Towards Finding Active Number of S-boxes in Block Ciphers Using Mixed Integer Linear Programming

Vikas Tiwari
Acharya Nagarjuna University, Nagarjuna Nagar, Guntur - 522510, Andhra Pradesh, India
C.R. Rao Advanced Institute of Mathematics, Statistics and Computer Science
University of Hyderabad Campus, Prof. CR Rao Road, Hyderabad - 500046, Telangana, India
E-mail: vikas.tiwari2403@gmail.com

Neelima Jampala, Appala Naidu Tentu and Ashutosh Saxena
C.R. Rao Advanced Institute of Mathematics, Statistics and Computer Science
University of Hyderabad Campus, Prof. CR Rao Road, Hyderabad, - 500046, Telangana, India
E-mail: {neelima.jampala, naidunit, saxenaaj}@gmail.com

Keywords: lightweight block cipher, cryptanalysis, MILP, CPLEX solver, differential attack

Received: January 29, 2021

Secure lightweight block ciphers have become an important aspect due to the fact that they are a popular choice for providing security in ubiquitous devices. Two of the most important attacks on block ciphers are differential cryptanalysis [1] and linear cryptanalysis [2]. Calculating the number of active S-boxes is one of the method to examine the security of block ciphers against differential attack. In this paper, we count the minimum number of active S-boxes for several rounds of the lightweight ciphers namely KLEIN, LED and AES. We utilized the method proposed in [9], where calculation of the minimum number of active S-boxes is formulated as a Mixed Integer Linear Programming (MILP) problem. The objective function is to minimize the number of active S-boxes, subject to the constraints imposed by the differential propagation of the cipher. The experimental results are presented in this paper and found to be encouraging.

Povzetek: Predstavljena je metoda na osnovi analize MILP problem za iskanje primernih šifer za ambientalne naprave.

1 Introduction

In recent years, designing cryptographic primitives has gathered attention from the research community which are used in resource constrained devices. This field of research is termed as lightweight cryptography. Lightweight block ciphers are generally used where the hardware resources are limited as well as where power consumption is minimum. Lightweight ciphers are very much effective for the transmission of data which concerns security and speed. Wireless networks are becoming very popular because of their increasing demands in the field of environmental monitoring, defence and healthcare. Typically, some scenarios where the data to be transmitted over unprotected communication links requires the secure and fast transmission of the data and proxy signatures [5]. Design of lightweight ciphers require a trade-off between the efficiency and security parameters.

Linear programming (LP)[17] is the analysis of minimizing or maximizing a linear objective func. $f(\alpha_1, \alpha_2, ..., \alpha_n)$, subject to linear inequalities which involves decision variables α_i , $1 \leq i \leq n$. For cryptanalysis problems, it is necessary to limit some decision variables to integer values, i.e. for some values of i, it is required that $\alpha_i \in Z$. The method which is used to construct as well as solve such problems is called Mixed Integer Linear Programming(MILP). These techniques have built many real-time scenarios in the area of business and economy, but their applications in cryptology have been limited. For differential and linear cryptanalysis, MILP can be used to solve the problem of determining the minimum no. of differentially/linearly active S-boxes which in turn can be used to search for the best differential/linear characteristic for an r-round block cipher. There has been sufficient research in this area but not efficient enough for many symmetric-key ciphers. Differential cryptanalysis and linear cryptanalysis are two major methods for the analysis of symmetric-key ciphers. Linear cryptanalysis (or plaintext attack) studies probabilistic linear relations known as linear approximations between parity bits of the plaintext, the ciphertext and the secret key. Differential cryptanalysis, primarily applicable to block ciphers, precisely verifies how i/p differences in the plaintext correspond to the o/p differences in the ciphertext. Resistance against most general cryptanalysis that is differential and linear cryptanalysis is a conventional benchmark for the design of new ciphers. In this paper, the minimum number of active S-boxes for differential cryptanalysis (and therefore, the security bounds against this attack) is calculated by solving an MILP problem.

1.1 Contribution

In this paper, we explored the use of MILP technique for differential cryptanalysis of block ciphers like KLEIN, LED & AES. Our contributions in this paper are as follows.

- We have generated MILP equations for KLEIN, LED and AES.
- In this work, the generated MILP equations are solved using the Cplex Optimizer (Solver)[12].
- Finally, we have calculated the active number of S-boxes using MILP for the above block ciphers which can be used for proving that whether these ciphers are resistant to differential cryptanalysis or not.

1.2 Organization of the paper

This paper is organized in the following manner: The literature review of differential attack and finding active S-boxes using MILP over past years is given in Section 2. Section 3 briefs the summary of the mathematical preliminaries of MILP. MILP on KLEIN, LED and AES ciphers is given in Section 4, 5 and 6 respectively. Finally, Section 7 concludes the paper.

2 Related work

N. Mouha et al. [9] has given a new procedure to prove security bounds against linear and differential cryptanalysis. They have used mixed-integer linear programming (MILP), a method which is practised in business and economics to solve optimization problems for evaluating the resistance of the cipher against differential and linear cryptanalysis. Generating the simple equations which are i/p to an MILP solver minimizes the task of designers and attackers. As very little programming is needed, both the possibility of human errors and time spent on cryptanalysis are significantly reduced. While designing secure and fast block ciphers, it is mandatory to evaluate immunity against linear and differential attacks. To evaluate the resistance against differential attack, obtaining the lower bound α on the number of active S-boxes is an efficient technique. Based on MILP, authors [19] have given a better technique for analysis of EPCBC. The block and key size of EPCBC is 48-bit and 96-bit respectively. In this paper, authors showed that 32 rounds of the cipher are secure enough to resist differential attacks. Authors in their work [18] have used the extended MILP technique to search linear trails and differential characteristics of block ciphers. Firstly, they show how to model an S-box operation by linear inequalities and propagation of division property of three primary operations (copy, bitwise AND, XOR). Using these, they are able to construct a linear in-equality system. This linear in-equality system precisely explains the division property propagations of a block cipher given an initial division property. Secondly, they have converted a search problem using Todo's framework into an MILP problem by choosing an appropriate objective function. To search integral distinguishers they have used this MILP problem significantly.
To prove the security of Skinny-64/192, authors [20] have used MILP which can categorize the non-linear function and linear function in a round function. By using the MILP program, they have automatically found a 11-round differential characteristic for Skinny-64/192 which is having the minimum number of active s-boxes. The experimental result shows that Skinny-64/192 is resistant to 11-round differential analysis and verifies the efficiency of the MILP method. Zhou Chunning et al. [21] have greatly improved the effectiveness of the MILP-based search algorithm for finding minimum no. of differentially/linearly active S-boxes, and search for the better differential/linear characteristics. By using divide-and-conquer technique, authors have divided the sets which consists of all feasible differential/linear characteristics into various smaller subsets, then indepen-

dently searched each subsets. Minimal solutions in the compact subsets are grouped to produce the minimal solution in the entire set. The authors [22] uses the mixed-integer linear programming (MILP) method for automatic search for differential characteristics of HIGHT cipher as well as LEA cipher. Authors have shown that the differential property of modular addition with one constant input is illustrated with fewer number of linear inequalities input to the MILP solver compared to the common case. Hongluan Zhao et al. [23] improved the search for differential trails of Midori64 given by Banik et al. [16] by using MILP technique. They have given a better 5-round differential characteristics of Midori64 with the MILP-based model. The probabilities of 5-round differential characteristics are 2^{-52} and 2^{-58} respectively.

3 Preliminaries

3.1 Differential cryptanalysis of block ciphers using MILP

Let's Δ denote a string which consists of n bytes $\Delta = (\Delta_0, \Delta_1, ..., \Delta_{n-1})$. The difference vector $\alpha = (\alpha_0, \alpha_1, ..., \alpha_{n-1})$ corresponding to Δ is defined as α, where $\alpha \in F_2^n$.

$$\alpha_i = \begin{cases} 0 & \text{if } \Delta_i = 0 \\ 1 & \text{otherwise} \end{cases}$$

Example 1:
Suppose Δ = { 01001000, 11110010, 00000000, 00011101 } then, α = { 1, 1, 0, 1 }. Non-zero byte gives non-zero bit and zero byte gives zero bit.

3.1.1 Equations for XOR operation:

For XOR operation, let the i/p difference vector and the corresponding o/p difference vector be denoted as $(\alpha_{in_1}^{\oplus}, \alpha_{in_2}^{\oplus})$ and $(\alpha_{out}^{\oplus})$ respectively. The minimum number of input and output bytes that contain non-zero differences is called as differential branch number. The differential branch number is 2 for XOR operation. The represent the branch number in equation form, a new binary dummy variable $d^{\oplus}$ is to be introduced. Here,

$$d^{\oplus} = \begin{cases} 0, & \text{if } (\alpha_{in_1}^{\oplus}, \alpha_{in_2}^{\oplus}, \text{and } \alpha_{out}^{\oplus} = 0) \\ 1, & \text{otherwise} \end{cases}$$

The following linear inequalities (in binary variables) which illustrate the relation between the i/p and o/p

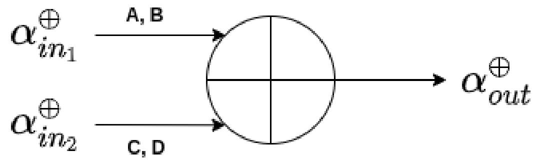

Figure 1: Input and Output for Xor Operation.

difference vectors:

$$\alpha_{in_1}^{\oplus} + \alpha_{in_2}^{\oplus} + \alpha_{out}^{\oplus} \geq 2d^{\oplus},$$
$$d^{\oplus} \geq \alpha_{in_1}^{\oplus},$$
$$d^{\oplus} \geq \alpha_{in_2}^{\oplus},$$
$$d^{\oplus} \geq \alpha_{out}^{\oplus}.$$

Example 2:
In figure 1, let (A, B) be the pair which is the first i/p to the XOR operation and (C, D) be the pair which is the second i/p to the XOR operation denote two sets of n-bit input for the XOR operation. Let A, B, C, D be represented as follows.

$$A = (a_0, a_1, ..., a_{n-1})$$
$$B = (b_0, b_1, ..., b_{n-1})$$
$$C = (c_0, c_1, ..., c_{n-1})$$
$$D = (d_0, d_1, ..., d_{n-1})$$

- The difference vector for the first input, denoted as $\alpha_{in_1}^{\oplus} = A \oplus B$
- The difference vector for the second input, denoted as $\alpha_{in_2}^{\oplus} = C \oplus D$
- The output difference vector, denoted as $\alpha_{out}^{\oplus} = A \oplus C \oplus B \oplus D$.
- Suppose $\alpha_{in_1}^{\oplus} = (a_0 \oplus b_0, a_1 \oplus b_1, ..., a_{n-1} \oplus b_{n-1}) = (1, 0, 0, 0, \ldots, 0)$ and $\alpha_{in_2}^{\oplus} = (c_0 \oplus d_0, c_1 \oplus d_1, \ldots, c_{n-1} \oplus d_{n-1}) = (0, 0, 0, 0, \ldots, 0)$
- $\alpha_{out}^{\oplus} = A \oplus C \oplus B \oplus D = A \oplus B \oplus C \oplus D = \alpha_{in_1}^{\oplus} \oplus \alpha_{in_2}^{\oplus} = (1, 0, 0, 0, \ldots, 0)$

Difference in 1 input byte will result in a difference of 1 output byte. Therefore, the differential branch number is 2 for the XOR operation.

3.1.2 Equations for linear transformation:

The equations for the linear transformation L, can be given in the following manner. Suppose L modifies the input difference vector $(\alpha^L_{in_1}, \alpha^L_{in_2}, ..., \alpha^L_{in_m})$ to the output difference vector $(\alpha^L_{out_1}, \alpha^L_{out_2}, . . . ,\alpha^L_{out_m})$. For the differential branch number β_D , once more a new binary dummy variable d^L is required to define the relation between the i/p and o/p difference vectors. d^L variable is defined as =

$$\begin{cases} 0, & \text{if } \alpha^L_{in_1}, \alpha^L_{in_1}, \cdot\cdot, \alpha^L_{in_m}, \alpha^L_{out_1}, \alpha^L_{out_1}, \cdot\cdot, \alpha^L_{out_m} = 0 \\ 1, & \text{otherwise} \end{cases}$$

Hence, linear transformation L can be described by the given linear inequalities:

$$x^L_{in_1} + x^L_{in_1} + .. + x^L_{in_m} + x^L_{out_1} + x^L_{out_1} + .. + x^L_{out_m} \geq \beta_D d^L$$

$$d^L \geq x^L_{in_1},$$

$$d^L \geq x^L_{in_2},$$

$$.....$$

$$d^L \geq x^L_{in_m}.$$

$$d^L \geq x^L_{out_1},$$

$$d^L \geq x^L_{out_2},$$

$$.....$$

$$d^L \geq x^L_{out_m}.$$

3.1.3 The objective function:

The objective function is to minimize the number of active S-boxes. This function is computationally equivalent to the summation of all independent variables that correspond to the S-box inputs.

3.1.4 Additional constraints

To avoid the trivial solution when the minimum active S-boxes is zero, an additional constraint should be added to ensure that the MILP solver does not output the trivial solution, where the minimum number of active S-boxes is zero. The constraint is that the sum of the input variables to the S-boxes should be atleast 1.

```
$sk^1 \leftarrow$ KEY;
STATE $\leftarrow$ PLAINTEXT;
for $i = 1$ to $N_R$ do
  $AddRoundKey$(STATE, $sk^i$);
  $SubNibbles$(STATE);
  $RotateNibbles$(STATE);
  $MixNibbles$(STATE);
  $sk^{i+1} = KeySchedule(sk^i, i)$;
end for
CIPHERTEXT $\leftarrow AddRoundKey$(STATE, $sk^{N_R+1}$);
```

Figure 2: KLEIN Encryption Routine.

3.2 Relation between the number of active S-boxes and security against differential cryptanalysis

Suppose the highest probability in the Difference Distribution Table(DDT) of an S-box is 2^{-m}. Suppose the minimum number of S-boxes that are active for r rounds of the cipher is n. Then the differential probability of the entire cipher is 2^{-mn}. Therefore the number of plaintext-ciphertext pairs needed to carry out differential cryptanalysis is 2^{mn}. If 2^{mn} exceeds the amount available of today's computational power then the cipher with r rounds is resistant to differential cryptanalysis.

4 MILP on Klein cipher

KLEIN block cipher [13] follows a typical Substitution-Permutation Network (SPN) structure. SPN structures used by other advanced block ciphers are AES and PRESENT [7] etc. To obtain a good amount of security as well as iterations, authors have given the no. of rounds N_R as 12/16/20 for KLEIN-64/80/96 respectively. A detailed block diagram of the KLEIN encryption engine is given in Figure 2.

4.1 The round transformation

The input and output of the KLEIN cipher are considered to be 1-D arrays of bytes. The operations will be minimized with byte-oriented algorithms in the round transformation process. Klein has major four operations, AddRoundKey, SubNibbles, RotateNibble and MixNibbles in round transformation.

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

Table 1: KLEIN S-box (4×4).

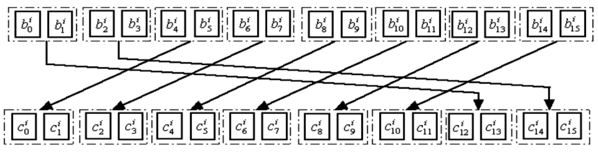
b_0^i b_1^i b_2^i b_3^i b_4^i b_5^i b_6^i b_7^i b_8^i b_9^i b_{10}^i b_{11}^i b_{12}^i b_{13}^i b_{14}^i b_{15}^i
c_0^i c_1^i c_2^i c_3^i c_4^i c_5^i c_6^i c_7^i c_8^i c_9^i c_{10}^i c_{11}^i c_{12}^i c_{13}^i c_{14}^i c_{15}^i

Figure 3: RotateNibbles

Add round key:

The round subkey is bitwise XORed to the input plaintext. The XORed o/p will be split into sixteen blocks of 4-bit nibbles and later each block will be the input to the 4×4 KLEIN S-box.

The SubNibbles step:

The S-box of the KLEIN cipher is both a permutation as well as involution which is given in Table 1. One can save the implementation costs for its inverse by selecting an involutive S-box. The only non-linear layer in KLEIN is the SubNibbles step, a natural requirement is a minimal resistance against differential and linear cryptanalysis.

The RotateNibbles steps:

The sixteen nibbles $b_0^i, b_1^i, \cdots, b_{15}^i$ will be rotated to the left by two bytes in every round. During decryption, these sixteen nibbles will be rotated to the right by the two bytes in every round. For the i^{th} round where $i \in [1, N_R]$, the overall process is given in Figure 3.

MixNibbles step:

In the i^{th} round, input nibbles will be divided into two halves(tuples). Each tuple is considered as an element of the polynomial ring $F_2^8[X]$ and is multiplied with a fixed polynomial $c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02$ modulo $x^4 + 1$. The output of this step s^{i+1} will be the input for the next round. The computation of the state s^{i+1} is given in Figure 4.

$$\begin{bmatrix} s_0^{i+1} \,||\, s_1^{i+1} \\ s_2^{i+1} \,||\, s_3^{i+1} \\ s_4^{i+1} \,||\, s_5^{i+1} \\ s_6^{i+1} \,||\, s_7^{i+1} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} c_0^i \,||\, c_1^i \\ c_2^i \,||\, c_3^i \\ c_4^i \,||\, c_5^i \\ c_6^i \,||\, c_7^i \end{bmatrix}$$

$$\begin{bmatrix} s_8^{i+1} \,||\, s_9^{i+1} \\ s_{10}^{i+1} \,||\, s_{11}^{i+1} \\ s_{12}^{i+1} \,||\, s_{13}^{i+1} \\ s_{14}^{i+1} \,||\, s_{15}^{i+1} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} c_8^i \,||\, c_9^i \\ c_{10}^i \,||\, c_{11}^i \\ c_{12}^i \,||\, c_{13}^i \\ c_{14}^i \,||\, c_{15}^i \end{bmatrix}$$

Figure 4: MixNibbles

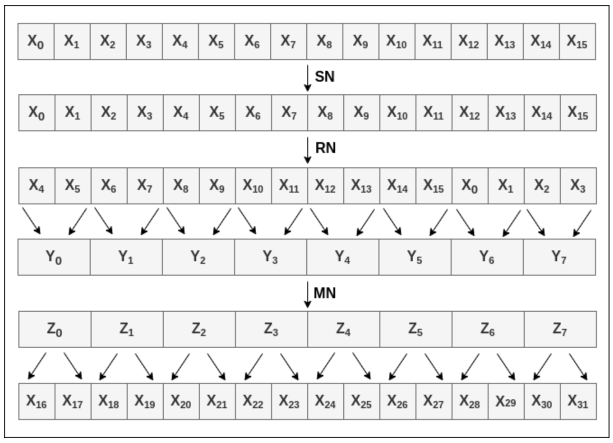
X0 X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15
SN
X0 X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15
RN
X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X0 X1 X2 X3
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
MN
Z0 Z1 Z2 Z3 Z4 Z5 Z6 Z7
X16 X17 X18 X19 X20 X21 X22 X23 X24 X25 X26 X27 X28 X29 X30 X31

Figure 5: The Variable in the First Round Update of KLEIN.

4.2 Result: active S-boxes for N rounds of KLEIN

KLEIN round function consists of four operations: AddRoundKey(AR), SubNibbles(SN), RotateNibble(RN) and MixNibbles(MN). The update of the first KLEIN round is shown in Figure 4. Each of the x variables corresponds to a nibble of the KLEIN state.

$$x_i = \begin{cases} 1, & \text{if } \Delta_i \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

The objective function is the sum of all the variables that are input to the SubNibbles operation, this gives the number of S-boxes that are active. The variable after the SubNibble operation remain unchanged, since the 4×4 S-box used in the KLEIN cipher is an involutive permutation which implies that a non-zero input difference would lead to nonzero output difference and a zero input difference would result in a zero output difference. After the RotateNibble step, the nibbles are grouped into bytes and given as input to the

MixNibbles. These bytes are represented by the variable Y and Z in the Figure 5.

S. No.	No. of Rounds	Time (Secs.)	Time (Ticks)	Ticks/ Seconds	Active S-box
1	01	0.01	1.2700	197.72	01
2	02	0.01	2.6300	345.62	05
3	03	0.01	6.0500	427.13	08
4	04	0.02	15.650	673.74	15
5	05	0.04	29.980	668.72	16
6	06	0.05	43.670	796.91	20
7	07	0.04	27.320	614.33	23
8	08	0.06	33.730	603.47	30
9	09	0.06	36.220	639.09	31
10	10	0.09	61.220	671.66	35
11	11	0.09	61.660	699.18	38
12	12	0.13	82.290	628.46	45
13	13	0.11	77.250	708.89	46
14	14	0.12	88.830	748.94	50
15	15	0.12	91.400	772.36	53
16	16	0.15	108.13	728.12	60

Table 2: Minimum Number of Differentially Active S-boxes for N rounds of KLEIN Cipher.

An active byte(Y variable or Z variable is 1) implies that only one of the left or right nibbles from which it was grouped is active. For example; $Y_0 = X_0||X_1$ is 1 then only one of X_0 or X_1 is 1. The matrix over F_2^8 used in the MixNibble step has a differential branch number of 5. C program was written for generating the constraints and the objective function that has to be minimized, which is then given as input to the CPLEX solver. The number of active S-boxes for various rounds of KLEIN cipher are given in Table 2.

5 MILP on LED cipher

LED is a 64-bit block cipher [10] with two different key sizes, 64-bit as well as 128-bit. The cipher state is conceptually arranged in a (4×4) matrix where each nibble represents polynomial of degree 3 over $GF(2)$, in other words it can be considered as an element of F_2^4 constructed using the irreducible polynomial $X^4 + X + 1$.

S-boxes: LED cipher uses the same S-box as PRESENT which has also been used in many lightweight ciphers. The hexadecimal notation of the PRESENT S-box is given in the following Table 3.

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

Table 3: Present S-box

MixColumnsSerial: The MDS matrix M over F_2^4 used in the MixColumnSerial step of LED is obtained by multiplying a hardware-friendly matrix A over F_2^4 four times,

$$(A)^4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{bmatrix}^4 = \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix} = M$$

5.1 Specification of LED:

The 64-bit plaintext m of LED is divided into 16 nibbles $m_0||m_1||\ ...m_{14}||m_{15}$. These nibbles are arranged in a 4×4 matrix.

$$\begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{04} & m_{05} & m_{06} & m_{07} \\ m_{08} & m_{09} & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{bmatrix}$$

This is the first initialized value of the cipher state and the state is initialized in row manner, this gives us extra hardware-friendly choice [11]. In this, the key will be divided into l nibbles as $k_0, k_1, \cdots, k_l$. Each subkey sk^i is arranged in a 4×4 matrix. Each element of the matrix in the i^{th} sub-key is obtained as sk_j^i = k($j + i \times 16$ mod l) which is given below in a matrix form.

$$\begin{bmatrix} sk_{00}^i & sk_{01}^i & sk_{02}^i & sk_{03}^i \\ sk_{04}^i & sk_{05}^i & sk_{06}^i & sk_{07}^i \\ sk_{08}^i & sk_{09}^i & sk_{10}^i & sk_{11}^i \\ sk_{12}^i & sk_{13}^i & sk_{14}^i & sk_{15}^i \end{bmatrix}$$

The same sub-key is used after every step for a 64-bit key. In the case of a 128-bit key, two sub-keys are used in alternate steps. The sub-key for the case when the key is of size 64-bit is given below in a matrix form.

$$\begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{04} & k_{05} & k_{06} & k_{07} \\ k_{08} & k_{09} & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{bmatrix}$$

The AddRoundKey, bitwise XORs the nibbles of subkey SK^i with the cipher output value. The encryption process is described using addRoundKey(state,SK^i) and a second operation, step(state). The block-wise structure is given in Figure 6. The number of steps s during the encryption process depends on the key size. For 64-bit keys, the s value is 4.

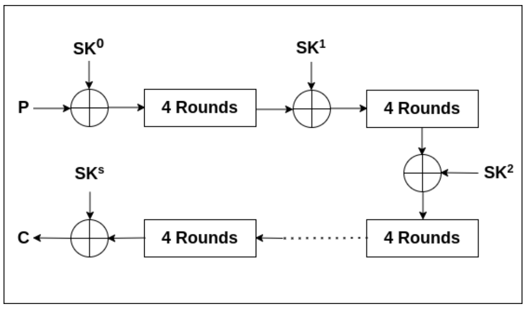

Figure 6: Use of subkeys SK^i in s-steps.

Each step consists of four rounds and each round has AddConstants, SubCells, ShiftRows, and MixColumnsSerial that are executed in sequence. The structure of these operations is given in Figure 7.

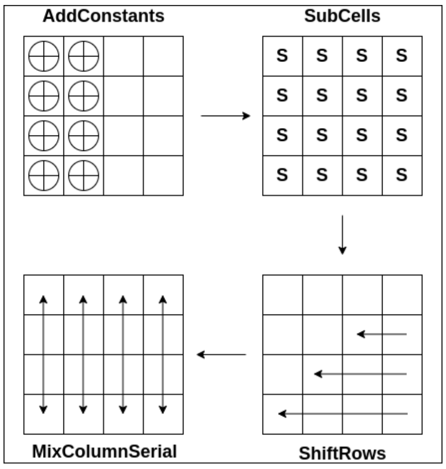

Figure 7: Single Round of LED Architecture.

AddConstants: The following state array is Xored with the output of previous round as shown in Figure 8.

$$\begin{bmatrix} 0 \oplus (ks_7||ks_6||ks_5||ks_4) & (rc_5||rc_4||rc_3) & 0 & 0 \\ 0 \oplus (ks_7||ks_6||ks_5||ks_4) & (rc_5||rc_4||rc_3) & 0 & 0 \\ 0 \oplus (ks_7||ks_6||ks_5||ks_4) & (rc_5||rc_4||rc_3) & 0 & 0 \\ 0 \oplus (ks_7||ks_6||ks_5||ks_4) & (rc_5||rc_4||rc_3) & 0 & 0 \end{bmatrix}$$

Figure 8: Round Constant with State.

$$\begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{04} & x_{05} & x_{06} & x_{07} \\ x_{08} & x_{09} & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} \rightarrow \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{04} & x_{05} & x_{06} & x_{07} \\ x_{08} & x_{09} & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} x_{16} & x_{17} & x_{18} & x_{19} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{24} & x_{25} & x_{26} & x_{27} \\ x_{28} & x_{29} & x_{30} & x_{31} \end{bmatrix} \leftarrow \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{05} & x_{06} & x_{07} & x_{04} \\ x_{10} & x_{11} & x_{08} & x_{09} \\ x_{15} & x_{12} & x_{13} & x_{14} \end{bmatrix} \hookleftarrow$$

Table 4: The Variable in the First Round Update of LED Cipher.

Where the 8-bits $(ks_7, ks_6, \cdots, ks_0)$ represent the key size in bits. The six bits $(rc_5, rc_4, rc_3, rc_2, rc_1, rc_0)$ are the round constants. The round constants from the previous round are shifted to the left once and the new value of rc_0 is calculated as $rc_5 \oplus rc_4 \oplus 1$. These newly updated round constants are used in the current round. The round constants are simply bitwise exclusive-or with the state.

SubCells: The values are substituted by the PRESENT S-box given in Table 3.

ShiftRow: The first row remains unchanged. The second row is circularly shifted to the left by one position. The third and fourth rows are shifted to the left by two and three positions respectively.

5.2 Result: active S-boxes for N rounds of LED

Each round of the LED cipher consists of four operations: AddConstants(AC), SubCells(SC), ShiftRows(SR), and MixColumnsSerial(MC). An AddRoundKey operation is performed after every four rounds(called a step). The update of the first round of LED is shown in Table 4.

Each nibble of the LED state is represented by a variable x. The variable will assume the value 1, for a non-zero difference, and the value 0 for a zero difference. The input and output difference after the AddConstants operation remains the same. The objective function is the sum of all the variables that are input to the subnibble operation, this corresponds to the number of active S-boxes. Since the 4×4 S-box used in LED is a permutation, no new variable needs to be introduced after the SubCell operation. The linear transformation of MixColumnsSerial operation of LED cipher has a differential branch number of 5(since it is a 4×4 invertible matrix). C program was written for generating the constraints and the objective function that has to be minimized which is then given as input to the CPLEX solver. The number of active S-boxes for various rounds of LED cipher are given in Table 5.

S. No.	No. of Rounds	Time (Secs.)	Time (Ticks)	Ticks/ Seconds	Active S-box
1	04	0.035	13.030	445.72	025
2	08	0.055	30.410	612.59	050
3	12	0.095	68.380	760.87	075
4	16	0.145	98.270	690.29	100
5	20	0.205	160.35	787.48	125
6	24	0.255	201.39	817.27	150
7	28	0.335	269.76	818.14	175
8	32	0.525	406.80	780.58	200

Table 5: Minimum Number of Differentially Active S-boxes for N rounds of LED Cipher.

6 MILP on AES cipher

AES [14] is an iterative cipher and it is based on the design principle known as substitution-permutation network. All computations performed in AES are on bytes. So, AES uses the same 16 bytes of block as an input and output. The state matrix is used to store these 16 bytes. Number of rounds in AES is totally dependent on the length of the key. For 10, 12 and 14 rounds, the key size is 128, 192 and 256 bits respectively. In each round of AES, the encryption process uses a different 128 bit round sub-key and these round sub-keys are generated by the key schedule algorithm of AES.

6.1 Specification of AES:

In each round of the AES cipher, four processes are involved which are SubBytes, ShiftRow, MixColumn and AddRoundKey respectively. The description of first round is given in Figure 9.

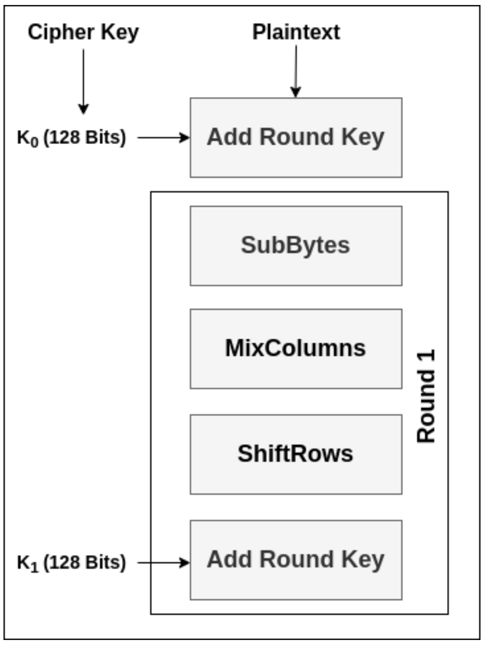

Figure 9: First Round.

Byte substitution (SubBytes):

The input to the sub-bytes is 128-bits (16-bytes), these 16 bytes will be substituted by looking up a fixed size table called S-box. The output of the S-box is also 16 bytes and this output will be stored in a matrix of 4×4.

Shiftrows:

In the ShiftRow operation, each row of the state matrix will be left shifted by a fixed number of bits. The entries which are discarded by shift operation are placed on the right side of the row. Shifting of the bits is given below as.

- No shifting in the 1^{st} row.
- 2^{nd} row is circularly shifted one (byte) position to the left.
- 3^{rd} row is circularly shifted two (byte) positions to the left.
- 4^{th} row is circularly shifted three (byte) positions to the left.

After completion of ShiftRow operation, the new 16 bytes are stored in a matrix.

$$\begin{bmatrix} x_{00} & x_{04} & x_{08} & x_{12} \\ x_{01} & x_{05} & x_{09} & x_{13} \\ x_{02} & x_{06} & x_{10} & x_{14} \\ x_{03} & x_{07} & x_{11} & x_{15} \end{bmatrix} \rightarrow \begin{bmatrix} x_{00} & x_{04} & x_{08} & x_{12} \\ x_{01} & x_{05} & x_{09} & x_{13} \\ x_{02} & x_{06} & x_{10} & x_{14} \\ x_{03} & x_{07} & x_{11} & x_{15} \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} x_{16} & x_{20} & x_{24} & x_{28} \\ x_{17} & x_{21} & x_{25} & x_{29} \\ x_{18} & x_{22} & x_{26} & x_{30} \\ x_{19} & x_{23} & x_{27} & x_{31} \end{bmatrix} \leftarrow \begin{bmatrix} x_{00} & x_{04} & x_{08} & x_{12} \\ x_{05} & x_{09} & x_{13} & x_{01} \\ x_{10} & x_{14} & x_{02} & x_{06} \\ x_{15} & x_{03} & x_{07} & x_{11} \end{bmatrix} \hookleftarrow$$

Table 6: Variables after First Round Execution of AES.

MixColumns:

In MixColumns transformation four bytes of each column is transformed by a special mathematical function. Here, four bytes of one column is completely transformed into four new bytes, which replaces the previous column. The result is stored in another new matrix consisting of 16 new bytes. In the last round, MixColumns transformation step is not performed.

AddRoundKey

This step is the simple XOR between the 16 bytes of the matrix and 16 bytes of round subkey. Output matrix of this function is 16 bytes (128 Bits). For the last round of the cipher, after the AddRoundKey step, the resulting 16 bytes is the ciphertext and for any other rounds, the 16 bytes goes as the input for another round.

6.2 Result: active S-boxes for N rounds of AES

The number of active S-boxes in a linear or differential characteristic of four AES rounds is at least 25 is proved by the four-round propagation theorem of AES [4]. In AES round Function, there are basically four major operations: AddRoundKey(AR), SubBytes(SB), Shift-Rows (SR) and Mix-Columns(MC). The variables after the first round of AES update is shown in Table 6.

Each variable in the AES state is a byte. If the difference is non-zero, the variable is 1 and vice-versa. The objective function is the addition of all the variables that are i/p to the SubByte operation, this corresponds to the number of active S-boxes. C program was written for generating the constraints and the objective function that has to be minimized which is then given as input to the CPLEX solver. The number of active S-boxes for various rounds of AES cipher are given in Table 7.

S. No.	**No. of Round**	**Time (Secs.)**	**Time (Ticks)**	**Ticks/ Seconds**	**Active S-box**
1	01	0.001	0.300	62.471	01
2	02	0.025	03.10	179.55	05
3	03	0.030	07.15	263.95	09
4	04	0.030	13.03	501.27	25
5	05	0.030	17.26	564.57	26
6	06	0.045	19.99	466.95	30
7	07	0.046	29.30	614.66	34
8	08	0.055	30.46	644.10	50
9	09	0.057	42.97	700.31	51
10	10	0.085	49.81	737.10	55
11	11	0.089	66.71	782.41	59
12	12	0.095	68.38	782.48	75
13	13	0.098	78.52	883.74	76
14	14	0.125	97.39	929.00	80

Table 7: Minimum Number of Differentially Active S-boxes for N rounds of AES Cipher.

7 Conclusion

In this paper, we have examined the MILP procedure to analyse the robustness of ciphers against differential cryptanalysis. The most important requirement is the input cipher should be a combination of s-box operations, linear permutation layers and/or Exclusive OR operations. The objective function of the MILP is to calculate the minimum number of differentially active S-boxes. Some of the off-the-shelf optimization packages like CPLEX can be used to solve MILP problems. The MILP technique is applied to KLEIN, LED and AES Ciphers. Since a very little programming is required to achieve above results, a programmer with limited experience can modify the cipher implementation and get the required MILP program. In the above scenario of KLEIN, LED and AES, CPLEX has taken comparatively very less time for proving the security against differential cryptanalysis.

7.1 Future work

As the extension of this work, the symmetries in the round function can be used to further speed-up the search. In addition to this, in scenarios where large number of rounds are present, the split approach can be used to get an approximate rough lower bound. In future, our research intention is also to tweak the internal parameters of CPLEX solver for evaluating the bounds against differential cryptanalysis in comparatively less amount of time.

References

[1] Biham E., Shamir A. (1991) Differential Cryptanalysis of DES-like Cryptosystems. Advances in Cryptology-CRYPTO' 90. CRYPTO 1990. Lecture Notes in Computer Science, vol 537. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-38424-3_1

[2] Matsui M. Linear Cryptanalysis Method for DES Cipher. Advances in Cryptology - EUROCRYPT '93. EUROCRYPT 1993. Lecture Notes in Computer Science, vol 765. Springer, Berlin, Heidelberg, 1994. https://doi.org/10.1007/3-540-48285-7_33

[3] Daemen J., Clapp C. Fast Hashing and Stream Encryption with Panama. Fast Software Encryption. FSE 1998. Lecture Notes in Computer Science, vol 1372. Springer, Berlin, Heidelberg, 1998. https://doi.org/10.1007/3-540-69710-1_5

[4] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, 2002. https://doi.org/10.1007/978-3-662-60769-5_3

[5] Das, M.L., Saxena, A., Gulati, V.P. An efficient proxy signature scheme with revocation, Informatica, Vol. 15 Issue 4, pp.455-464, 2004. https://doi.org/10.15388/Informatica.2004.072

[6] C. S. Ma and R. H. Miller, MILP optimal path planning for real-time applications, 2006 American Control Conference, Minneapolis, MN, , pp. 6 pp.-, 2006. https://doi.org/10.1109/ACC.2006.1657504

[7] Bogdanov A. et al. PRESENT: An Ultra-Lightweight Block Cipher. Cryptographic Hardware and Embedded Systems - CHES 2007. CHES 2007. Lecture Notes in Computer Science, vol 4727. Springer, Berlin, Heidelberg 2007. https://doi.org/10.1007/978-3-540-74735-2_31

[8] Borghoff J., Knudsen L.R., Stolpe M. Bivium as a Mixed-Integer Linear Programming Problem. Cryptography and Coding. IMACC 2009. Lecture Notes in Computer Science, vol 5921. Springer, Berlin, Heidelberg 2009. https://doi.org/10.1007/978-3-642-10868-6_9

[9] Mouha N., Wang Q., Gu D., Preneel B. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. Information Security and Cryptology. Inscrypt 2011. Lecture Notes in Computer Science, vol 7537. Springer, Berlin, Heidelberg, 2011. https://doi.org/10.1007/978-3-642-34704-7_5

[10] Guo J., Peyrin T., Poschmann A., Robshaw M. The LED Block Cipher. Cryptographic Hardware and Embedded Systems – CHES 2011. CHES 2011. Lecture Notes in Computer Science, vol 6917. Springer, Berlin, Heidelberg, 2011. https://doi.org/10.1007/978-3-642-23951-9_22

[11] Moradi A., Poschmann A., Ling S., Paar C., Wang H. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. Advances in Cryptology – EUROCRYPT 2011. EUROCRYPT 2011. Lecture Notes in Computer Science, vol 6632. Springer, Berlin, Heidelberg, 2011. https://doi.org/10.1007/978-3-642-20465-4_6

[12] https://www.ibm.com/in-en/analytics/cplex-optimizer

[13] Gong Z., Nikova S., Law Y.W. KLEIN: A New Family of Lightweight Block Ciphers. Security and Privacy. RFIDSec 2011. Lecture Notes in Computer Science, vol 7055. Springer, Berlin, Heidelberg, 2012. https://doi.org/10.1007/978-3-642-25286-0_1

[14] Fathy A., Tarrad I.F., Hamed H.F.A., Awad A.I. Advanced Encryption Standard Algorithm: Issues and Implementation Aspects. Advanced Machine Learning Technologies and Applications. AMLTA 2012. Communications in Computer and Information Science, vol 322.

Springer, Berlin, Heidelberg, 2012.
https://doi.org/10.1007/978-3-642-35326-0_51

[15] Sun S., Hu L., Wang P., Qiao K., Ma X., Song L. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. Advances in Cryptology - ASIACRYPT 2014. ASIACRYPT 2014. Lecture Notes in Computer Science, vol 8873. Springer, Berlin, Heidelberg, 2014.
https://doi.org/10.1007/978-3-662-45611-8_9

[16] Banik S. et al. Midori: A Block Cipher for Low Energy. Advances in Cryptology – ASIACRYPT 2015. ASIACRYPT 2015. Lecture Notes in Computer Science, vol 9453. Springer, Berlin, Heidelberg, 2015.
https://doi.org/10.1007/978-3-662-48800-3_17

[17] Bhattacharya, Rajeev, Linear Programming. Palgrave Encyclopedia of Strategic Management, ISBN 978-1-137-49190-9, Palgrave Macmillan UK, 2014.
https://ssrn.com/abstract=2981081

[18] Xiang Z., Zhang W., Bao Z., Lin D. Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. ASIACRYPT 2016. ASIACRYPT 2016. Lecture Notes in Computer Science, vol 10031. Springer, Berlin, Heidelberg, 2016.
https://doi.org/10.1007/978-3-662-53887-6_24

[19] Ping Yang, Chuankun Wu, Wentao Zhang, Automatic Security Analysis of EPCBC against Differential Attacks, Procedia Computer Science, Volume 107, 2017, Pages 176-182, ISSN 1877-0509, 2017.
https://doi.org/10.1016/j.procs.2017.03.075

[20] Pei Zhang, Wenying Zhang. Differential Cryptanalysis on Block Cipher Skinny with MILP Program. Hindawi Security and Communication Networks Volume 2018.
https://doi.org/10.1155/2018/3780407

[21] Zhou, C., Zhang, W., Ding, T., & Xiang, Z. Improving the MILP-based Security Evaluation Algorithm against Differential/Linear Cryptanalysis Using A Divide-and-Conquer Approach. IACR Transactions on Symmetric Cryptology, 438–469, 2020.
https://doi.org/10.13154/tosc.v2019.i4.438-469

[22] E. Bagherzadeh and Z. Ahmadian, MILP-based automatic differential search for LEA and HIGHT block ciphers, in IET Information Security, vol. 14, no. 5, pp. 595-603, 2020.
https://doi.org/10.1049/iet-ifs.2018.5539

[23] H. Zhao, G. Han, L. Wang and W. Wang, MILP-Based Differential Cryptanalysis on Round-Reduced Midori64, in IEEE Access, vol. 8, pp. 95888-95896, 2020.