# Impossible Differential Attack on SAND-64

Nobuyuki Sugio[0000−0001−7313−1755]

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

**Abstract.** SAND is an AND-RX-based lightweight block cipher proposed by Chen et al. 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-64 using the Constraint Programming (CP) and reveal 56 types of impossible differential distinguishers up to 11 rounds. Furthermore, we demonstrate a key recovery attack on 17-round SAND-64. The complexities for the attack require $2^{56}$ data, $2^{127}$ encryptions, and $2^{60}$ bytes of memory, respectively. Although this result currently achieves the best attack on round-reduced SAND-64, this attack does not threaten the security of SAND-64 against impossible differential attack.

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

## 1 Introduction

SAND is proposed by Chen et al. in Designs, Codes and Cryptography 2022 [1]. 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 [2], linear attack [3], and other cryptanalytic methods using Mixed Integer Linear Programming (MILP). In the previous research on SAND-64, the designers applied the method proposed in [4] 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 a method to search for impossible differential distinguishers using deterministic differentials [5], [6]. 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 study, we search for impossible differential distinguishers of SAND-64 using Hadipour et al.'s deterministic differentials and reveal 56 types of impos-

sible differential distinguishers up to 11 rounds. Furthermore, we demonstrate a key recovery attack on 17-round SAND-64. The attack results are summarized in Table 1.

**Table 1.** Attack Results

| Number of rounds | Type | Data | Time | Memory | Method |
|---|---|---|---|---|---|
| 10 | Distinguisher | - | - | - | Impossible Differential Attack [1] |
| 11 | Distinguisher | - | - | - | Impossible Differential Attack [11] |
| 11 | Distinguisher | - | - | - | Impossible Differential Attack (**Ours**) |
| 12 | Distinguisher | $2^{63}$ | - | - | Integral Attack [1] |
| 12 | Distinguisher | $2^{63}$ | - | - | Integral Attack [13] |
| 15 | Key Recovery | $2^{63}$ | $2^{105}$ | $2^{52}$ | Integral Attack [13] |
| 16 | Key Recovery | $2^{63}$ | $2^{109.91}$ | $2^{85}$ | Integral Attack [13] |
| 17 | Key Recovery | $2^{56}$ | $2^{127}$ | $2^{60}$ | Impossible Differential Attack (**Ours**) |

### 1.2 Structure of This Paper

The structure of this paper is as follows. Section 2 provides related works on SAND-64. 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 lightweight cipher SAND-64. In Section 6, we present a key recovery attack on SAND-64 using impossible differential distinguishers. Finally, Section 7 concludes the paper and discusses future directions.

## 2 Related Works

The designers evaluated SAND's security against various attacks such as differential attack [2], linear attack [3], integral attack [7], impossible differential attack [8], and zero-correlation linear attack [9], [10]. They searched for impossible differentials and zero-correlation linear approximations using a nibble-level search. As a result, they identified 10-round distinguishers both impossible differential and zero-correlation of SAND-64 [1].

Zhang et al. developed a systematic search framework for AND-RX ciphers to find impossible differential distinguishers [11]. Applying their method to SAND, they found 11-round impossible differential distinguishers of SAND-64.

Mirzaie et al. applied the conventional bit-based division property technique [12] to find integral distinguishers of SAND-64 [13]. As a result, they discovered a 12-round integral distinguisher with 23 balanced bits. The data complexity for the distinguisher is $2^{63}$. Building on this distinguisher, Mirzaie et al. mounted key recovery attacks on 15- and 16-round SAND-64. The time complexities are $2^{105}$ encryptions for the 15-round attack and $2^{109.9}$ encryptions for the 16-round attack, respectively.

## 3 Impossible Differential Attack

Impossible differential attack was proposed by Biham et al. [8]. 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 $\Delta_Y$ corresponding to a given input differential $\Delta_X$ over $r$-round of the cipher. If such a pair of input-output differentials ($\Delta_X$, $\Delta_Y$) 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 [14], [15]. The following outlines them. For more details, please refer to the references [14], [15].

Figure 1 illustrates the notations for an impossible differential attack. Let $\Delta_X$ and $\Delta_Y$ be input (resp. output) differences of the impossible differential. Let $r_\Delta$ be the number of rounds of the impossible differential. Let $\Delta_{in}$ and $\Delta_{out}$ 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 ($\Delta_X$, $\Delta_{in}$) or ($\Delta_Y$, $\Delta_{out}$).

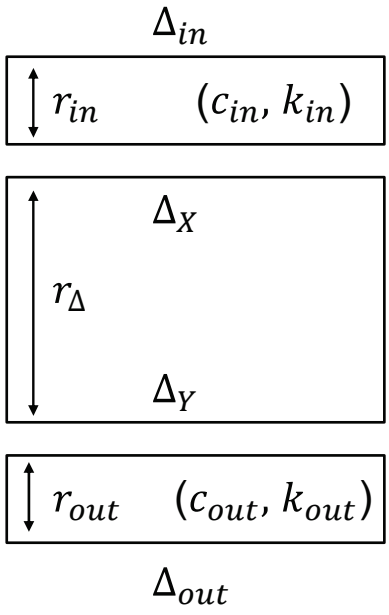



**Fig. 1.** Notations for an impossible differential attack [14]

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

$\frac{1}{2^{c_{in}}}$ (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 $\Delta_X$ from $\Delta_{in}$ (resp. $\Delta_Y$ from $\Delta_{out}$).

The probability that for a given key, a pair of inputs already satisfying the differences $\Delta_{in}$ and $\Delta_{out}$ 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

$$C_N = \max\left\{ \min_{\Delta \in \{\Delta_{in}, \Delta_{out}\}} \left\{ \sqrt{N 2^{n+1-|\Delta|}} \right\}, N 2^{n+1-|\Delta_{in}|-|\Delta_{out}|} \right\}. \tag{1}$$

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 cell-wise model [5] and bit-wise model [6] 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 [6].

The following outlines the CP models for deterministic differential transitions. For more details, please refer to the references [5], [6]. 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) [6]**

*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) [6]**

*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 $0 \leq i \leq n-1$.*

**CP model 3 (S-box) [6]**

*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 [6].

CP models to search deterministic differential distinguishers are constructed as follows. We define integer variables $XU_r$ and $XL_r$, $0 \leq r \leq R$ 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-64

SAND is a lightweight block cipher proposed by Chen et al. [1]. 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-64, which has a block size of 64 bits and a secret key length of 128 bits. The recommended number of SAND-64 is 48. In the following description, we set $n = 32$.

### 5.1 Preliminaries

The notations used in the description of SAND-64 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-64 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 $\alpha$ and $\beta$ are fixed as $\alpha = 0$ and $\beta = 1$.

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:

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

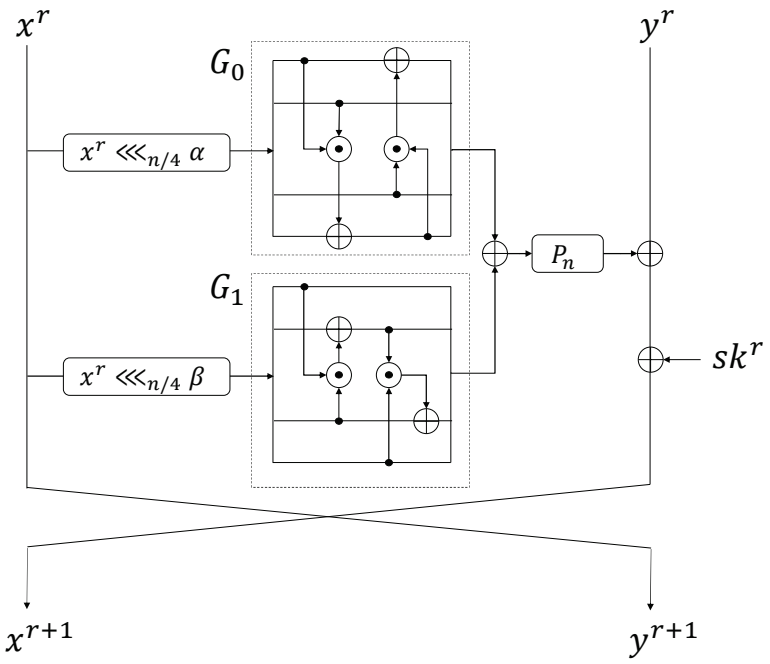



**Fig. 2.** Round function of SAND-64

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:

$$y_{\frac{n}{4}\cdot i+p_8(j)} = x_{\frac{n}{4}\cdot i+j}, \quad \text{for } 0 \le j < \frac{n}{4},\ 0 \le i < 4.$$

The linear function $P_n$ can be regarded as applying the permutation $p_8$ to each of the four $\frac{n}{4}$-bit words in parallel. The permutation $p_8$ is defined in Table 2.

**Table 2.** Permutation $p_8$ in SAND-64

| $j$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| $p_8(j)$ | 7 | 4 | 1 | 6 | 3 | 0 | 5 | 2 |

### 5.4 Key Schedule

SAND-64 generates round keys from a 128-bit secret key $K$. The secret key is treated as a concatenation of four 32-bit words: $K = K^3||K^2||K^1||K^0$. Figures 3 and 4 illustrate the key schedule of SAND-64.

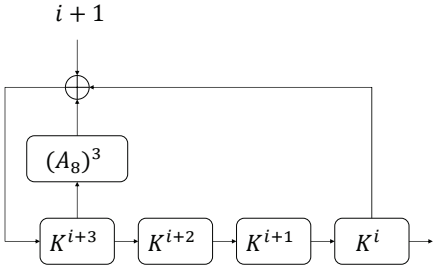



**Fig. 3.** Key schedule of SAND-64

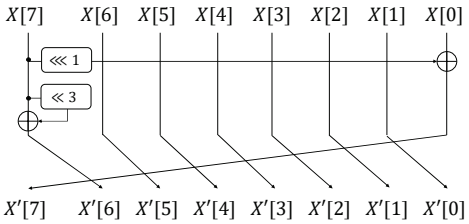



**Fig. 4.** Operation $A_8$

The constant $i+1,\ 0 \le i < R-4$ is a round-dependent constant. The update of the linear feedback shift register (LFSR) is defined as:

$$K^{i+4} \leftarrow (A_8)^3(K^{i+3}) \oplus K^i \oplus (i+1).$$

The operation $A_8$ processes data in 4-bit units and is applied to $K^{i+3}$ three times in succession. The round key $sk^r$ $(0 \le r < R)$ is derived from $K^r$ as follows:

$$K^r = \begin{bmatrix} K^r_{31} \ldots K^r_7 & K^r_3 \\ K^r_{30} \ldots K^r_6 & K^r_2 \\ K^r_{29} \ldots K^r_5 & K^r_1 \\ K^r_{28} \ldots K^r_4 & K^r_0 \end{bmatrix},$$

$$sk^r = K^r_{31} \ldots K^r_3 || K^r_{30} \ldots K^r_2 || K^r_{29} \ldots K^r_1 || K^r_{28} \ldots K^r_0.$$

## 6 Impossible Differential Attack on SAND-64

### 6.1 Construction of the CP Model

We construct a constraint programming (CP) model to search for impossible differential distinguishers of SAND-64 using deterministic differentials. 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 32-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 32-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 32-bit integer variables $\Delta_{Pl}$ and $\Delta_{Pr}$,

which represent the plaintext differences, and impose the following constraints:

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

We define 32-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:

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

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 [6] and use the S-box analyzer[1] to derive the constraints for $G_0$ and $G_1$.

We define 32-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:

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

By constructing these constraints for each round $r$ $(0 \le r \le R)$, 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:

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

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 characteristic.

### 6.2 Impossible Differential distinguishers of SAND-64

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

[1] https://github.com/hadipourh/sboxanalyzer
[2] https://www.minizinc.org/
[3] https://developers.google.com/optimization

**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 |

We obtained a solution in approximately 1 minute and 30 seconds, revealing 56 types of 11-round impossible differential distinguishers. The results are presented in Table 4.

**Table 4.** 11-round impossible differential distinguishers of SAND-64

| | | | |
|---|---|---|---|
| $\Delta_{Pl}$ | 0000 0000 0000 0000 0000 0000 0000 0000 | $\Delta_{Pr}$ | 0000 0000 0000 0000 ???0 0000 0000 0000 |
| $XU_0$ | 0000 0000 0000 0000 0000 0000 0000 0000 | $YU_0$ | 0000 ?000 0000 ?000 0000 ?000 0000 0000 |
| $XU_1$ | 0000 ?000 0000 ?000 0000 ?000 0000 0000 | $YU_1$ | 0000 0000 0000 0000 0000 0000 0000 0000 |
| $XU_2$ | 0?00 ?000 0?00 ?000 0?00 ?000 0?00 0000 | $YU_2$ | 0000 ?000 0000 ?000 0000 ?000 0000 0000 |
| $XU_3$ | 0??0 ??00 0??0 ??00 0??0 ??00 0??0 0?00 | $YU_3$ | 0?00 ?000 0?00 ?000 0?00 ?000 0?00 0000 |
| $XU_4$ | 0??0 ???? 0??0 ???? 0??0 ???? 0??0 0??? | $YU_4$ | 0??0 ??00 0??0 ??00 0??0 ??00 0??0 0?00 |
| $XU_5$ | ???? ???? ???? ???? ???? ???? ???? 0??? | $YU_5$ | 0??0 ???? 0??0 ???? 0??0 ???? 0??0 0??? |
| $XL_5$ | ???? ???? ???? ???? ???? ???? ???? 1??? | $YL_5$ | ???? ???? ???? ???? ???? ???? ???? ???? |
| $XL_6$ | 0??0 ???? 0??0 ???? 0??0 ???? 0??0 1??? | $YL_6$ | ???? ???? ???? ???? ???? ???? ???? 1??? |
| $XL_7$ | 0??0 ??00 0??0 ??00 0??0 ??00 0??0 0?00 | $YL_7$ | 0??0 ???? 0??0 ???? 0??0 ???? 0??0 1??? |
| $XL_8$ | 0?00 ?000 0?00 ?000 0?00 ?000 0?00 1000 | $YL_8$ | 0??0 ??00 0??0 ??00 0??0 ??00 0??0 0?00 |
| $XL_9$ | 0000 ?000 0000 ?000 0000 ?000 0000 1000 | $YL_9$ | 0?00 ?000 0?00 ?000 0?00 ?000 0?00 1000 |
| $XL_{10}$ | 0000 0000 0000 0000 0000 0000 0000 0000 | $YL_{10}$ | 0000 ?000 0000 ?000 0000 ?000 0000 1000 |
| $XL_{11}$ | 0000 ?000 0000 ?000 0000 ?000 0000 1000 | $YL_{11}$ | 0000 0000 0000 0000 0000 0000 0000 0000 |
| $\Delta_{Cl}$ | 0000 0000 0000 0000 ???1 0000 0000 0000 | $\Delta_{Cr}$ | 0000 0000 0000 0000 0000 0000 0000 0000 |

### 6.3 Key Recovery Attack on 17-Round SAND-64

Using the 11-round impossible differential distinguishers shown in Table 4, we perform a key recovery attack on 17-round SAND-64, 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 of Boura et al. [14], [15].

At $r_{in} = 3$, the bit conditions required to obtain the output difference at the 3rd round $(\Delta x^3, \Delta y^3)$ from the plaintext difference $(\Delta x^0, \Delta y^0)$ are $c_{in} = 16+11+8 = 35$. Additionally, the number of key bits involved is $k_{in} = 12$ bits ($sk^0 = 8$ bits, $sk^1 = 4$ bits).

Similarly, at $r_{out} = 3$, the bit conditions required to obtain the output difference at the 14th round $(\Delta x^{14}, \Delta y^{14})$ from the ciphertext difference $(\Delta x^{17}, \Delta y^{17})$ are $c_{out} = 16 + 12 + 8 = 36$. Additionally, the number of key bits involved is $k_{out} = 12$ bits ($sk^{15} = 4$ bits, $sk^{16} = 8$ bits).

To reduce the number of key candidates by at least half, the necessary number of differential pairs $(\Delta_{in}, \Delta_{out})$ is $N_{min} = 2^{71}$. The computational complexity to obtain these pairs is estimated using Equation (1) as follows:

$$C_N = \max\left\{\sqrt{2^{71}2^{64+1-40}}, 2^{71}2^{64+1-40-40}\right\} = 2^{56}$$

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

$$T = \left(2^{56} + \left(2^{71} + 2^{12+12}\right) \times \frac{6}{17} + 2^{128} \times \frac{1}{2}\right) = 2^{127}$$

which corresponds to the number of 17-round SAND-64 encryptions.

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

$$M = 2^{71} \times 64 \times 4 \times \frac{1}{8} = 2^{76}$$

bytes. Although this memory complexity is larger than that of full-codebook, we prepair the memory for the plaintexts and ciphertexts corresponding to the plaintexts for $C_N$ instead of $N_{min}$. In this case, $M = 2^{56}\times 64\times 2\times \frac{1}{8} = 2^{60}$ bytes. The time complexity $T$ for the key recovery is not changed by generating a differential pair $(\Delta_{in}, \Delta_{out})$ from $C_N$ and then by sieving a key candidate sequentially. The attack result is summarized in Table 1.

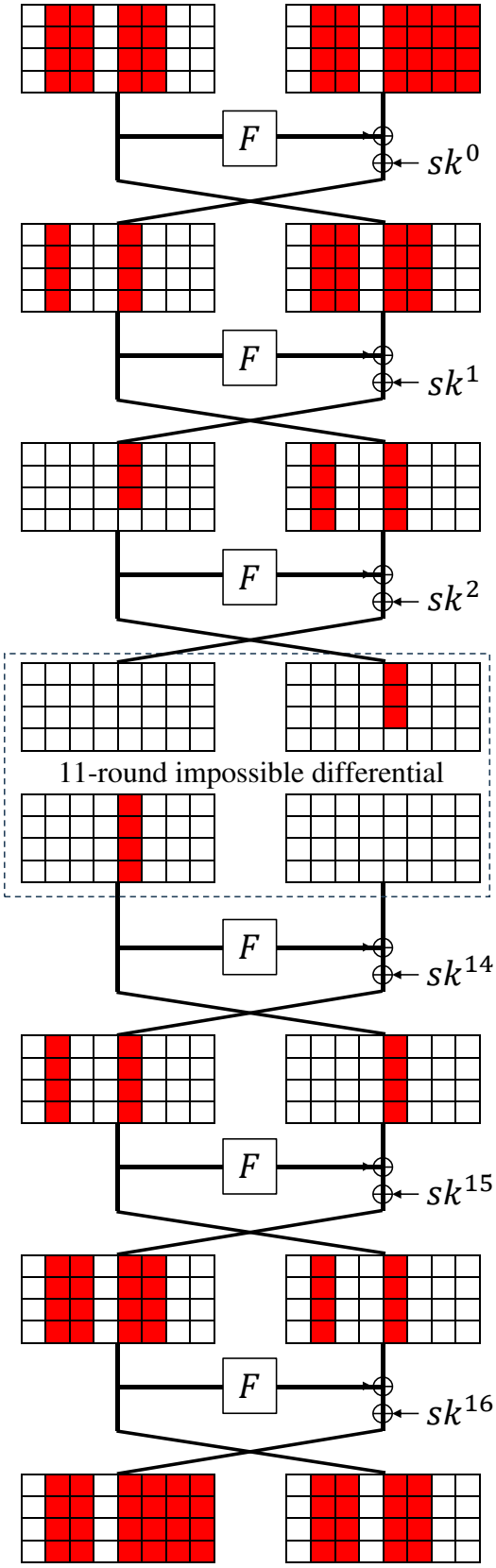



**Fig. 5.** Key Recovery Attack on 17-Round SAND-64

## 7 Conclusion and Future Work

In this paper, we conducted an impossible differential attack on the lightweight block cipher SAND-64. We demonstrated that a key recovery attack on 17-round SAND-64 can be applicable using impossible differential distinguishers. Although this result currently achieves the best attack on round-reduced SAND-64, this attack does not threaten the security of SAND-64 against impossible differential attack.

Future research directions are as follows. The first direction is the improvement of the key recovery attack on SAND-64. In the presented key recovery attack on 17-round SAND-64, the first round is unaffected by the round key. Therefore, by selecting a difference $\Delta_{Pr}$ that cancels out the plaintext difference $\Delta_{Pl}$, the attacker can control the output difference $(\Delta x^1, \Delta y^1)$ of the first round. Using this technique, it may be possible to relax the bit conditions on the plaintext side, denoted as $c_{in}$.

The second direction is to apply the search for impossible differential distinguishers using deterministic differentials to SAND-128 and evaluate its security against impossible differential attack.

## References

1. 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).
2. Biham, E., and Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard", Springer-Verlag, New York, pp. 79-88 (1993).
3. 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).
4. 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).
5. 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).
6. 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).
7. Knudsen, L. R., and Wagner, D.: Integral cryptanalysis, Proc. of Fast Software Encryption, FSE2002, Vol.2365 of LNCS, pp.112-127. Springer-Verlag, (2002).
8. 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).
9. 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).

10. 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).
11. 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).
12. 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).
13. Mirzaie, A., Ahmadi, S., Aref, R. M.: Integral Cryptanalysis of Reduced-Round SAND-64 Based on Bit-Based Division Property, ISC International Journal of Information Security (ISeCure), vol. 15, no. 3, (2023).
14. 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).
15. Boura, C., Lallemand, V., Naya-Plasencia, M., and Suder, V.: Making the Impossible Possible, Cryptology, Vol.31, pp.101–133, Springer-Verlag (2018).