軽量暗号SAND-64とSLAに対する安全性評価

杉尾 信行[1,a)]

概要：SAND は Chen らによって提案された AND-RX 型軽量ブロック暗号である．SAND は構造の違いから SAND-64，SAND-128 の 2 種類が存在する．本稿では，SAND-64 に対して不能差分特性の探索を行い，56 個の 11 段不能差分特性が存在することを明らかにした．また，本稿で発見した不能差分特性を用いて，15 段の SAND-64 に対する鍵回復攻撃が可能であることを示す．
SLA は Ibrahim らによって提案された SPN 型軽量ブロック暗号である．本稿では，2 つの選択平文を用いて SLA に対するフルラウンドの識別攻撃が可能であることを示す．

キーワード：軽量暗号，不能差分攻撃，AND-RX，SAND，SLA

Security Evaluation on the lightweight block ciphers SAND-64 and SLA

Nobuyuki Sugio[1,a)]

Abstract: SAND, a lightweight AND-RX based block cipher proposed by Chen et al., has two versions, SAND-64 and SAND-128, differing in their structure. In this paper, we conduct an exhaustive search for impossible differential paths in SAND-64 and identify 56 types of 11-round impossible differential characteristic. We further demonstrate that these characteristics can be utilized to mount a key recovery attack on 15-round of SAND-64.
SLA is a lightweight SPN based block cipher proposed by Ibrahim et al.. In this paper, we demonstrate a distinguish attack on full-round SLA using only 2 chosen plaintexts.

Keywords: SAND, AND-RX, lightweight cipher, impossible differential attack

1. はじめに

SAND は Disigns, Codes and Cryptography 2022 で Chen らによって提案された軽量ブロック暗号である [1]．内部構造は SIMON [2] と同じ AND-RX 型であり，ブロック長に応じて SAND-64，SAND-128 の 2 種類が存在する．本稿では，SAND-64 を解析対象とする．SAND-64 はブロック長は 64-bit，秘密鍵長は 128-bit，推奨段数は 48 段である．

SAND に対する安全性評価として，提案者らは混合整数線形計画問題（Mixed Integer Linear programming Problem，MILP）を用いて，差分攻撃 [3] 等の安全性評価を実施している．また，不能差分特性 [4] の探索に関して，10 段の不能差分特性が存在することを明らかにしている．

SLA は Ibrahim らによって提案された軽量ブロック暗号である [5]．内部構造は SPN 型であり，ブロック長は 64-bit，秘密鍵長は 80-または 128-bit，推奨段数は 16 段である．秘密鍵長に応じて SLA-80，SLA-128 の 2 種類が存在する．鍵生成を除いて両者の構造に違いはないため，本稿では，両者を SLA として解析対象とする．

SLA に対する安全性評価として，提案者らは差分攻撃 [3] と線形攻撃 [6] に対する評価を行っている．

1.1 既存研究の課題

SAND-64 に関して，提案者らによる不能差分特性の探索では，文献 [7], [8] の手法を適用し，入力と出力にそれぞれ 1-bit ずつ差分を設定しながら対象段数の差分特性が存

[1] 北海道科学大学
Hokkaido University of Science
[a)] sugio-n@hus.ac.jp

在するかどうか調査を行っている．この手法では不能差分特性の探索範囲が限定される為，他の不能差分特性を見落としている可能性がある．

また，SLA に関して，著者の調べた限りにおいて，SLA に関する第三者評価は 2024 年 8 月時点で行われていない．

1.2 関連研究

Hadipour らは決定的差分（deterministic differentials）を用いて不能差分特性を探索する手法を提案した [9], [10]．この手法では，入力と出力の差分をそれぞれ指定する必要が無く，効率的に不能差分特性を探索することが可能である．

1.3 本稿の貢献

本研究では，SAND-64 に対して Hadipour らの決定的差分を用いて不能差分特性を探索し，56 個の 11 段不能差分特性が存在することを明らかにした．得られた 11 段の不能差分特性を用いて，15 段の SAND-64 に対する鍵回復攻撃が可能であることを示す．

また，SLA に対して差分攻撃を用いてフルラウンドの識別攻撃が可能であることを示す．攻撃結果を表 1 に示す．

表 1 Attack Results

Cipher	Data	Time	Memory
SAND-64	2^{58}	2^{127}	2^{63}
SLA	2	2	-

1.4 本稿の構成

本稿の構成を以下に示す．第 2 章にて，差分攻撃と不能差分攻撃について説明する．第 3 章にて，制約プログラミングと決定的差分を用いた不能差分特性探索について紹介する．第 4 章と第 5 章にて，軽量暗号 SAND-64 と SLA の内部構造を説明する．第 6 章にて，SAND-64 に対して不能差分特性を用いた鍵回復攻撃を行う．第 7 章にて，SLA に対する差分特性を用いた識別攻撃を行う．最後に，第 8 章にて，まとめと今後の課題を示す．

2. 暗号攻撃手法

2.1 差分攻撃

差分攻撃は, Biham らによって提案された手法である [3]．平文ペア (P, P^*) に関する排他的論理和差分を $\Delta P = P \oplus P^*$ とする．入力 $X = P \oplus K$ と $X^* = P^* \oplus K$ に関する排他的論理和差分 ΔX は以下の式で表される．

$$\Delta X = X \oplus X^* = (P \oplus K) \oplus (P^* \oplus K) = \Delta P$$

ΔX を入力差分，ΔY を出力差分とする．S-box の差分確率は以下の式で定義される．

$$DP(\Delta X \to \Delta Y) = \frac{\#\{X | S(X) \oplus S(X \oplus \Delta X) = \Delta Y\}}{2^n} \quad (1)$$

式 (1) は S-box に入力される鍵 K とは独立に成立する．平文 P が一様分布に従う場合，出力差分 ΔY は入力差分 ΔP に対して差分確率 DP で期待される．

2.2 不能差分攻撃

不能差分攻撃は，Biham らによって提案された手法である [4]．この攻撃では，確率ゼロの差分を生成する鍵候補を棄却することで，対象暗号の鍵回復攻撃を行う．攻撃者は，暗号の r 段にわたって任意の入力差分 Δ_X に対応する，理論上起こり得ない出力差分 Δ_Y を探索する．仮に，そのような入出力差分のペア (Δ_X, Δ_Y) が存在する場合，それは r 段不能差分識別子と呼ばれる．この識別子を用いることで，対象暗号に対する識別攻撃や鍵回復攻撃が可能となる．

Boura らは，不能差分攻撃に必要な平文数，計算量，メモリ量を定式化した [11], [12]．以下に，その内容を概説する．

図 1 に不能差分攻撃に用いる記号を示す．Δ_X と Δ_Y をそれぞれ入力差分と出力差分とする．r_Δ を不能差分特性の段数とする．Δ_{in} と Δ_{out} をそれぞれ取り得るすべての入力差分と出力差分の集合とする．r_{in} と r_{out} をそれぞれ差分パス (Δ_X, Δ_{in})， 又は (Δ_Y, Δ_{out}) の段数とする．

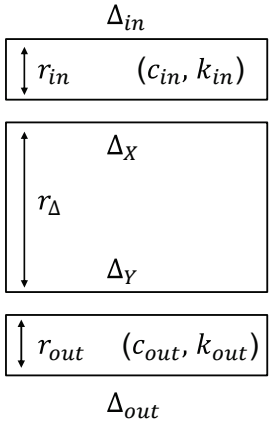

図 1 不能差分攻撃で用いる記号 [11]

差分 $(\Delta_X \to \Delta_{in})$ (または $(\Delta_Y \to \Delta_{out})$) は確率 1 で生じるが，差分 $(\Delta_X \leftarrow \Delta_{in})$ (または $(\Delta_Y \leftarrow \Delta_{out})$) が成立する確率は $\frac{1}{2^{c_{in}}}$ (または $\frac{1}{2^{c_{out}}}$) である．但し，c_{in} (又は c_{out}) は差分 Δ_{in} から Δ_X (または Δ_{out} から Δ_Y) を得るために満たされなければならないビット条件である．

与えられた鍵に対して，差分 Δ_{in} と Δ_{out} を満たす入力ペアが全てのビット条件を満たす確率は $2^{-(c_{in}+c_{out})}$ である．従って，ある鍵が鍵候補集合に残る確率 P は，N 個の異なる入力（または出力）ペアに対して $P = (1 - 2^{-(c_{in}+c_{out})})^N$ となる．Boura らは，

$$P = (1 - 2^{-(c_{in}+c_{out})})^N < \frac{1}{2}$$

を満たす N の最小値，すなわち N_{min} が近似的に $N_{min} = 2^{c_{in}+c_{out}}$ となることを示した．N 個の差分ペア $(\Delta_{in}, \Delta_{out})$ を得るための計算量は，

$$C_N = \max\left\{\min_{\Delta\in\{\Delta_{in},\Delta_{out}\}}\left\{\sqrt{N2^{n+1-|\Delta|}}\right\}, N2^{n+1-|\Delta_{in}|-|\Delta_{out}|}\right\}. \tag{2}$$

で与えられる．この計算量 C_N は，必要な平文数も表している．鍵回復攻撃に必要な計算量は，

$$T = \left(C_N + \left(N + 2^{|k_{in}\cup k_{out}|}\frac{N}{2^{c_{in}+c_{out}}}\right)C'_E + 2^{|K|}P\right)C_E, \tag{3}$$

となる．但し，N は $P = (1 - 2^{-(c_{in}+c_{out})})^N < \frac{1}{2}$ を満たすものとし，C'_E は部分暗号化の計算量とフルラウンドの計算量の比を表し，最後の項は全数探索に必要な計算量を表す．攻撃において保存する必要があるのは，N 個のペアのみである．従って，攻撃のメモリ量は N によって決まる．

3. 制約プログラミングと暗号解読への適用

3.1 制約プログラミング

制約プログラミング (Constraint Programming) は特定の制約を満たす解を見つけることを目的としたプログラミングパラダイムである．制約プログラミングでは，変数 $\mathcal{X}$、制約 $\mathcal{C}$、そして定義域 $\mathcal{D}$ にて構成される．制約プログラミングを用いる際には，対象とする問題に必要な変数を定義し，それらが満たすべき制約を指定し，各変数の取り得る値の範囲を設定する．制約プログラミングが実行可能（feasible）である時，対象とする問題で解が得られる．

3.2 決定的差分（deterministic differentials）

Hadipour らは決定的差分（deterministic differentials）を用いて不能差分特性を探索する手法を提案した [9], [10]．本章では，ビット単位で決定的差分を探索する手法 [10] を概説する．

X と Y を定義域 $\{-1, 0, 1\}$ を有する整数変数とし，それぞれ差分値が不明，0，1 を表すものとする．

CP model 1 (Branching) [10]

$f : \mathbb{F}_2 \to \mathbb{F}_2^n$，$f(x) = (y_0, y_1, \cdots, y_{n-1})$，ただし $y_0 = y_1 = \cdots = y_{n-1} = x$ において，決定的差分特性の有効な遷移は以下を満たす：

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

CP model 2 (XOR) [10]

$f : \mathbb{F}_2 \to \mathbb{F}_2^n$，$f(x_0, x_1, \cdots, x_{n-1}) = y$，ただし $y = x_0 \oplus x_1 \oplus \cdots \oplus x_{n-1}$ において，決定的差分特性の有効な遷移は以下を満たす：

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

CP model 3 (S-box) [10]

S-box の制約モデルは差分分布表（differential distribution table，DDT）から導出される．なお，sbox analyzer を用いて S-box の制約モデルを導出する方法は，文献 [10] の付録 N に示してある．

上記の制約モデルを用いて R 段のブロック暗号 E に対する不能差分識別子を見つける為の CP モデルの構成方法は以下の通りである．暗号 E のブロック長を n ビットとする。定義域 $\{-1, 0, 1\}$ を有する整数変数 XU_r および XL_r $(0 \le r \le R)$ を定義し，それぞれ r 段目の forward（暗号化）方向および backward（復号）方向における内部状態の差分パターンを表すものとする．R 段における forward（暗号化）方向および backward（復号）方向の決定的差分特性に対する CP モデルをそれぞれ独立に構築する．制約条件 $\sum_{i=0}^{n-1} XU_0[i] \neq 0$ と $\sum_{i=0}^{n-1} XL_R[i] \neq 0$ を追加し，自明な解をすべて除外する．$CSP_U(XU_0, \cdots, XU_R)$ と $CSP_L(XL_0, \cdots, XL_R)$ をそれぞれ forward（暗号化）方向および backward（復号）方向の差分伝播に関する制約とする．識別器全体を通して少なくとも 1 点で 2 つの決定的差分伝播間に矛盾が生じることを保証するため，以下の制約を追加する．

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

構築した CP モデルが CP ソルバーにて充足可能である場合，r 段の不能差分識別子が存在することを意味する．

4. 軽量暗号 SAND-64

SAND は Chen らによって提案された軽量ブロック暗号である [1]．内部構造は AND-RX 型であり，ブロック長に応じて SAND-64，SAND-128 の 2 種類が存在する．本稿では SAND-64 を対象とし，ブロック長 64-bit，秘密鍵長 128-bit，推奨段数は 48 段である．また，以下の説明において $n = 32$ とする．

4.1 準備

SAND-64 の説明で用いる記号を以下に示す．

- $x = (x_{n-1}, x_{n-2}, \ldots, x_0)$：$n$-bit 変数であり，$x_{n-1}$ が最上位ビット（MSB），x_0 が最下位ビット（LSB）を示す．変数 x において，$4 \times \frac{n}{4}$ の配列を用いて表される．

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

- $x||y$：変数 x と y の結合

- $x \ll s$：変数 x を s-bit 左シフト
- $x \lll t$：変数 x を t-bit 左巡回シフト
- $x \lll_{\frac{n}{4}} t$：変数 x を 4 つの $\frac{n}{4}$-bit ワード $x = (x_{n-1}, x_{n-2}, \ldots, x_0) = x\{3\}||x\{2\}||x\{1\}||x\{0\}$ に分割し，各ワード $x\{i\}$ 内で t-bit 左巡回シフトを行う．すなわち，$x \lll_{\frac{n}{4}} t = (x\{3\} \lll_{\frac{n}{4}} t)||(x\{2\} \lll_{\frac{n}{4}} t)||(x\{1\} \lll_{\frac{n}{4}} t)||(x\{0\} \lll_{\frac{n}{4}} t)$.
- $x \odot y$：ビット毎の AND 演算
- $x \oplus y$：ビット毎の排他的論理和演算
- $x[i]$：変数 x の i 番目ニブル（4-bit）．
  $x = (x_{n-1}, x_{n-2}, \ldots, x_0)$ において，

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

4.1.1 StateLoading

$P = (Pl, Pr)$ を平文とし，$Pl = (Pl_{n-1}, \ldots, Pl_1, Pl_0)$ を左 n-bit，$Pr = (Pr_{n-1}, \ldots, Pr_1, Pr_0)$ を右 n-bit とする．Pl と Pr を $4 \times \frac{n}{4}$ の配列とみなす．

$$
Pl = \begin{bmatrix} Pl_{n-1} & \ldots & Pl_7 & Pl_3 \\ Pl_{n-2} & \ldots & Pl_6 & Pl_2 \\ Pl_{n-3} & \ldots & Pl_5 & Pl_1 \\ Pl_{n-4} & \ldots & Pl_4 & Pl_0 \end{bmatrix} = \begin{bmatrix} x^0\{3\} \\ x^0\{2\} \\ x^0\{1\} \\ x^0\{0\} \end{bmatrix}
$$

$$
Pr = \begin{bmatrix} Pr_{n-1} & \ldots & Pr_7 & Pr_3 \\ Pr_{n-2} & \ldots & Pr_6 & Pr_2 \\ Pr_{n-3} & \ldots & Pr_5 & Pr_1 \\ Pr_{n-4} & \ldots & Pr_4 & Pr_0 \end{bmatrix} = \begin{bmatrix} y^0\{3\} \\ y^0\{2\} \\ y^0\{1\} \\ y^0\{0\} \end{bmatrix}
$$

1 段目入力 (x^0, y^0) は平文 P から一行ずつ代入される．

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

$C = (Cl, Cr)$ を暗号文とし，$Cl = (Cl_{n-1}, \ldots, Cl_1, Cl_0)$ を左 n-bit，$Cr = (Cr_{n-1}, \ldots, Cr_1, Cr_0)$ を右 n-bit とする．Cl と Cr を $4 \times \frac{n}{4}$ の配列とみなす．暗号文 C は R 段目出力 (x^R, y^R) から以下の様に一行ずつ代入される．

$$
Cl = \begin{bmatrix} Cl_{n-1} & \ldots & Cl_7 & Cl_3 \\ Cl_{n-2} & \ldots & Cl_6 & Cl_2 \\ Cl_{n-3} & \ldots & Cl_5 & Cl_1 \\ Cl_{n-4} & \ldots & Cl_4 & Cl_0 \end{bmatrix} = \begin{bmatrix} x^R\{3\} \\ x^R\{2\} \\ x^R\{1\} \\ x^R\{0\} \end{bmatrix}
$$

$$
Cr = \begin{bmatrix} Cr_{n-1} & \ldots & Cr_7 & Cr_3 \\ Cr_{n-2} & \ldots & Cr_6 & Cr_2 \\ Cr_{n-3} & \ldots & Cr_5 & Cr_1 \\ Cr_{n-4} & \ldots & Cr_4 & Cr_0 \end{bmatrix} = \begin{bmatrix} y^R\{3\} \\ y^R\{2\} \\ y^R\{1\} \\ y^R\{0\} \end{bmatrix}
$$

4.2 段関数

SAND-64 の段関数を図 2 に示す．r 段の入力を (x^r, y^r)，段鍵を sk^r，出力を (x^{r+1}, y^{r+1}) とする．段関数には 2 種類の非線形関数 G_0 と G_1，および線形関数 P_n が存在する．また，$\alpha = 0$，$\beta = 1$ である．

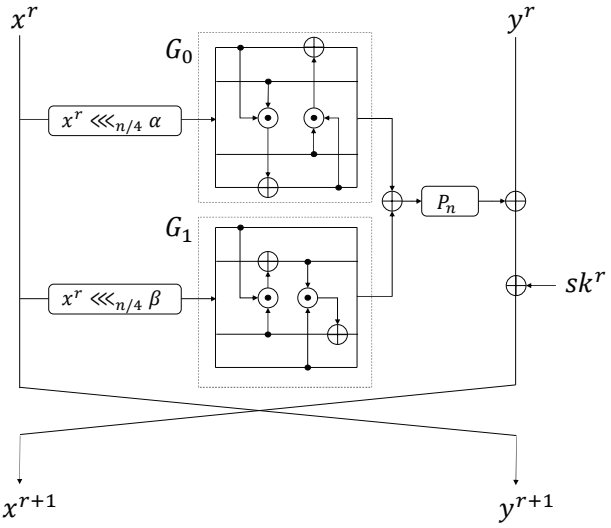

図 2 SAND-64 の段関数

G_0 と G_1 の入力を n-bit 変数 $x = x\{3\}||x\{2\}||x\{1\}||x\{0\}$ とし，出力を $y = y\{3\}||y\{2\}||y\{1\}||y\{0\}$ とする．G_0 において以下の式が成立する．

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

同様に，G_1 において以下の式が成立する．

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

線形関数 P_n の i 番目の入力ワード $x\{i\} = (x_{\frac{n}{4}\cdot i+\frac{n}{4}-1}, \ldots, x_{\frac{n}{4}\cdot i+1}, x_{\frac{n}{4}\cdot i})$ に対し，出力ワード $y\{i\}$ は以下の式で定義される．

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

線形関数 P_n は $\frac{n}{4}$-bit ワードに対する置換 p_8 を 4 つ並列に適用したものと見なすことができる．置換 p_8 を表 2 に示す．

表 2 SAND-64's Permutation p_8

j	0	1	2	3	4	5	6	7
$P_8(j)$	7	4	1	6	3	0	5	2

4.3 鍵生成

SAND-64 は 128-bit の秘密鍵から段鍵を生成する．秘密鍵を 32-bit 毎のワードの結合 $K = K^3||K^2||K^1||K^0$ と見なす．SAND-64 の鍵生成部を図 3，図 4 に示す．

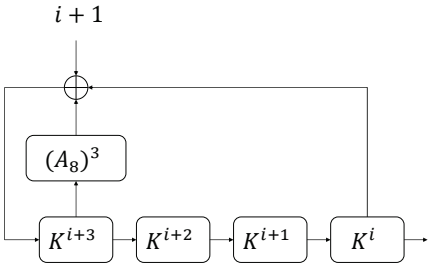

図 3 SAND-64 の鍵生成部

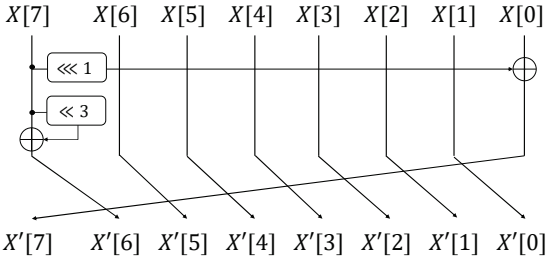

図 4 A_8

$i+1$, $0 \le i \le R-4$ は段毎の定数である．LFSR の更新は以下の式で定義される．

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

A_8 は 4-bit 単位で処理を行なう関数であり，K^{i+3} に対して 3 回繰り返して適用される．段鍵 sk^r，$(0 \le r < R)$ は K^r が以下の通り設定される．

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

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

5. 軽量暗号 SLA

SLA は Ibrahim らによって提案された軽量ブロック暗号である [5]．内部構造は SPN 型であり，ブロック長は 64-bit，秘密鍵長は 80-または 128-bit，推奨段数は 16 段である．秘密鍵長に応じて SLA-80，SLA-128 の 2 種類が存在する．鍵生成を除いて両者の構造に違いはないため，本稿では，両者を SLA として解析対象とする．

5.1 準備

SLA の説明で用いる記号を以下に示す．

- $X = (x_{63}, x_{62}, \dots, x_0)$：64-bit 変数であり，$x_{63}$ が最上位ビット（MSB），x_0 が最下位ビット（LSB）を示す．

5.2 段関数

SLA の段関数を図 5 に示す．K_i XOR は段鍵の排他的論理和を表す．

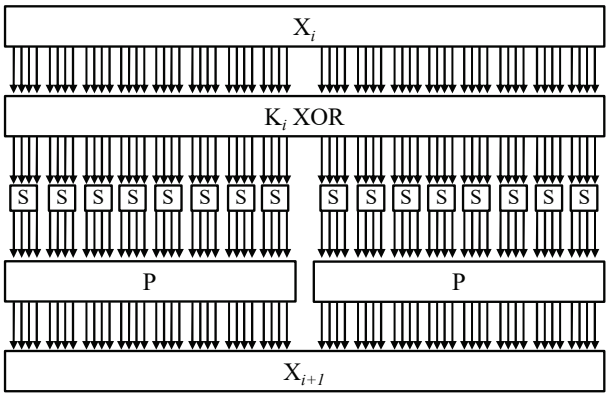

図 5 SLA の段関数

SLA は 4-bit 入出力の S-box を 16 個並列に適用する．S-box テーブルを表 3 に示す．

表 3 SLA's S-box

i	0	1	2	3	4	5	6	7
$S(i)$	f	8	3	e	0	7	b	a
i	8	9	a	b	c	d	e	f
$S(i)$	5	d	9	c	6	4	2	1

線形関数 P は上位 32-bit と下位 32-bit をそれぞれ表 4 に従い，入れ替えを行う．線形関数 P に関して，上位 2-bit（30, 31bit 目）は位置が変わることなく，そのまま出力される．

5.3 鍵生成

本稿では SLA に対する鍵回復攻撃は行わない為，鍵生成部の説明は省略する．

6. SAND-64 の不能差分攻撃

6.1 CP モデルの構築

3 章に示す CP モデルを用いて SAND-64 に対する不能差分特性を探索する為の CP モデルを構築する．以下に示す整数変数の定義域は $\{-1, 0, 1\}$ とし，それぞれ差分値が不明，0，1 を表すものとする．図 2 に示す段関数から，32-bit の整数変数 XU_r，YU_r $(0 \le r \le R)$ を定義する．これらの変数は forward（暗号化）方向における内部状態の差分を表す変数である．同様に，32-bit の整数変数 XL_r，YL_r $(0 \le r \le R)$ を定義する．これらの変数は backward

表 4 SLA's Permutation P

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(j)$	29	27	23	15	22	13	18	5	2	12	16	1	10	28	25	19
j	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(j)$	7	6	4	0	8	24	17	3	14	20	9	26	21	11	30	31

（復号）方向における内部状態の差分を表す変数である．XU_0，YU_0（XL_0，YL_0 も同様）は入力差分を表し，XU_R，YU_R（XL_R，YL_R も同様）は R 段目の出力差分を表すものとする．

整数変数 XU_r，YU_r を用いて forward（暗号化）方向における CP モデルの構築を以下に示す．平文差分を表す 32-bit の整数変数 ΔPl，ΔPr を定義し，以下の制約式を構成する．

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

非線形関数 G_0 と G_1 の出力差分を表す 32-bit の整数変数 G_0U_r，G_1U_r $(1 \le r \le R)$ を定義し，以下の制約式を構成する．

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

G_0 と G_1 の制約式は差分分布表（differential distribution table, DDT）から導出出来る．なお，本稿では文献 [10] の付録 N を参考に，sbox analyzer[*1]を用いて G_0 と G_1 の制約式を導出した．

線形関数 P_n の出力差分を表す 32-bit の整数変数 PU_r $(1 \le r \le R)$ を定義し，以下の制約式を構成する．

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

段関数の出力に関して，以下の制約式を構成する．

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

上記の制約式を r 段 $(0 \le r \le R)$ 作成し，forward（暗号化）方向における CP モデルの構築が完成する．同様の方法にて，整数変数 XL_r，YL_r を用いた backward（復号）方向における CP モデルを構築可能である．

また，以下に示す制約条件を追加し，自明な解をすべて除外する．

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

最後に，識別器全体を通して少なくとも 1 箇所で forward（暗号化）方向と backward（復号）方向の決定的差分伝播間に矛盾が生じることを保証するため，以下の制約を追加する．

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

構築した CP モデルが CP ソルバーにて充足可能である場合，r 段の不能差分識別子が存在することを意味する．

6.2 SAND-64 の不能差分特性

本稿では，前節で構築した CP モデルを MiniZinc[*2]にて実装し，CP ソルバーは OR-Tools[*3]を用いた．本稿で使用した計算機環境を表 5 に示す．

表 5 Computer Environment

Environment	Details
OS	Windows 11
Platform	MiniZinc 2.8.5
Solver	OR-Tools CP-SAT 9.10.4067
CPU	AMD Ryzen 9 5950X
Memory	128 GB

SAND-64 に対する不能差分特性を探索した結果，1 分 30 秒程で解が得られ，11 段の不能差分特性が 56 個存在することが判明した．結果を表 6 に示す．

6.3 15 段 SAND-64 に対する鍵回復攻撃

表 6 に示す 11 段の不能差分特性を用いて，図 6 に示す 15 段 SAND-64 に対する鍵回復攻撃を行う．説明の都合上，StateLoading は割愛する．また，段関数を F と略記する．図 6 において，差分が存在する箇所を赤で示している．鍵回復攻撃に必要な平文数，計算量，およびメモリ量は Boura らの手法 [11], [12] にて見積りを行う．

$r_{in} = 2$ において，平文差分から 2 段目の出力差分 $(\Delta x^0, \Delta y^0) \rightarrow (\Delta x^2, \Delta y^2)$ を得るためのビット条件は $c_{in} = 13 + 8 = 21$ である．また，$k_{in} = 4\text{bits}$ $(sk^0_{j_1} :\ j_1 = 12, 13, 14, 15)$ である．同様に，$r_{out} = 2$ において，暗号文差分から 13 段目の出力差分 $(\Delta x^{15}, \Delta y^{15}) \rightarrow (\Delta x^{13}, \Delta y^{13})$ を得るためのビット条件は $c_{out} = 12 + 8 = 20$ である．また，$k_{out} = 4\text{bits}$ $(sk^{14}_{j_2} :\ j_2 = 12, 13, 14, 15)$ である．

鍵候補を半分以下に減らす為に必要な $N_{min} = 2^{41}$ 個の差分ペア $(\Delta_{in}, \Delta_{out})$ を得るための計算量は，式 (2) より，

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

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

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

表 6　11-round impossible differentials of SAND-64

Pl	0000 0000 0000 0000 0000 0000 0000 0000	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
Cl	0000 0000 0000 0000 ???1 0000 0000 0000	Cr	0000 0000 0000 0000 0000 0000 0000 0000

$$C_N = \max\left\{\left\{\sqrt{2^{41}2^{64+1-24}}\right\}, 2^{41}2^{64+1-24-24}\right\} = 2^{58}$$

である．この計算量 C_N は，必要な平文数も表している．鍵回復攻撃に必要な計算量は式（3）より，

$$T = \left(2^{58} + \left(2^{41} + 2^8\right) \times \frac{4}{15} + 2^{128} \times \frac{1}{2}\right) = 2^{127}$$

回の 15 段 SAND-64 暗号化計算量である．また，N_{min} 個の差分ペアの保持に必要なメモリ量は

$$M = 2^{58} \times 64 \times 4 \times \frac{1}{8} = 2^{63}$$

バイトである．攻撃結果を表 1 に示す．

7. SLA に対する差分攻撃

7.1 フルラウンド SLA に対する識別攻撃

図 5 から，以下に示すフルラウンドの差分特性が存在する．

$$(\alpha, 0) \rightarrow (\beta, 0), \text{ or } (0, \alpha) \rightarrow (0, \beta) \qquad (4)$$

α と β は 32-bit の差分を表す．式 (4) から，2 つの選択平文を用いてフルラウンドの SLA に対する識別攻撃が可能である．

7.2 SLA の改良に関する議論

SLA は差分攻撃に対して脆弱である為，改良について提案を行う．

7.2.1 線形関数 P の改良

まず線形関数 P が上位 32-bit と下位 32-bit を独立に処理するため，64-bit で攪拌される様に修正する必要がある．本稿では，表 7 に示す軽量暗号 PRESENT [13] の線形関数に置き換えることを提案する．

7.2.2 段数の改良

SLA の推奨段数は 16 段である．差分攻撃以外の解読手

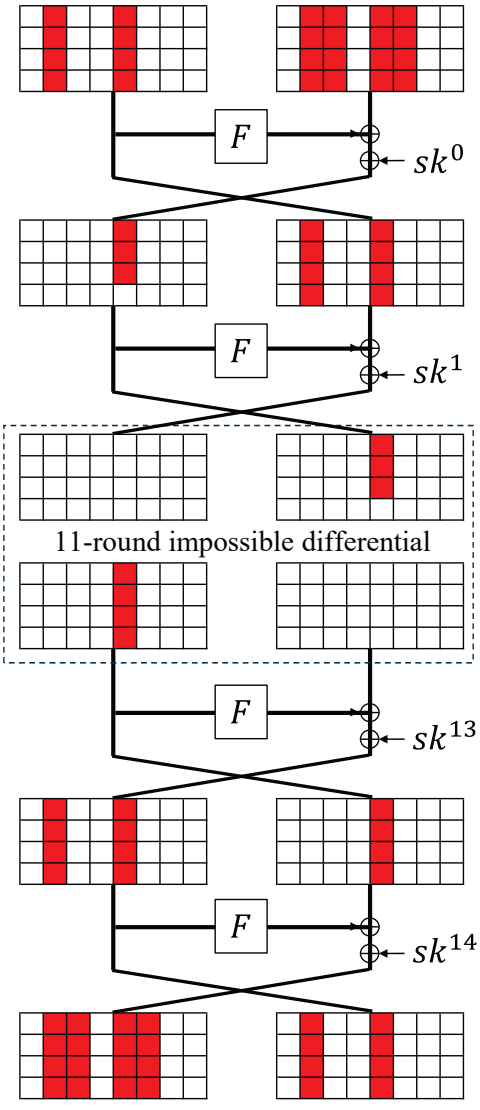

図 6　15 段 SAND-64 に対する鍵回復攻撃

表 7 Bit-permutation used in PRESENT [13]

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(j)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
j	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(j)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
j	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(j)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
j	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(j)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

法に対しても安全性を高める必要があるため，段数を増やす必要があると思われる．PRESENT の推奨段数は 31 段のため，本稿では 31 段以上とすることを提案する．

8. まとめと今後の課題

本稿では，軽量暗号 SAND-64 と SLA に対する安全性評価を行った．SAND-64 に対して不能差分特性を用いて 15 段の SAND-64 に対する鍵回復攻撃が可能であることを示した．また，軽量暗号 SLA に対してフルラウンド差分特性を用いて識別攻撃が可能であることを示した．

今後の課題を以下に示す．1 点目は SAND-64 に対する鍵回復攻撃の改良である．今回示した 15 段 SAND-64 に対する鍵回復攻撃において，1 段目は段鍵の影響を受けない．そのため，平文差分 ΔPl を打ち消す差分を ΔPr に選ぶことにより，1 段目の出力差分 $(\Delta x^1, \Delta y^1)$ を攻撃者が制御可能である．この手法を用いることで，平文側のビット条件 c_{in} を緩和できる可能性がある．2 点目は決定的差分を用いた不能差分特性の探索を SAND-128 に適用し，不能差分攻撃に対する安全性評価を行うことである．

参考文献

[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] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK Families of Lightweight Block Ciphers”, IACR Cryptology ePrint Archive, Report 2013/404 (2013).

[3] Biham, E., and Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard”, Springer-Verlag, New York, pp. 79-88 (1993).

[4] 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).

[5] Ibrahim, N., Agbinya, J.: Design of a Lightweight Cryptographic Scheme for Resource-Constrained Internet of Things Devices, Journal of Applied Sciences, Vol. 13(7), 4398 (2023).

[6] 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).

[7] 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).

[8] Sasaki, Y. and Todo, Y.: New Impossible Differential Search Tool from Design and Cryptanalysis Aspects, Proc. 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2017, Vol.10212 of LNCS, pp.185–215, Springer-Verlag (2017).

[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] 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).

[12] Boura, C., Lallemand, V., Naya-Plasencia, M., and Suder, V.: Making the Impossible Possible, Cryptology, Vol.31, pp.101–133, Springer-Verlag (2018).

[13] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, “PRESENT: An Ultra-Lightweight Block Cipher”, in Proceeding of the CHES, vol. 4727 of LNCS, pp. 450–466, 2007.