910129.

[7] Andrew Makhorin. "GLPK (GNU linear programming kit)". In: *http://www.gnu. org/s/glpk/glpk. html* (2008).

[8] Mitsuru Matsui. "Linear cryptanalysis method for DES cipher". In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1993, pp. 386–397.

[9] Nicky Mouha et al. "Differential and linear cryptanalysis using mixed-integer linear programming". In: *Information Security and Cryptology: 7th International Conference, Inscrypt 2011, Beijing, China, November 30–December 3, 2011. Revised Selected Papers 7*. Springer. 2012, pp. 57–76.

[10] Manfred Padberg and Giovanni Rinaldi. "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems". In: *SIAM Review* 33.1 (1991), pp. 60–100. DOI: 10.1137/1033003.

[11] Sagemath Sage Developers. *the Sage Mathematics Software System (Version 9.5), 2022*.

[12] IBM ILOG CPLEX Optimization Studio. "IBM Ilog Cplex Optimization Studio". In: *CPLEX Parameters reference* (2016), p. 25.

[13] Siwei Sun et al. "Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES (L) and other bit-oriented block ciphers". In: *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20*. Springer. 2014, pp. 158–178.

Stage	Binary Index	Characteristic
Key	KL	{0: 0.0, 1: 1.0, 2: 0.0, 3: 1.0, 4: 0.0}
	KR	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
1. round	R1	{0: 0.0, 1: 1.0, 2: 0.0, 3: 1.0, 4: 0.0}
input	R0	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
1. round	R2	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
middle values	R3-3a-3b-3c-3d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
2. round	R4	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R3	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
2. round	R4a	{0: 0.0, 1: 1.0, 2: 0.0, 3: 1.0, 4: 0.0}
middle values	R4b-4c-4d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
3. round	R5	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
input	R4	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
3. round middle values	R5a-5b-5c-5d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
4. round	R6	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R5	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
4. round	R6a	{0: 0.0, 1: 1.0, 2: 0.0, 3: 1.0, 4: 0.0}
middle values	R6b-6c-6d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
5. round	R7	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
input	R6	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
5. round	R7a	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
middle values	R7b-7c-7d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
6. round	R8	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R7	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
6. round	R8a	{0: 0.0, 1: 1.0, 2: 0.0, 3: 1.0, 4: 0.0}
middle values	R8b-8c-8d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
7. round	R9	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
input	R8	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
7. round middle values	R9a-9b-9c-9d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
8. round	R10	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}
input	R9	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
8. round	R10a	{0: 0.0, 1: 1.0, 2: 0.0, 3: 1.0, 4: 0.0}
middle values	R10b-10c-10d	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
9. round	R11	{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0}
input	R10	{0: 1.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 1.0}

Table 6. Best 8 round related-key differential characteristic of ITUbee algorithm