Impossible Differential Attack on SAND-128

Nobuyuki Sugio[0000−0001−7313−1755]

Hokkaido University of Science, Sapporo, Japan
sugio-n@hus.ac.jp

Abstract. Impossible differential attack is one of the major cryptanalytical methods for symmetric-key block ciphers. In this paper, we evaluate the security of SAND-128 against impossible differential attack. SAND is an AND-RX-based lightweight block cipher proposed by Chen et al. in Designs, Codes and Cryptography 2022. There are two variants of SAND, namely SAND-64 and SAND-128, due to structural differences. In this paper, we search for impossible differential distinguishers of SAND-128 using the Constraint Programming (CP) and reveal 14-round impossible differential distinguishers. The number of 14-round distinguishers is $2^{14} \times 7 = 114,688$. Furthermore, we demonstrate a key recovery attack on 21-round SAND-128. The complexities for the attack require 2^{124} data, $2^{127.2}$ encryptions, and 2^{122} bytes of memory, respectively. Although this result currently achieves the best attack on round-reduced SAND-128, this attack does not threaten the security of SAND-128 against impossible differential attack.

Keywords: Impossible differential attack, lightweight cipher, SAND, Constraint Programming

1 Introduction

Impossible differential attack was independently proposed by Biham et al. [1] and Knudsen [2]. Impossible differential was defined as the differential with probability zero. This attack eliminates the key candidates which generate differential with probability zero. In this technique, an attacker searches for an impossible output differential Δ_Y corresponding to a given input differential Δ_X over r-round of the cipher. If such a pair of input-output differentials (Δ_X, Δ_Y) exists, it is termed as an r-round impossible differential distinguisher. This distinguisher can be used to conduct a distinguishing attack or a key recovery attack against the target cipher.

Boura et al. have formalized the necessary number of data, time, and memory complexities for the impossible differential attack [3], [4]. In their method, an attacker estimates the sets of all possible input difference Δ_{in} (resp. those of output difference Δ_{out}) of the target cipher using truncated differentials with probability 1. Because the necessary number of data is depend on the sets of all possible input difference Δ_{in} (resp. those of output difference Δ_{out}), an attacker needs to minimize them in order to attack the target cipher efficiently.

Table 1. Attack Results on SAND-128

Number of rounds	Type	Data	Time	Memory	Method
14	Distinguisher	-	-	-	Impossible Differential [5]
14	Distinguisher	-	-	-	Impossible Differential [15]
14	Distinguisher	-	-	-	Impossible Differential (**Ours**)
16	Distinguisher	2^{127}	-	-	Integral [5]
17	Distinguisher	2^{127}	-	-	Integral [17]
20	Key Recovery	2^{127}	2^{119}	2^{76}	Integral [17]
21	Key Recovery	2^{124}	$2^{127.2}$	2^{122}	Impossible Differential (**Ours**)

SAND is proposed by Chen et al. in Designs, Codes and Cryptography 2022 [5]. SAND is a family of lightweight Feistel block ciphers (with 64- or 128-bit block size, both using a 128-bit key) designed using only AND, rotation, and XOR (AND-RX) operations, while enabling classical S-box-based cryptanalytic techniques.

For security evaluation, the designers analyzed differential attack [6], linear attack [7], and other cryptanalytical methods using Mixed Integer Linear Programming (MILP). In the previous research on SAND-128, the designers applied the method proposed in [8] to search for impossible differential distinguishers. They fixed the number of active S-boxes in the input and output differences as one, then performed an exhaustive search at nibble level. However, this method limits the search space, potentially overlooking other impossible differential distinguishers.

Hadipour et al. proposed methods to search for impossible differential distinguishers using deterministic differentials [9], [10]. These approaches do not require predefined input and output differences, allowing for a more efficient search for impossible differential distinguishers.

1.1 Contributions of This Paper

In this paper, we evaluate the security of SAND-128 against impossible differential attack. We firstly search for impossible differential distinguishers of SAND-128 using the bit-based Hadipour et al.'s method [10] and reveal 14-round impossible differential distinguishers. The number of 14-round distinguishers is $2^{14} \times 7 = 114,688$. Furthermore, we demonstrate a key recovery attack on 21-round SAND-128. The attack results are summarized in Table 1.

1.2 Structure of This Paper

The structure of this paper is as follows. Section 2 provides related works on SAND-128. Section 3 explains an overview of impossible differential attack. Section 4 introduces constraint programming and the use of deterministic differentials in searching for impossible differential distinguishers. Sections 5 describes the structures of SAND-128. In Section 6, we illustrate 14-round impossible

differential distinguishers and present a key recovery attack on 21-round SAND-128. Section 7 provides a deeper discussion of the implications of our findings, highlights the novelty of our approach, and outlines its limitations in the broader context of cryptanalysis on SAND-128. Finally, Section 8 concludes the paper.

2 Related Works

The designers evaluated SAND's security against various attacks such as differential attack [6], linear attack [7], integral attack [12], impossible differential attack [1], and zero-correlation linear attack [13], [14]. They searched for impossible differentials and zero-correlation linear approximations using a nibble-level search using the method proposed in [8]. As a result, they identified 14-round distinguishers both impossible differential and zero-correlation for SAND-128 [5].

Zhang et al. developed a systematic search framework for AND-RX ciphers to find impossible differential distinguishers [15]. Applying their method to SAND, they found 456 types of 14-round impossible differential distinguisher of SAND-128.

Mirzaie et al. applied the conventional bit-based division property technique [16] to find integral distinguishers on reduced-round SAND-128 [17]. They discovered a 17-round integral distinguisher with nine balanced bits. The data complexity for the distinguisher is 2^{127}. Building on this distinguisher, Mirzaie et al. mounted a key recovery attack on 20-round SAND-128. The time and memory complexities are 2^{119} encryptions and 2^{76} bytes, respectively.

3 Impossible Differential Attack

Impossible differential attack was proposed by Biham et al. [1]. Impossible differential was defined as the differential with probability zero. This attack eliminates the key candidates which generate differential with probability zero. In this technique, an attacker searches for an impossible output differential Δ_Y corresponding to a given input differential Δ_X over r-round of the cipher. If such a pair of input-output differentials (Δ_X, Δ_Y) exists, it is termed as an r-round impossible differential distinguisher. This distinguisher can be used to conduct a distinguishing attack or a key recovery attack against the target cipher.

Boura et al. have formalized the necessary number of data, time, and memory complexities for the impossible differential attack [3], [4]. The following outlines them. For more details, please refer to the references [3], [4].

Figure 1 illustrates the notations for an impossible differential attack. Let Δ_X and Δ_Y be input (resp. output) differences of the impossible differential. Let r_Δ be the number of rounds of the impossible differential. Let Δ_{in} and Δ_{out} be set of all possible input (resp. output) differences of the cipher. Let r_{in} and r_{out} be the number of rounds of the differential paths (Δ_X, Δ_{in}) or (Δ_Y, Δ_{out}).

The differential $(\Delta_X \rightarrow \Delta_{in})$ (resp. $(\Delta_Y \rightarrow \Delta_{out})$) occurs with probability 1 while the differential $(\Delta_{in} \rightarrow \Delta_X)$ (resp. $(\Delta_{out} \rightarrow \Delta_Y)$) is verified with probability

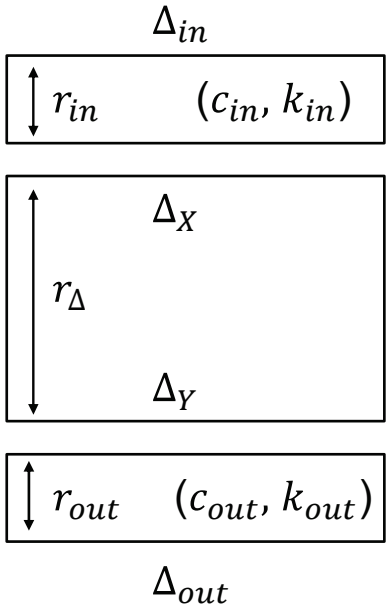

Fig. 1. Notations for an impossible differential attack [3]

$\frac{1}{2^{c_{in}}}$ (resp. $\frac{1}{2^{c_{out}}}$), where c_{in} (resp. c_{out}) is the number of bit-conditions that have to be verified to obtain Δ_X from Δ_{in} (resp. Δ_Y from Δ_{out}).

The probability that for a given key, a pair of inputs already satisfying the differences Δ_{in} and Δ_{out} verifies all the $(c_{in} + c_{out})$ bit-conditions is $2^{-(c_{in}+c_{out})}$. Therefore, the probability that a key trial is kept in the candidate keys set is $p_k = (1 - 2^{-(c_{in}+c_{out})})^N$ with N different input (or output) pairs.

Boura et al. have formalized the smallest value of N, denoted by N_{min}, verifying

$$p_k = (1 - 2^{-(c_{in}+c_{out})})^{N_{min}} < \frac{1}{2}$$

is approximately $N_{min} = 2^{c_{in}+c_{out}}$. The cost of obtaining N pairs of $(\Delta_{in}, \Delta_{out})$ is evaluated as

$$C_N = \max\left\{\min_{\Delta\in\{\Delta_{in},\Delta_{out}\}}\left\{\sqrt{N2^{n+1-|\Delta|}}\right\}, N2^{n+1-|\Delta_{in}|-|\Delta_{out}|}\right\}. \tag{1}$$

The cost C_N also represents the amount of needed data. The time complexity is

$$T = \left(C_N + \left(N + 2^{|k_{in}\cup k_{out}|}\frac{N}{2^{c_{in}+c_{out}}}\right)C'_E + 2^{|K|}p_k\right)C_E, \tag{2}$$

with N such that $p_k = (1 - 2^{-(c_{in}+c_{out})})^N < \frac{1}{2}$, and where C'_E is the ratio of the cost of partial encryption to the full encryption, and where the last term represents the brute-force search complexity.

The only elements that need to be stored are the N pairs. Therefore, the memory complexity for the attack is determined by N.

4 Constraint Programming and Its Application to Cryptanalysis

4.1 Constraint Programming

Constraint Programming (CP) is a programming paradigm that aims to solve problems by employing mathematical and computational techniques to meet specific conditions, known as constraints. Constraints $\mathcal{C}$ are conditions that the values of variables must satisfy. They are expressed in mathematical or logical form. Variables $\mathcal{X}$ are elements that can take on specific values. Each variable has a domain $\mathcal{D}$, which is the range of possible values it can assume. A constraint problem consists of a set of variables and the constraints imposed on them. The goal is to find a combination of variable values that satisfies all the constraints.

4.2 Application to Cryptanalysis

Hadipour et al. proposed the cell-wise model [9] and the bit-wise model [10] to search for impossible differential distinguishers and zero-correlation linear trails using CP. In the bit-wise model, specific constraints were set to track the encryption and decryption processes for each round at the bit level and an attacker could find bit-wise impossible differential distinguishers and zero-correlation linear trails of a target cipher. Specifically, Hadipour et al. applied it to ASCON and this method discovered 5-round impossible differential distinguishers and zero-correlation linear trails [10].

The following outlines the CP models for deterministic differential transitions. For more details, please refer to the references [9], [10]. In the bit-wise models, let integer variables X and Y with the domain of $\{-1, 0, 1\}$ to indicate whether the differential is unknown, zero, or one, respectively.

CP model 1 (Branching) [10]

For $f : \mathbb{F}_2 \to \mathbb{F}_2^n$, $f(x) = (y_0, y_1, \cdots, y_{n-1})$, where $y_0 = y_1 = \cdots = y_{n-1} = x$, the valid transitions for deterministic differential trails satisfy the following:

$$\texttt{Branch}(X, Y[0], \cdots, Y[n-1]) := \bigwedge_{i=0}^{n-1} (Y[i] = X),$$

where X and $Y[i]$ are integer variables with the domain of $\{-1, 0, 1\}$ for all $0 \leq i \leq n-1$.

CP model 2 (XOR) [10]

For $f : \mathbb{F}_2^n \to \mathbb{F}_2$, $f(x_0, x_1, \cdots, x_{n-1}) = y$, where $y = x_0 \oplus x_1 \oplus \cdots \oplus x_{n-1}$, the valid transitions for deterministic differential trails satisfy the following:

$$\texttt{XOR}(X[0], \cdots, X[n-1], Y) := \begin{cases} \text{if } \bigvee_{i=0}^{n-1}(X[i] = -1) \text{ then } Y = -1 \\ \text{else } Y = \sum_{i=0}^{n-1} X[i] \bmod 2 \text{ endif} \end{cases}$$

where $X[i]$ and Y are integer variables with the domain of $\{-1, 0, 1\}$ for all $0 \le i \le n-1$.

CP model 3 (S-box) [10]

CP model for S-box can be derived from differential distribution table (DDT).

The way how to encode deterministic behaviours of S-box using sbox analyzer is explained in Appendix N of [10].

CP models to search deterministic differential distinguishers are constructed as follows. We define integer variables XU_r and XL_r, $0 \le r \le R$ to represent the active pattern of the internal state after r rounds of a block cipher in the forward and backward directions. CP models for deterministic differential trails in forward and backward directions over R rounds are independently constructed. The constraints $\sum_{i=0}^{n-1} XU_0[i] \neq 0$ and $\sum_{i=0}^{n-1} XL_R[i] \neq 0$ are added to exclude all trivial solutions. The following constraints are added to ensure the inconsistency between the two deterministic differential propagations at least one point throughout the distinguisher:

$$CSP := \bigvee_{r=0}^{R-1} \left(\bigvee_{i=0}^{n-1} (XU_r[i] + XL_r[i] = 1) \right)$$

If CP models are feasible using a CP-solver, it means that there exist r-round impossible differentials. Otherwise, one can not find any impossible differential at r-round using deterministic differential trails.

5 Lightweight Block Cipher SAND-128

SAND is a lightweight block cipher proposed by Chen et al. [5]. It employs an AND-RX structure and has two variants, SAND-64 and SAND-128, depending on the block size. In this paper, we focus on SAND-128, which has a block size of 128 bits and a secret key length of 128 bits. The recommended number of SAND-128 is 54. In the following description, we set $n = 64$.

5.1 Preliminaries

The notations used in the description of SAND-128 are defined as follows:

- $x = (x_{n-1}, x_{n-2}, \ldots, x_0)$: An n-bit variable, where x_{n-1} represents the most significant bit (MSB) and x_0 represents the least significant bit (LSB). The variable x is represented using a $4 \times \frac{n}{4}$ array:

$$x = \begin{bmatrix} x_{n-1} \ldots x_7 \; x_3 \\ x_{n-2} \ldots x_6 \; x_2 \\ x_{n-3} \ldots x_5 \; x_1 \\ x_{n-4} \ldots x_4 \; x_0 \end{bmatrix}$$

- $x||y$: Concatenation of variables x and y
- $x \ll s$: Left shift of variable x by s bits
- $x \lll t$: Left cyclic shift of variable x by t bits
- $x \lll_{\frac{n}{4}} t$: Variable x is divided into four $\frac{n}{4}$-bit words $x = (x_{n-1}, x_{n-2}, \ldots, x_0) = x\{3\}||x\{2\}||x\{1\}||x\{0\}$, and each word $x\{i\}$ undergoes a left cyclic shift by t bits:

$$x \lll_{\frac{n}{4}} t = (x\{3\} \lll_{\frac{n}{4}} t)||(x\{2\} \lll_{\frac{n}{4}} t)||(x\{1\} \lll_{\frac{n}{4}} t)||(x\{0\} \lll_{\frac{n}{4}} t).$$

- $x \odot y$: Bitwise AND operation
- $x \oplus y$: Bitwise XOR operation
- $x[i]$: The i-th nibble (4-bit) of variable x. Given $x = (x_{n-1}, x_{n-2}, \ldots, x_0)$:

$$\begin{aligned} x[\tfrac{n}{4} - 1] &= (x_{n-1}, x_{n-2}, x_{n-3}, x_{n-4}), \\ &\vdots \\ x[1] &= (x_7, x_6, x_5, x_4), \\ x[0] &= (x_3, x_2, x_1, x_0). \end{aligned}$$

5.2 State Loading

Let $P = (Pl, Pr)$ be the plaintext, where $Pl = (Pl_{n-1}, \ldots, Pl_1, Pl_0)$ represents the left n-bit part and $Pr = (Pr_{n-1}, \ldots, Pr_1, Pr_0)$ represents the right n-bit part. The variables Pl and Pr are represented as $4 \times \frac{n}{4}$ arrays:

$$Pl = \begin{bmatrix} Pl_{n-1} \ldots Pl_7 \; Pl_3 \\ Pl_{n-2} \ldots Pl_6 \; Pl_2 \\ Pl_{n-3} \ldots Pl_5 \; Pl_1 \\ Pl_{n-4} \ldots Pl_4 \; Pl_0 \end{bmatrix} = \begin{bmatrix} x^0\{3\} \\ x^0\{2\} \\ x^0\{1\} \\ x^0\{0\} \end{bmatrix}, \; Pr = \begin{bmatrix} Pr_{n-1} \ldots Pr_7 \; Pr_3 \\ Pr_{n-2} \ldots Pr_6 \; Pr_2 \\ Pr_{n-3} \ldots Pr_5 \; Pr_1 \\ Pr_{n-4} \ldots Pr_4 \; Pr_0 \end{bmatrix} = \begin{bmatrix} y^0\{3\} \\ y^0\{2\} \\ y^0\{1\} \\ y^0\{0\} \end{bmatrix}$$

The input (x^0, y^0) for the first round is assigned row by row from plaintext P:

$$x^0 = Pl_{n-1} \ldots Pl_7 \; Pl_3|| \ldots ||Pl_{n-4} \ldots Pl_4 \; Pl_0 = x^0\{3\}||x^0\{2\}||x^0\{1\}||x^0\{0\}$$
$$y^0 = Pr_{n-1} \ldots Pr_7 \; Pr_3|| \ldots ||Pr_{n-4} \ldots Pr_4 \; Pr_0 = y^0\{3\}||y^0\{2\}||y^0\{1\}||y^0\{0\}$$

Let $C = (Cl, Cr)$ be the ciphertext, where $Cl = (Cl_{n-1}, \ldots, Cl_1, Cl_0)$ represents the left n-bit part and $Cr = (Cr_{n-1}, \ldots, Cr_1, Cr_0)$ represents the right n-bit part. The ciphertext C is assigned row by row from the round R output (x^R, y^R):

$$Cl = \begin{bmatrix} Cl_{n-1} \ldots Cl_7 \; Cl_3 \\ Cl_{n-2} \ldots Cl_6 \; Cl_2 \\ Cl_{n-3} \ldots Cl_5 \; Cl_1 \\ Cl_{n-4} \ldots Cl_4 \; Cl_0 \end{bmatrix} = \begin{bmatrix} x^R\{3\} \\ x^R\{2\} \\ x^R\{1\} \\ x^R\{0\} \end{bmatrix}, \; Cr = \begin{bmatrix} Cr_{n-1} \ldots Cr_7 \; Cr_3 \\ Cr_{n-2} \ldots Cr_6 \; Cr_2 \\ Cr_{n-3} \ldots Cr_5 \; Cr_1 \\ Cr_{n-4} \ldots Cr_4 \; Cr_0 \end{bmatrix} = \begin{bmatrix} y^R\{3\} \\ y^R\{2\} \\ y^R\{1\} \\ y^R\{0\} \end{bmatrix}$$

5.3 Round Function

The round function of SAND-128 is illustrated in Figure 2. Let (x^r, y^r) denote the input of round r, sk^r denote the round key, and (x^{r+1}, y^{r+1}) denote the output. The round function consists of two types of nonlinear functions, G_0 and G_1, as well as a linear function P_n. In addition, the parameters α and β are fixed as $\alpha = 0$ and $\beta = 1$.

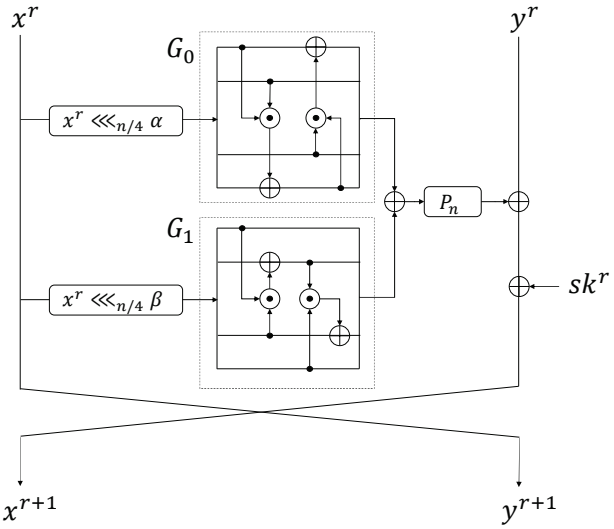

Fig. 2. Round function of SAND-128

Let the input to G_0 and G_1 be an n-bit variable $x = x\{3\}||x\{2\}||x\{1\}||x\{0\}$ and the output be $y = y\{3\}||y\{2\}||y\{1\}||y\{0\}$. The function G_0 is defined as follows:

$$\begin{aligned} y\{3\} &= y\{0\} \odot x\{1\} \oplus x\{3\}, \\ y\{2\} &= x\{2\}, \\ y\{1\} &= x\{1\}, \\ y\{0\} &= x\{3\} \odot x\{2\} \oplus x\{0\}. \end{aligned}$$

Similarly, the function G_1 is defined as follows:

$$\begin{aligned} y\{3\} &= x\{3\}, \\ y\{2\} &= x\{3\} \odot x\{1\} \oplus x\{2\}, \\ y\{1\} &= y\{2\} \odot x\{0\} \oplus x\{1\}, \\ y\{0\} &= x\{0\}. \end{aligned}$$

The linear function P_n transforms the input word $x\{i\} = (x_{\frac{n}{4}\cdot i+\frac{n}{4}-1}, \ldots, x_{\frac{n}{4}\cdot i+1}, x_{\frac{n}{4}\cdot i})$ into the output word $y\{i\}$ according to the following equation:

$$y_{\frac{n}{4}\cdot i+p_{\frac{n}{4}}(j)} = x_{\frac{n}{4}\cdot i+j}, \quad \text{for } 0 \le j < \frac{n}{4},\ 0 \le i < 4.$$

The linear function P_n can be regarded as applying the permutation p_{16} to each of the four $\frac{n}{4}$-bit words in parallel. The permutation p_{16} is defined in Table 2.

Table 2. Permutation p_{16} in SAND-128

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$p_{16}(j)$	14	15	8	9	2	3	12	13	6	7	0	1	10	11	4	5

5.4 Key Schedule

SAND-128 generates round keys sk^r $(0 \le r < R)$ from a 128-bit secret key K. The secret key is treated as a concatenation of two 64-bit words: $K = K^1||K^0$. Figures 3 and 4 illustrate the key schedule of SAND-128.

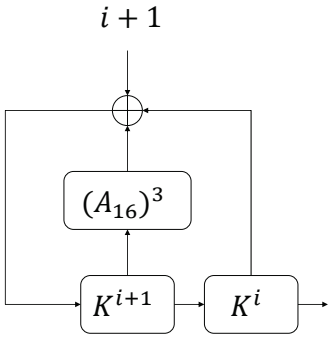

Fig. 3. Key schedule of SAND-128

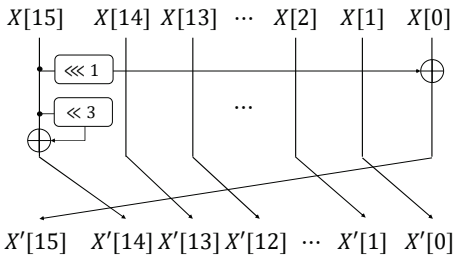

Fig. 4. Operation A_{16}

The constant $i+1$ $(0 \le i < R-2)$ is a round-dependent constant. The update of the linear feedback shift register (LFSR) is defined as:

$$K^{i+2} \leftarrow (A_{16})^3(K^{i+1}) \oplus K^i \oplus (i+1).$$

The operation A_{16} processes data in 4-bit units and is applied to K^{i+1} for three times in succession. The round key sk^r $(0 \le r < R)$ is loaded from K^r as follows:

$$K^r = \begin{bmatrix} K^r_{63} \ldots K^r_7 & K^r_3 \\ K^r_{62} \ldots K^r_6 & K^r_2 \\ K^r_{61} \ldots K^r_5 & K^r_1 \\ K^r_{60} \ldots K^r_4 & K^r_0 \end{bmatrix},$$

$$sk^r = K^r_{63} \dots K^r_3 || K^r_{62} \dots K^r_2 || K^r_{61} \dots K^r_1 || K^r_{60} \dots K^r_0.$$

The operation A_{16} is also nibble-oriented, with the 64-bit input X split into 16 nibbles $X[15]|| \cdots ||X[1]||X[0]$. Then, the output is calculated as follows.

$$(X[15] \lll 1) \oplus X[0] || X[15] \oplus (X[15] \ll 3) || X[14] || X[13] || \cdots || X[2] || X[1].$$

6 Impossible Differential Attack on SAND-128

6.1 Construction of the CP Model

We construct a constraint programming (CP) model to search for impossible differential distinguishers of SAND-128 using the method explained in Section 4. The domain of the following integer variables is defined as $\{-1, 0, 1\}$, where each value represents an unknown, 0, and 1 differences, respectively.

Based on the round function illustrated in Figure 2, we define 64-bit integer variables XU_r and YU_r $(0 \le r \le R)$, which represent the internal state differences in the forward (encryption) direction. Similarly, we define 64-bit integer variables XL_r and YL_r $(0 \le r \le R)$, which represent the internal state differences in the backward (decryption) direction. The variables XU_0 and YU_0 (as well as XL_0 and YL_0) represent the input differences, while XU_R and YU_R (as well as XL_R and YL_R) represent the output differences after R rounds.

The construction of the CP model in the forward (encryption) direction using XU_r and YU_r is described below. We define 64-bit integer variables Δ_{Pl} and Δ_{Pr}, which represent the plaintext differences, and impose the following constraints:

$$\begin{aligned} XU_0 &= \texttt{StateLoading}(\Delta_{Pl}), \\ YU_0 &= \texttt{StateLoading}(\Delta_{Pr}). \end{aligned}$$

We define 64-bit integer variables G_0U_r and G_1U_r $(1 \le r \le R)$ to represent the output differences of the nonlinear functions G_0 and G_1, and impose the following constraints:

$$\begin{aligned} G_0U_r &= G_0(XU_r), \\ G_1U_r &= G_1(XU_r \lll_{\frac{n}{4}} 1). \end{aligned}$$

The constraints for G_0 and G_1 can be derived from the differential distribution table (DDT). In this paper, we refer to Appendix N of [10] and use the S-box analyzer[1] to derive the constraints for G_0 and G_1.

We define 64-bit integer variables PU_r $(1 \le r \le R)$ to represent the output differences of the linear function P_n, and impose the following constraint:

$$PU_r = P_n(\texttt{XOR}(G_0U_r, G_1U_r)).$$

For the round function output, the following constraints are imposed:

$$\begin{aligned} XU_{r+1} &= \texttt{XOR}(YU_r, PU_r), \\ YU_{r+1} &= XU_r. \end{aligned}$$

[1] `https://github.com/hadipourh/sboxanalyzer`

By constructing these constraints for each round r ($0 \leq r \leq R$), the CP model for the forward (encryption) direction is completed. Similarly, the CP model for the backward (decryption) direction can be constructed using integer variables XL_r and YL_r.

Additionally, we introduce the following constraints to eliminate trivial solutions:

$$\sum_{i=0}^{n-1} XU_0[i] + \sum_{i=0}^{n-1} YU_0[i] \neq 0,$$
$$\sum_{i=0}^{n-1} XL_R[i] + \sum_{i=0}^{n-1} YL_R[i] \neq 0.$$

Finally, to ensure that there is at least one contradiction between the deterministic difference propagation paths in the forward (encryption) and backward (decryption) directions, we introduce the following constraints:

$$\left\{ \bigvee_{r=0}^{R-1} \left(\bigvee_{i=0}^{n-1} (XU_r[i] + XL_r[i] = 1) \right) \right\} \vee \left\{ \bigvee_{r=0}^{R-1} \left(\bigvee_{i=0}^{n-1} (YU_r[i] + YL_r[i] = 1) \right) \right\}.$$

If the constructed CP model is satisfiable in the CP solver, it indicates the existence of an r-round impossible differential distinguisher.

6.2 Impossible Differential distinguishers of SAND-128

We implemented the CP model constructed in the previous section using MiniZinc[2], and employed OR-Tools[3] as a CP solver. The computing environment used in this study is summarized in Table 3.

Table 3. Computing Environment

Environment	Details
OS	Windows 11
Platform	MiniZinc 2.9.2
Solver	OR-Tools CP-SAT 9.12.4544
CPU	AMD Ryzen 9 5950X
Memory	128 GB

As a result of searching for impossible differential distinguishers of SAND-128, we obtained a solution in approximately five minutes, revealing 14-round impossible differential distinguishers of SAND-128. The number of 14-round distinguishers is $2^{14} \times 7 = 114,688$. The results are presented in Tables 4 and 5. In both tables, the nibble-wise symbols **0** and **?** denote **0** = 0000 and **?** =????, respectively.

In addition, when attempting to search for 15-round impossible differential distinguishers, the result was `UNSATISFIABLE`, and no valid solution could be obtained.

[2] `https://www.minizinc.org/`

[3] `https://developers.google.com/optimization`

Table 4. 14-round Impossible Differentials of SAND-128 (Left 64-bit)

Pl	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	?0??	**0**	**0**	**0**	**0**	**0**
XU_0	**0**	**0**	00?0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	00?0	**0**	**0**	**0**	00?0	**0**
XU_1	000?	?000	**0**	?000	000?	**0**	**0**	0100	000?	?000	**0**	?000	000?	?000	**0**	??00
XU_2	**0**	**?**	00?0	0??0	**0**	??11	**0**	0?00	**0**	???0	00?0	0??0	**0**	**?**	00?0	0??0
XU_3	?00?	**?**	??00	?0??	000?	0???	?100	01??	?00?	**?**	?000	?0??	?00?	**?**	??00	**?**
XU_4	**?**	**?**	???0	0???	???1	??0?	??00	0???	???0	**?**	???0	0???	**?**	**?**	???0	0???
XU_5	**?**	**?**	**?**	?0??	**?**	**?**	**?**	01??	**?**	**?**	**?**	?0??	**?**	**?**	**?**	**?**
XU_6	**?**	**?**	**?**	**?**	**?**	?? 1 ?	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
XU_7	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
XU_{14}	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
XL_0	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
XL_5	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
XL_6	**?**	**?**	**?**	**?**	**?**	?? 0 ?	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
XL_7	**?**	**?**	**?**	?0??	**?**	0???	**?**	00??	**?**	**?**	**?**	?0??	**?**	**?**	**?**	?0??
XL_8	???0	**?**	???0	0???	??00	??0?	??00	0???	???0	**?**	???0	0???	???0	**?**	???0	0???
XL_9	000?	0???	?000	?0??	000?	0???	**0**	00??	000?	0???	?000	?0??	000?	0???	?000	?0??
XL_{10}	**0**	???0	00?0	0?00	**0**	??00	**0**	0?00	**0**	???0	00?0	0?00	**0**	???0	00?0	0?00
XL_{11}	000?	**0**	**0**	?000	000?	**0**	**0**	**0**	000?	**0**	**0**	?000	000?	**0**	**0**	?000
XL_{12}	**0**	**0**	00?0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	00?0	**0**	**0**	**0**	00?0	**0**
XL_{13}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
XL_{14}	**0**	**0**	00?0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	00?0	**0**	**0**	**0**	00?0	**0**
Cl	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	?0??	**0**	**0**	**0**	**0**	**0**

Table 5. 14-round Impossible Differentials of SAND-128 (Right 64-bit)

Pr	**0**	**0**	**0**	**?**	?0??	**0**	**0**	**0**	**0**	**0**	**0**	**0**	?0??	010?	**0**	**0**
YU_0	000?	?000	**0**	?000	000?	**0**	**0**	0100	000?	?000	**0**	?000	000?	?000	**0**	??00
YU_1	**0**	**0**	00?0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	00?0	**0**	**0**	**0**	00?0	**0**
YU_2	000?	?000	**0**	?000	000?	**0**	**0**	0100	000?	?000	**0**	?000	000?	?000	**0**	??00
YU_3	**0**	**?**	00?0	0??0	**0**	??11	**0**	0?00	**0**	???0	00?0	0??0	**0**	**?**	00?0	0??0
YU_4	?00?	**?**	??00	?0??	000?	0???	?100	01??	?00?	**?**	?000	?0??	?00?	**?**	??00	**?**
YU_5	**?**	**?**	???0	0???	???1	??0?	??00	0???	???0	**?**	???0	0???	**?**	**?**	???0	0???
YU_6	**?**	**?**	**?**	?0??	**?**	**?**	**?**	01??	**?**	**?**	**?**	?0??	**?**	**?**	**?**	**?**
YU_7	**?**	**?**	**?**	**?**	**?**	?? 1 ?	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
YU_8	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
YU_{14}	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
YL_0	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
YL_6	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
YL_7	**?**	**?**	**?**	**?**	**?**	?? 0 ?	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**
YL_8	**?**	**?**	**?**	?0??	**?**	0???	**?**	00??	**?**	**?**	**?**	?0??	**?**	**?**	**?**	?0??
YL_9	???0	**?**	???0	0???	??00	??0?	??00	0???	???0	**?**	???0	0???	???0	**?**	???0	0???
YL_{10}	000?	0???	?000	?0??	000?	0???	**0**	00??	000?	0???	?000	?0??	000?	0???	?000	?0??
YL_{11}	**0**	???0	00?0	0?00	**0**	??00	**0**	0?00	**0**	???0	00?0	0?00	**0**	???0	00?0	0?00
YL_{12}	000?	**0**	**0**	?000	000?	**0**	**0**	**0**	000?	**0**	**0**	?000	000?	**0**	**0**	?000
YL_{13}	**0**	**0**	00?0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	00?0	**0**	**0**	**0**	00?0	**0**
YL_{14}	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Cr	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

6.3 Key Recovery Attack on 21-Round SAND-128

Using the 14-round impossible differential distinguishers shown in Tables 4 and 5, we perform a key recovery attack on 21-round SAND-128, as illustrated in Figure 5. For simplicity, the `StateLoading` process is omitted. Additionally, the round function is abbreviated as F. In Figure 5, the locations where differences exist are highlighted in red. The data, time, and memory complexities for the key recovery attack are estimated using the method proposed in [3], [4].

At $r_{in} = 4$, the bit conditions required to obtain the output difference at the fourth round $(\Delta x^4, \Delta y^4)$ from the plaintext difference $(\Delta x^0, \Delta y^0)$ are $c_{in} = 28 + 20 + 12 + 8 = 68$. Additionally, the number of key bits involved is $k_{in} = 28$ bits ($sk^0 = 16$ bits, $sk^1 = 8$ bits, $sk^2 = 4$ bits).

Similarly, at $r_{out} = 3$, the bit conditions required to obtain the output difference at the 18th round $(\Delta x^{18}, \Delta y^{18})$ from the ciphertext difference $(\Delta x^{21}, \Delta y^{21})$ are $c_{out} = 24 + 16 + 8 = 48$. Additionally, the number of key bits involved is $k_{out} = 12$ bits ($sk^{19} = 4$ bits, $sk^{20} = 8$ bits).

To reduce the number of key candidates by at least half, the necessary number of differential pairs $(\Delta_{in}, \Delta_{out})$ is $N_{min} = 2^{68+48} = 2^{116}$. The computational complexity to obtain these differential pairs is estimated from Equation (1) as follows:

$$C_N = \max\left\{\left\{\sqrt{2^{116}2^{128+1-51}}\right\}, 2^{116}2^{128+1-70-51}\right\} = 2^{124}$$

This computational complexity C_N also represents the necessary number of plaintexts. The total time complexity required for the key recovery attack is estimated from Equation (2) as follows:

$$T = \left(2^{124} + \left(2^{116} + 2^{28+12}\right) \times \frac{7}{21} + 2^{128} \times \frac{1}{2}\right) \approx 2^{127.2}$$

which corresponds to the number of 21-round SAND-128 encryptions.

Additionally, the memory required to store N_{min} differential pairs is estimated as:

$$M = 2^{116} \times 128 \times 4 \times \frac{1}{8} = 2^{122}$$

bytes. The attack result is summarized in Table 1.

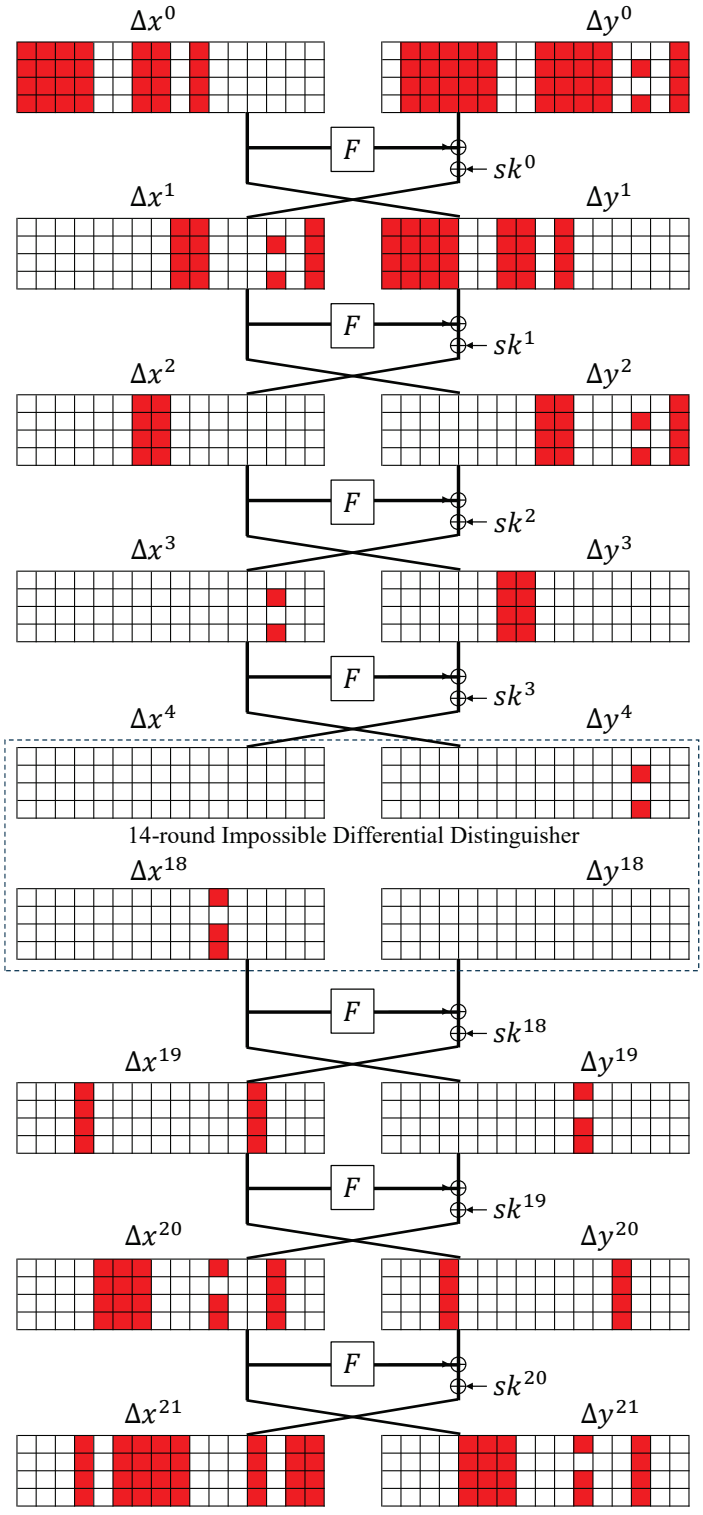

Fig. 5. Key Recovery Attack on 21-Round SAND-128

7 Discussions

This section provides a deeper discussion of the implications of our findings, highlights the novelty of our approach, and outlines its limitations in the broader context of cryptanalysis on SAND-128.

7.1 Advantages of the CP-Based Search Method

The use of Constraint Programming (CP) with the bit-wise deterministic differential model allowed us to efficiently discover a large number of impossible differential distinguishers of SAND-128. Compared to previous works that relied on fixed input/output differences at the nibble level [5] or the systematic search framework [15], the CP-based approach can search for impossible differential distinguishers in wider solution space. This flexibility led to the discovery of $2^{14} \times 7 = 114{,}688$ 14-round impossible differential distinguishers, exceeding the diversity of existing results.

7.2 Interpretation of the UNSAT Result for 15 Rounds

When attempting to find impossible differentials over 15 rounds, the CP solver returned an `UNSATISFIABLE` result. While this does not formally prove their nonexistence, it suggests that such distinguishers are unlikely under the current bit-based modeling assumptions. This observation is consistent with earlier results [5], and supports the designers' claim that SAND-128 maintains resistance against this class of attack up to 14 rounds.

7.3 Theoretical vs. Practical Impact

Although our key recovery attack on 21-round SAND-128 is purely theoretical, it demonstrates the non-random behavior of the cipher under reduced rounds. This adds to the cryptanalytic understanding of SAND and may serve as a foundation for future attacks with reduced complexity, either through optimization or hybrid techniques.

7.4 Comparison with Other Cryptanalytic Techniques

Our approach complements other cryptanalytic results such as integral attacks [17]. While integral attacks have shown slightly lower time complexities, they remain limited to fewer rounds. Our 21-round analysis demonstrates the depth of propagation that impossible differentials can achieve, despite larger complexities demand. Therefore, the CP-based framework provides a valuable alternative and can be combined with other techniques for future explorations.

7.5 Limitations and Future Potential

A limitation of the current CP model lies in its computational complexity and symbolic abstraction, which may restrict its applicability for higher rounds or more complex structures. Additionally, the model assumes deterministic behavior, which could miss probabilistic characteristics exploitable in other forms of analysis. Future work could consider extending this approach to search related-key differentials, combining CP with SMT/SAT solving, or leveraging algebraic representations tailored to the AND-RX structure.

8 Conclusion

In this paper, we conducted an impossible differential attack on the lightweight block cipher SAND-128. We demonstrated that a key recovery attack on 21-round SAND-128 can be applicable using impossible differential distinguishers. The complexities for the attack require 2^{124} data, $2^{127.2}$ encryptions, and 2^{122} bytes of memory, respectively. Although this result currently achieves the best attack on round-reduced SAND-128, this attack does not threaten the security of SAND-128 against impossible differential attack.

References

1. Biham, E., Biryukov, A., and Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials", Advances in Cryptology-EUROCRYPT'99, vol. 1592 of LNCS, pp. 12–23 (1999).
2. Knudsen L.: DEAL-a 128-bit block cipher, Complexity, 258(2), 1998.
3. Boura, C., Naya-Plasencia, M., and Suder, V.: Scrutinizing and Improving Impossible Differential Attacks: Applications to CLEFIA, Camellia, LBlock and Simon, Proc. 20th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2014, Vol. 8873 of LNCS, pp.179-199, Springer-Verlag (2014).
4. Boura, C., Lallemand, V., Naya-Plasencia, M., and Suder, V.: Making the Impossible Possible, Cryptology, Vol. 31, pp.101–133, Springer-Verlag (2018).
5. Chen, S., Fan, Y., Sun, L., Fu, Y., Zhou, H., Li, Y., Wang, M., Wang W., and Guo, C.: SAND: an AND-RX Feistel lightweight block cipher supporting S-box-based security evaluations, Designs, Codes and Cryptography, Vol. 90, pp. 155–198 (2022).
6. Biham, E., and Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard", Springer-Verlag, New York, pp. 79-88 (1993).
7. Matsui, M.: Linear Cryptanalysis Method for DES Cipher, Proc. Workshop on the Theory and Application of Cryptographic Techniques, EUROCRYPT ' 93, Vol. 765 of LNCS, pp.386–397, Springer-Verlag (1993).
8. Cui T., Jia K., Fu K., Chen S., Wang M.: New automatic search tool for impossible differentials and zero-correlation linear approximations, IACR Cryptology ePrint Archive, Report 2016/689 (2016).
9. Hadipour, H., Sadeghi, S. and Eichlseder, M.: Finding the impossible: Automated search for full impossible differential, zero-correlation, and integral attacks, Proc. EUROCRYPT 2023, Vol. 14007 of LNCS, pp. 128–157, Springer-Verlag (2023).

10. Hadipour, H., Gerhalter, S., Sadeghi, S. and Eichlseder, M.: Improved Search for Integral, Impossible Differential and Zero-Correlation Attacks, Application to Ascon, ForkSKINNY, SKINNY, MANTIS, PRESENT and QARMAv2, IACR Transactions on Symmetric Cryptology, Vol. 2024, No. 1, pp. 234–325 (2024).
11. Sun, L., Gerault, D., Wang, W., Wang, M. (2020). On the Usage of Deterministic (Related-Key) Truncated Differentials and Multidimensional Linear Approximations for SPN Ciphers, IACR Transactions on Symmetric Cryptology, Vol. 2020(3), pp.262-287, (2020).
12. Knudsen, L. R., and Wagner, D.: Integral cryptanalysis, Proc. of Fast Software Encryption, FSE2002, Vol. 2365 of LNCS, pp.112-127. Springer-Verlag, (2002).
13. Bogdanov, A., Wang, M.: Zero Correlation Linear Cryptanalysis with Reduced Data Complexity, Proc. of the 19th International Workshop on Fast Software Encryption, FSE 2012, Vol. 7549 of LNCS, pp. 29–48, (2012).
14. Bogdanov, A., and Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers, Designs, Codes and Cryptography, Vol. 70, pp. 369-383, (2014).
15. Zhang, K., Wang, S., Lai, X., Wang, L., Guan, J., Hu, B.: Impossible Differential Cryptanalysis and a Security Evaluation Framework for AND-RX Ciphers, IEEE Transactions on Information Theory, Vol. 70, no. 8, pp. 6025–6040, (2024).
16. Xiang, Z., Zhang, W., Bao, Z., and Lin, D.: Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers, Proc. 22nd International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT2016, Vol.10031 of LNCS, pp.648-678, Springer-Verlag (2016).
17. Mirzaie, A., Ahmadi, S., Aref, M. R.: Division Property-Based Integral Attack on Reduced-Round SAND-128, ISC International Journal of Information Security (ISeCure), Vol. 16, no. 3, (2024).