A New Method for Designing Lightweight S-Boxes with High Differential and Linear Branch Numbers, and Its Application*

Hangi Kim[1], Yongjin Jeon[1], Giyoon Kim[1], Jongsung Kim[1,2**], Bo-Yeon Sim[1], Dong-Guk Han[1,2], Hwajeong Seo[3], Seonggyeom Kim[4], Seokhie Hong[4], Jaechul Sung[5], and Deukjo Hong[6]

[1] Department of Financial Information Security, Kookmin University, Republic of Korea
[2] Department of Information Security, Cryptology, and Mathematics, Kookmin University, Republic of Korea
[3] Division of IT Convergence Engineering, Hansung University, Republic of Korea
[4] School of Cyber Security, Korea University, Republic of Korea
[5] Department of Mathematics, University of Seoul, Republic of Korea
[6] Department of Information Technology & Engineering, Jeonbuk National University, Republic of Korea

Abstract. Bit permutations are efficient linear functions often used for lightweight cipher designs. However, they have low diffusion effects, compared to word-oriented binary and MDS matrices. Thus, the security of bit permutation-based ciphers is significantly affected by differential and linear branch numbers (DBN and LBN) of nonlinear functions. In this paper, we introduce a widely applicable method for constructing S-boxes with high DBN and LBN. Our method exploits constructions of S-boxes from smaller S-boxes and it derives/proves the required conditions for smaller S-boxes so that the DBN and LBN of the constructed S-boxes are at least 3. These conditions enable us to significantly reduce the search space required to create such S-boxes. In order to make cryptographically good and efficient S-boxes, we propose a unbalanced-Bridge structure that accepts one 3-bit and two 5-bit S-boxes, and produces 8-bit S-boxes. Using the proposed structure, we develop a variety of new lightweight S-boxes that provide not only both DBN and LBN of at least 3 but also efficient bitsliced implementations including at most 11 nonlinear bitwise operations. The new S-boxes are the first that exhibit these characteristics. Moreover, we propose a block cipher PIPO based on one of the new S-boxes, which supports a 64-bit plaintext and a 128 or 256-bit key. Our implementations demonstrate that PIPO outperforms existing block ciphers (for the same block and key lengths) in both side-channel protected and unprotected environments, on an 8-bit AVR. The

* This paper is partially based on the paper "PIPO: A Lightweight Block Cipher with Efficient Higher-Order Masking Software Implementations" [43] presented at the 23rd annual International Conference on Information Security and Cryptology (ICISC 2020).

** Corresponding author, jskim@kookmin.ac.kr

security of PIPO has been scrutinized with regards to state-of-the-art cryptanalysis.

Keywords: Lightweight S-boxes · Differential and linear branch numbers · PIPO · Higher-order masking

1 Introduction

The fourth industrial revolution encompasses a wide range of advanced technologies. One of its core elements is the Internet of Things (IoT), which binds together people, objects, processes, data, applications, and services to make networked connections more relevant and valuable than ever before. However, trustworthy systems are required to enable secure and reliable IoT-based infrastructures, and an essential building block for such systems is cryptography.

Most devices in IoT environments are miniature and resource-constrained. Therefore, lightweight cryptography, which is an active area of research, is essential. Some lightweight block ciphers such as PRESENT [25] and CLEFIA [64] have been standardized by ISO/IEC. In addition, a lightweight cryptography standardization project is ongoing at NIST.

In 1996, Paul Kocher first introduced side-channel attacks, which extract secret information by analyzing side-channel information [51]. Since secure designs for mathematical cryptanalysis cannot guarantee security against side-channel attacks, various countermeasures have been studied. With side-channel attacks becoming more advanced and the associated equipment cost decreasing [71], the application of side-channel countermeasures to ciphers has become crucial. Recently, various studies have been actively conducted on efficient implementations of side-channel countermeasures, especially on efficient masked implementations. To minimize performance overhead, the focus has been on reducing the number of nonlinear operations. Several lightweight block ciphers, with the design goal of low nonlinear operation count, have been proposed [2,3,40].

The lightweightness of block ciphers and the efficiency of their side-channel protected implementations depend significantly on their nonlinear functions. Many of lightweight block ciphers use 4-bit S-boxes [2,9,13,25,42] or 8-bit S-boxes [1,14,40,48,64] as nonlinear functions. One of the main design approaches of lightweight 8-bit S-boxes is to use existing structures, such as Feistel, Lai-Massey and MISTY, employing smaller S-boxes (*e.g.,* 3, 4, or 5-bit S-boxes). However, most related studies have focused on the S-box construction to combine with the linear functions such as word-oriented binary or MDS matrices [1,28,40].

Contributions. In this paper, we introduce a construction method for a different type of lightweight 8-bit S-boxes that are well-suited to a linear bit permutation layer, based on which we develop many of new S-boxes with both DBN and LBN of at least 3 and with efficient masked software implementations. We employ one of them to design a new lightweight versatile block cipher PIPO[1],

[1] PIPO stands for "Plug-In" and "Plug-Out", representing its use in side-channel protected and unprotected environments, respectively.

which can be used in diverse resource-constrained environments, because it is secure and efficient for multiple platforms. Our proposed S-box construction and cipher have the following characteristics and advantages.

1. Our S-box construction methodology enables both DBN and LBN of at least 3, and this property, in combination with a bit permutation, enhances security. It can be used in the construction of a variety of S-boxes from smaller S-boxes. In this study, the Feistel, Lai–Massey, and unbalanced-MISTY structures as well as our proposed unbalanced-Bridge structure have been analyzed. Our framework eliminates all the input and output differences (or masks) where the sum of their Hamming weights is two, during which some conditions of the employed smaller S-boxes are induced. These conditions could accelerate the S-box search, resulting in more than 8,000 new lightweight 8-bit S-boxes with both DBN and LBN of 3. Their bitsliced implementations include 11 nonlinear bitwise operations each. One of them, whose cryptographic properties and efficiency are overall superior or comparable to those of state-of-the-art lightweight S-boxes, was employed for PIPO. Our methodology was also used to find more than 1,000 8-bit S-boxes with DBN of 4 and LBN of 3. To the best of our knowledge, all the aforementioned S-boxes are the first S-boxes with such properties. Furthermore, we found 6 and 7-bit new S-boxes with both DBN and LBN of 3 which are more efficient than existing ones.
2. During the PIPO design process, the focus was on minimizing the number of nonlinear operations bacause this is the most significant factor for efficient higher-order masking implementations. Consequently, PIPO-64/128 achieves fast higher-order masking implementations on an 8-bit AVR, and its execution time increases less with the number of shares (*i.e.*, the masking order) compared with other lightweight 64-bit block ciphers with 128-bit keys. PIPO also shows excellent performance on 8-bit microcontrollers. For the 128-bit key version, the bitsliced implementation for a single-block data requires only 320 bytes of ROM, 31 bytes of RAM, and 197 cycles/byte on an 8MHz ATmega CPU. Accordingly, PIPO-64/128 outperforms other lightweight 64-bit block ciphers with 128-bit keys in terms of 8-bit AVR implementations. It is also competitive in round-based hardware implementations. Using 130nm CMOS technology, the round-based and area optimized implementation of PIPO-64/128 requires only 1,446 gates and achieves 492 Kbps at 100KHz. Although more gates are required to implement PIPO-64/128 than CRAFT-64/128 [13], Piccolo-64/128 [63], and SIMON-64/128 [12], it can be implemented with at least twice the throughput. Accordingly, PIPO-64/128 records a higher FOM. Furthermore, PIPO can be efficiently implemented with minimal memory consumption, other than for storing a plaintext (followed by an intermediate state) and a key. Predefined tables are unnecessary for the nonlinear and linear layers, due to their efficient bitsliced implementations. The advantage of low memory usage elevates PIPO as the preferred choice for low-resource devices.

Organization. In section 2, we introduce a method for constructing S-boxes with DBN and LBN greater than 2. In section 3, we describe the S-box selection procedure for PIPO and new other S-boxes, based on a comparison of our and existing S-boxes. Section 4 specifies the PIPO cipher and its design choices, and section 5 offers our security and performance evaluations of PIPO. Section 6 compares higher-order masking implementations of PIPO and other ciphers. Finally, section 7 concludes the paper, and suggests future studies.

Notation and Definitions. The following notation and definitions are used throughout this paper.

DDT	Difference Distribution Table of an n-bit S-box whose $(\Delta\alpha, \Delta\beta)$ entry is $\#\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus \Delta\alpha) = \Delta\beta\}$, where $\Delta\alpha, \Delta\beta \in \mathbb{F}_2^n$.
LAT	Linear Approximation Table of an n-bit S-box whose $(\lambda_\alpha, \lambda_\beta)$ entry is $\#\{x \in \mathbb{F}_2^n \mid \lambda_\alpha \bullet x = \lambda_\beta \bullet S(x)\} - 2^{n-1}$, where $\lambda_\alpha, \lambda_\beta \in \mathbb{F}_2^n$, and the symbol $\bullet$ denotes the canonical inner product in $\mathbb{F}_2^n$.
Differential uniformity	$\max\limits_{\Delta\alpha \neq 0, \Delta\beta} \#\{x \in \mathbb{F}_2^n \mid S(x) \oplus S(x \oplus \Delta\alpha) = \Delta\beta\}$.
Non-linearity	$2^{n-1} - 2^{-1} \times \max\limits_{\lambda_\alpha, \lambda_\beta \neq 0} \lvert \Phi(\lambda_\alpha, \lambda_\beta) \rvert$, where $\Phi(\lambda_\alpha, \lambda_\beta) = \sum\limits_{x \in \mathbb{F}_2^n} (-1)^{\lambda_\beta \bullet S(x) \oplus \lambda_\alpha \bullet x}$.
DBN	Differential Branch Number of an S-box defined as $\min\limits_{a, b \neq a} (wt(a \oplus b) + wt(S(a) \oplus S(b)))$.
LBN	Linear Branch Number of an S-box defined as $\min\limits_{a, b, \Phi(a,b) \neq 0} (wt(a) + wt(b))$.

2 Construction of S-Boxes with Differential and Linear Branch Numbers Greater than 2

In this section, we describe how to construct S-boxes with DBN>2 and LBN>2. In [61], Sarkar et. al. proposed a method for constructing S-boxes with both DBN and LBN of 3 using resilient Boolean functions, and designed such 5 and 6-bit S-boxes. Our method takes a different approach: it uses smaller S-boxes to create S-boxes with DBN>2 (or LBN>2) by eliminating all the input and output differences (or masks) where the sum of their Hamming weights is 2. During this elimination process, relevant conditions of the employed smaller S-boxes can be induced. In this section, we focus on the construction of 8-bit S-boxes.

Several methods have been proposed in the literature to construct 8-bit S-boxes from smaller ones. These methods typically rely on one of the Feistel, Lai-Massey, or (unbalanced-)MISTY structures, as depicted in Fig. 1-(A), (B), and (C), respectively [1,28,40,47,48,54,57]. In Fig. 1, S_i^j represents the j-th and i-bit S-box, and Fig. 1-(D) depicts our proposed structure, named a unbalanced-Bridge structure. Among the structures in Fig. 1, both (A) and (B) use three 4-bit S-boxes and 12 XOR operations on a bit level, whereas both (C) and (D) use one 3-bit and two 5-bit S-boxes and 6 XOR operations.

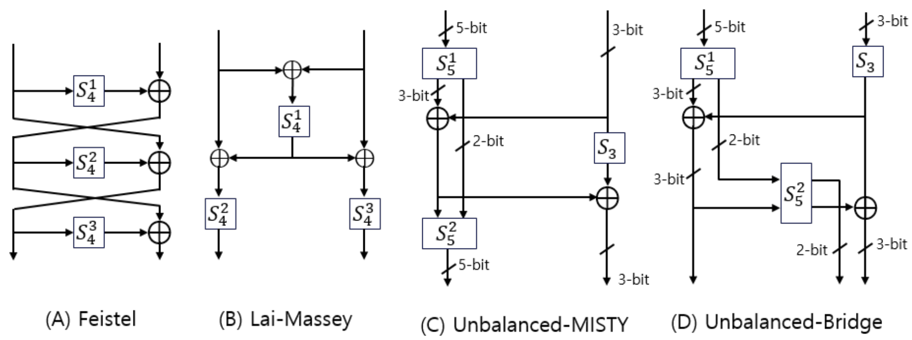

Fig. 1. Constructions of 8-bit S-boxes from smaller S-boxes

In this section, we use the following notation.

$\rho_c : \mathbb{F}_2^5 \to \mathbb{F}_2^5,\ \ \rho_c(x||y) = y||x,\ $ for $x \in \mathbb{F}_2^3$, $y \in \mathbb{F}_2^2$,

$\tau_n : \mathbb{F}_2^5 \to \mathbb{F}_2^n,\ \ \tau_n(x||y) = x,\ $ for $x \in \mathbb{F}_2^n$, $y \in \mathbb{F}_2^{5-n}$, $n \in \{1,2,3,4\}$,

$\tau'_n : \mathbb{F}_2^5 \to \mathbb{F}_2^n,\ \ \tau'_n(x||y) = y,\ $ for $x \in \mathbb{F}_2^{5-n}$, $y \in \mathbb{F}_2^n$, $n \in \{1,2,3,4\}$,

$\mathfrak{F}_A^1 : \mathbb{F}_2^3 \to \mathbb{F}_2^5,\ \ \mathfrak{F}_A^1(X) = (S_5^1)^{-1}(X||A)\ $ for $A \in \mathbb{F}_2^2$,

$\mathfrak{F}_A^2 : \mathbb{F}_2^3 \to \mathbb{F}_2^5,\ \ \mathfrak{F}_A^2(X) = S_5^2(X||A)\ $ for $A \in \mathbb{F}_2^2$,

$0^{(i)}$: i-bit zeros.

The unbalanced-Bridge structure depicted in Fig. 1-(D) can be defined as follows. Let $S_8(X_L||X_R) = C_L(X_L, X_R)||\ C_R(X_L, X_R)$, where X_L and X_R represent the input variables of S_8 which are in $\mathbb{F}_2^5$ and $\mathbb{F}_2^3$, respectively. Then, $C_L(X_L, X_R) = \tau_3(S_5^1(X_L)) \oplus S_3(X_R)$ and $C_R(X_L, X_R) = \rho_c(S_5^2(S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)}))) \oplus (0^{(2)}||S_3(X_R))$ with $C_L : \mathbb{F}_2^5 \times \mathbb{F}_2^3 \to \mathbb{F}_2^3$ and $C_R : \mathbb{F}_2^5 \times \mathbb{F}_2^3 \to \mathbb{F}_2^5$. Proposition 1 shows the conditions for which an 8-bit S-box constructed using Fig. 1-(D) is bijective.

Proposition 1. *The 8-bit S-box constructed using the unbalanced-Bridge structure of Fig. 1-(D) is bijective if and only if the following three conditions are all satisfied:*

i) S_3 *is bijective.*
ii) S_5^1 *is bijective.*

iii) For all $y \in \mathbb{F}_2^3$, $f_y(x) = \tau_2'(S_5^2(y||x))$ *is a bijective function with* $f_y : \mathbb{F}_2^2 \to \mathbb{F}_2^2$.

Proof. Refer to Appendix B.1.

In order to guarantee the bijectivity of S-boxes generated from the Lai-Massey and unbalanced-MISTY structures, all the smaller S-boxes except for S_4^1 should be bijective, whereas the Feistel structure always offers bijective S-boxes regardless of the smaller S-boxes.

Since all the structures in Fig. 1 have two input branches, S-boxes with DBN>2 can be constructed by eliminating four cases $(\Delta 0||\Delta a, \Delta 0||\Delta c)$, $(\Delta 0||\Delta a, \Delta d||\Delta 0)$, $(\Delta b||\Delta 0, \Delta 0||\Delta c)$, $(\Delta b||\Delta 0, \Delta d||\Delta 0)$, where $(\Delta\alpha, \Delta\beta)$ represents the input and output difference pair of the S-boxes, and $wt(\Delta a) = wt(\Delta b) = wt(\Delta c) = wt(\Delta d) = 1$. Some conditions of the employed smaller S-boxes are required to rule out these four cases. We take some examples from the Feistel structure below. The input and output variables of the 3-round Feistel are related as follows.

$$\begin{aligned} C_L(X_L, X_R) &= X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)), \\ C_R(X_L, X_R) &= X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))). \end{aligned}$$

We define a variable Y as $Y = X_R \oplus S_4^1(X_L)$.

A case concerning DBN. $(\Delta 0||\Delta a, \Delta 0||\Delta c)$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta c$. The first equation is expressed as

$$\begin{aligned} &X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)) \oplus X_L \oplus S_4^2(X_R \oplus \Delta a \oplus S_4^1(X_L)) \\ &\quad = S_4^2(X_R \oplus S_4^1(X_L)) \oplus S_4^2(X_R \oplus \Delta a \oplus S_4^1(X_L)) = \Delta 0. \end{aligned}$$

By applying Y, we obtain

$$S_4^2(Y) \oplus S_4^2(Y \oplus \Delta a) = \Delta 0. \qquad (1)$$

Similarly, the second equation is expressed as

$$\begin{aligned} &X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \oplus X_R \oplus \Delta a \oplus S_4^1(X_L) \\ &\quad \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus \Delta a \oplus S_4^1(X_L))) \\ &\quad = S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L) \oplus \Delta a)) \oplus \Delta a = \Delta c. \end{aligned}$$

By applying Eq. (1), we get

$$\Delta a = \Delta c.$$

Therefore, the $(\Delta 0||\Delta a, \Delta 0||\Delta c)$ case is an impossible case if $\Delta a \neq \Delta c$. Otherwise, since the function $(X_L, X_R) \mapsto (X_L, Y)$ is bijective, the $(\Delta 0||\Delta a, \Delta 0||\Delta c)$ case does not happen if and only if there is no Y satisfying Eq. (1). This means the entries of the $(\Delta a, \Delta 0)$ in DDT of S_4^2 have to be zero. Refer to condition $i)$ of Theorem 1. S-boxes with LBN>2 can be made in the same way.

A case concerning LBN. $(0||\lambda_a, 0||\lambda_c)$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_R \bullet \lambda_a = C_R(X_L, X_R) \bullet \lambda_c$. The equation is expressed as

$$X_R \bullet \lambda_a = (X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)))) \bullet \lambda_c.$$

It follows

$$(X_R \oplus S_4^1(X_L)) \bullet \lambda_a \oplus S_4^1(X_L) \bullet \lambda_a = (X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)))) \bullet \lambda_c.$$

The equation becomes

$$Y \bullet \lambda_a \oplus S_4^1(X_L) \bullet \lambda_a = (Y \oplus S_4^3(X_L \oplus S_4^2(Y))) \bullet \lambda_c \tag{2}$$

by using the definition of Y. As mentioned before, the function $(X_L, X_R) \mapsto (X_L, Y)$ is bijective. The $(0||\lambda_a, 0||\lambda_c)$ case has zero bias if and only if the equation (2) is not biased. This means $\#\{(X,Y) \in (\mathbb{F}_2^4)^2 | (Y \oplus S_4^1(X)) \bullet \lambda_a = (Y \oplus S_4^3(X \oplus S_4^2(Y))) \bullet \lambda_c\} = 2^7$. Refer to condition $i)$ of Theorem 2.

The following theorems present the necessary and sufficient conditions of smaller S-boxes so that the 8-bit S-boxes constructed by Feistel, Lai-Massey, unbalanced-MISTY and unbalanced-Bridge structures have both differential and linear branch numbers greater than 2. All the proofs of the following theorems are given in Appendix B.

Theorem 1. *The DBN of bijective 8-bit S-boxes, constructed using the Feistel structure depicted in Fig. 1-(A), is greater than 2 if and only if conditions $i) \sim iv)$ are all satisfied ($\Delta\alpha$ and $\Delta\beta$ below represent arbitrary 4-bit differences where $wt(\Delta\alpha) = wt(\Delta\beta) = 1$). For each $\Delta\alpha$ and $\Delta\beta$;*

i) the entry of the $(\Delta\alpha, \Delta 0)$ in DDT of S_4^2 is 0,
ii) at least one entry of the $(\Delta\alpha, \Delta\beta)$ in DDT of S_4^2 and $(\Delta\beta, \Delta\alpha)$ in DDT of S_4^3 is 0,
iii) at least one entry of the $(\Delta\alpha, \Delta\beta)$ in DDT of S_4^1 and $(\Delta\beta, \Delta\alpha)$ in DDT of S_4^2 is 0,
iv) at least one of $S_4^2(Y) \oplus S_4^2(Y \oplus S_4^1(X) \oplus S_4^1(X \oplus \Delta\alpha)) = \Delta\alpha \oplus \Delta\beta$ and $S_4^3(S_4^2(Y) \oplus X) \oplus S_4^3(S_4^2(Y) \oplus X \oplus \Delta\beta) = S_4^1(X) \oplus S_4^1(X \oplus \Delta\alpha)$ has no solution pair (X,Y), where $X, Y \in \mathbb{F}_2^4$.

Theorem 2. *The LBN of bijective 8-bit S-boxes, constructed using the Feistel structure depicted in Fig. 1-(A), is greater than 2 if and only if conditions $i) \sim iv)$ are all satisfied (λ_α and λ_β below represent arbitrary 4-bit masks where $wt(\lambda_\alpha) = wt(\lambda_\beta) = 1$). For each λ_α and λ_β;*

i) $\#\{(X,Y) \in (\mathbb{F}_2^4)^2 | (Y \oplus S_4^1(X)) \bullet \lambda_\alpha = (Y \oplus S_4^3(X \oplus S_4^2(Y))) \bullet \lambda_\beta\} = 2^7$,
ii) at least one entry of the $(\lambda_\alpha, \lambda_\beta)$ in LAT of S_4^1 and $(\lambda_\beta, \lambda_\alpha)$ in LAT of S_4^2 is 0,
iii) at least one entry of the $(\lambda_\alpha, \lambda_\beta)$ in LAT of S_4^2 and $(\lambda_\beta, \lambda_\alpha)$ in LAT of S_4^3 is 0,

iv) *the entry of the* $(0, \lambda_\alpha)$ *in LAT of* S_4^2 *is* 0.

Theorem 3. *The DBN of bijective 8-bit S-boxes, constructed using the* Lai-Massey *structure depicted in Fig. 1-(B), is greater than 2 if and only if conditions i)* $\sim$ *iv) are all satisfied (*$\Delta\alpha$ *and* $\Delta\beta$ *below represent arbitrary 4-bit differences where* $wt(\Delta\alpha) = wt(\Delta\beta) = 1$*). For each* $\Delta\alpha$ *and* $\Delta\beta$*;*

- *i)* *at least one entry of the* $(\Delta\alpha, \Delta 0)$ *in DDT of* S_4^1 *and* $(\Delta\alpha, \Delta\beta)$ *in DDT of* S_4^3 *is* 0,
- *ii)* *at least one entry of the* $(\Delta\alpha, \Delta\alpha)$ *in DDT of* S_4^1 *and* $(\Delta\alpha, \Delta\beta)$ *in DDT of* S_4^2 *is* 0,
- *iii)* *at least one entry of the* $(\Delta\alpha, \Delta\alpha)$ *in DDT of* S_4^1 *and* $(\Delta\alpha, \Delta\beta)$ *in DDT of* S_4^3 *is* 0,
- *iv)* *at least one entry of the* $(\Delta\alpha, \Delta 0)$ *in DDT of* S_4^1 *and* $(\Delta\alpha, \Delta\beta)$ *in DDT of* S_4^2 *is* 0.

Theorem 4. *The LBN of bijective 8-bit S-boxes, constructed using the* Lai-Massey *structure depicted in Fig. 1-(B), is greater than 2 if and only if conditions i)* $\sim$ *iv) are all satisfied (*λ_α *and* λ_β *below represent arbitrary 4-bit masks where* $wt(\lambda_\alpha) = wt(\lambda_\beta) = 1$*). For each* λ_α *and* λ_β*;*

- *i)* *at least one entry of the* $(0, \lambda_\alpha)$ *in LAT of* S_4^1 *and* $(\lambda_\alpha, \lambda_\beta)$ *in LAT of* S_4^3 *is* 0,
- *ii)* *at least one entry of the* $(\lambda_\alpha, \lambda_\alpha)$ *in LAT of* S_4^1 *and* $(\lambda_\alpha, \lambda_\beta)$ *in LAT of* S_4^2 *is* 0,
- *iii)* *at least one entry of the* $(\lambda_\alpha, \lambda_\alpha)$ *in LAT of* S_4^1 *and* $(\lambda_\alpha, \lambda_\beta)$ *in LAT of* S_4^3 *is* 0,
- *iv)* *at least one entry of the* $(0, \lambda_\alpha)$ *in LAT of* S_4^1 *and* $(\lambda_\alpha, \lambda_\beta)$ *in LAT of* S_4^2 *is* 0.

Theorem 5. *The DBN of bijective 8-bit S-boxes, constructed using the unbalanced-*MISTY *structure depicted in Fig. 1-(C), is greater than 2 if and only if conditions i) and ii) are both satisfied (*$\Delta\alpha$*,* $\Delta\beta$ *and* $\Delta\gamma$ *below represent arbitrary 5, 5 and 3-bit differences, respectively, where* $wt(\Delta\alpha) = wt(\Delta\beta) = wt(\Delta\gamma) = 1$*). For each* $\Delta\alpha$*,* $\Delta\beta$ *and* $\Delta\gamma$*;*

- *i)* *at least one entry of the* $(\Delta\gamma, \Delta\gamma)$ *in DDT of* S_3 *and* $(\Delta\gamma || 0^{(2)}, \Delta\alpha)$ *in DDT of* S_5^2 *is* 0,
- *ii)* *for each* $A, B(\neq A) \in \mathbb{F}_2^2$*, at least one of* $\mathfrak{F}_A^1(X) \oplus \mathfrak{F}_B^1(X) = \Delta\alpha$ *and* $\mathfrak{F}_A^2(X) \oplus \mathfrak{F}_B^2(X) = \Delta\beta$ *has no solution* X*, where* $X \in \mathbb{F}_2^3$.

Theorem 6. *The LBN of bijective 8-bit S-boxes, constructed using the unbalanced-*MISTY *structure depicted in Fig. 1-(C), is greater than 2 if and only if conditions i) and ii) are both satisfied (*λ_α*,* λ_β *and* λ_γ *below represent arbitrary 5,5 and 3-bit masks, respectively, where* $wt(\lambda_\alpha) = wt(\lambda_\beta) = wt(\lambda_\gamma) = 1$*). For each* λ_α*,* λ_β *and* λ_γ*;*

- *i)* *at least one entry of the* $(\lambda_\gamma, \lambda_\gamma)$ *in LAT of* S_3 *and* $(\lambda_\alpha, \lambda_\gamma || 0^{(2)})$ *in LAT of* S_5^1 *is* 0, *ii)* $\sum_{A\in\mathbb{F}_2^2} X \cdot Y = 0$ *where* X *is the entry* $(0, \lambda_\alpha)$ *in LAT of* $\mathfrak{F}_A^1$ *and* Y *is the entry* $(0, \lambda_\beta)$ *in LAT of* $\mathfrak{F}_A^2$.

Theorem 7. *The DBN of bijective 8-bit S-boxes constructed using the unbalanced-Bridge structure of Fig. 1-(D) is greater than 2 if and only if conditions i), ii), and iii) are all satisfied (* $\Delta\alpha$ *and* $\Delta\beta$ *below represent arbitrary differences where* $wt(\Delta\alpha) = wt(\Delta\beta) = 1$*):*

i) *For each* $\Delta\alpha, \Delta\beta \in \mathbb{F}_2^3$, *at least one of the entry* $(\Delta\alpha, \Delta\beta)$ *in DDT of* S_3 *and the entry* $(\Delta\beta||0^{(2)}, \Delta\beta||0^{(2)})$ *in DDT of* S_5^2 *is* 0,

ii) *For each* $\Delta\alpha, \Delta\beta \in \mathbb{F}_2^5$, *for each* $A, B(\neq A) \in \mathbb{F}_2^2$, *at least one of* $\mathfrak{F}_A^1(X) \oplus \mathfrak{F}_B^1(X) = \Delta\alpha$ *and* $\mathfrak{F}_A^2(X) \oplus \mathfrak{F}_B^2(X) = \Delta\beta$ *has no solution* X, *where* $X \in \mathbb{F}_2^3$,

iii) *For each* $\Delta\alpha \in \mathbb{F}_2^3$ *and* $\Delta\beta \in \mathbb{F}_2^5$, *for each* $A, B \in \mathbb{F}_2^2$, *at least one of* $\mathfrak{F}_A^1(X) \oplus \mathfrak{F}_B^1(X \oplus \Delta\alpha) = \Delta\beta$ *and* $\mathfrak{F}_A^2(X) \oplus \mathfrak{F}_B^2(X \oplus \Delta\alpha) = \Delta 0$ *has no solution* X, *where* $X \in \mathbb{F}_2^3$.

Theorem 8. *The LBN of bijective 8-bit S-boxes constructed using the unbalanced-Bridge structure of Fig. 1-(D) is greater than 2 if and only if conditions i), ii), and iii) are all satisfied (* λ_α *and* λ_β *below represent arbitrary masks where* $wt(\lambda_\alpha) = wt(\lambda_\beta) = 1$*):*

i) *For each* $\lambda_\alpha, \lambda_\beta \in \mathbb{F}_2^3$, *at least one of the entry* $(\lambda_\alpha, \lambda_\beta)$ *in LAT of* S_3 *and the entry* $(0, \lambda_\beta||0^{(2)})$ *in LAT of* S_5^2 *is* 0,

ii) *For each* $\lambda_\alpha \in \mathbb{F}_2^5$ *and* $\lambda_\beta \in \mathbb{F}_2^3$, $\sum_{A\in\mathbb{F}_2^2} X \cdot Y = 0$ *where* X *is the entry* $(\lambda_\beta, \lambda_\alpha)$ *in LAT of* $\mathfrak{F}_A^1$ *and* Y *is the entry* $(\lambda_\beta, \lambda_\beta||0^{(2)})$ *in LAT of* $\mathfrak{F}_A^2$,

iii) *For each* $\lambda_\alpha, \lambda_\beta \in \mathbb{F}_2^5$ *satisfying* $\tau_3(\lambda_\beta) = 0$, $\sum_{A\in\mathbb{F}_2^2} X \cdot Y = 0$ *where* X *is the entry* $(0, \lambda_\alpha)$ *in LAT of* $\mathfrak{F}_A^1$ *and* Y *is the entry* $(0, \lambda_\beta)$ *in LAT of* $\mathfrak{F}_A^2$.

In practice, most S-boxes searched from the above theorems have both DBN and LBN of 3. In order to provide higher DBN or LBN of S-boxes, additional conditions are generally required (*e.g.*, a search for S-boxes of DBN of 4 requires additional conditions for eliminating input and output differences where the sum of their Hamming weights is three).

In the above theorems, conditions of smaller S-boxes are different for each structure, leading to different numbers of the required smaller S-box computations. In order to find an S-box with DBN (or LBN) of 3, then the Feistel, Lai-Massey, unbalanced-MISTY and unbalanced-Bridge structures depicted in Fig. 1 require about 11,200, 1,000, 600, and 1,700 (or 13,300, 1,600, 800, and 900) smaller S-box computations, respectively, which were confirmed in our simulations. Employed smaller S-boxes or their combinations are early aborted once they do not meet any of the conditions in Theorems 1∼8. Note that the method described in this section can be applied to any of S-box extension structures.

3 S-Box Selection for PIPO and New Other S-Boxes

We focused on the following three criteria when selecting the 8-bit S-box for PIPO, named S_8.

1. It should offer an efficient bitsliced implementation including 11 or fewer nonlinear operations.
2. Its DBN and LBN should both be greater than 2.
3. Its differential uniformity should be 16 or less, and its non-linearity should be 96 or more.

Criterion 1 minimizes the number of nonlinear operations required by PIPO, which allows for efficient higher-order masking implementations. Criteria 2 and 3 ensure the cryptographic strengths of the S_8 against differential cryptanalysis (DC) and linear cryptanalysis (LC). Any inferior criteria will lead to the implementation of more rounds to achieve acceptable security against these attacks, eventually resulting in a weak proposal. The thresholds of the criteria were selected based on the properties of the existing lightweight 8-bit S-boxes (refer to Table 1).

Previously proposed lightweight 8-bit S-boxes constructed from three smaller S-boxes, such as the Fantomas, Robin [40], FLY [48], LILLIPUT [1] S-boxes, do not meet at least one of the above three design criteria. We obseve that 8-bit S-box constructions using three 4-bit S-boxes would be hard to satisfy criterion 1, even though they conform to criteria 2 and 3; the Feistel and Lai-Massey have been experimentally verified by our simulations.

In order to construct S_8 satisfying all the three criteria, our proposed structure depicted in Fig. 1-(D) is used. It is designed based on three conditions listed below. First, it should use 3 and 5-bit S-boxes instead of 4-bit S-boxes. Second, all eight output bits should be generated from at least two smaller S-boxes (to meet criterion 3). Finally, at least one non-bijective smaller S-box can be adopted to increase the number of possible combinations of smaller S-boxes. Since (D) allows S_5^2 to be either bijective or non-bijective, the search pool in (D) is larger than that in the unbalanced-MISTY structure.

Proposition 2. *The number of possible combinations of S_3, S_5^1, and S_5^2 in the unbalanced-Bridge structure of Fig. 1-(D) is $32! \times 8! \times 98304^8 \approx 2^{265.6}$, whereas that in the structure of unbalanced-MISTY of Fig. 1-(C) is $32! \times 8! \times 32! \approx 2^{250.6}$.*

Proof. Refer to Appendix B.2.

Our S_8 search process is outlined as follows. First, we generated 3-bit and 5-bit S-box sets; for 3-bit S-boxes we ran an exhaustive search with AND, OR, XOR, and NOT instructions while restricting the number of nonlinear (resp. linear) operations to 3 (resp. 4), and for 5-bit S-boxes we ran an exhaustive search with AND, OR, and XOR instruction while restricting the number of nonlinear (resp. linear) operations to 4 (resp. 7) with a differential uniformity of 8 or less. Second, we classified two 5-bit S-boxes and one 3-bit S-box that satisfy the conditions of Proposition 1 as well as Theorems 7 and 8. During this process, the search space for S_8 was significantly reduced because the early abort technique was used to select S_3, S_1^5, and S_2^5. Third, we randomly chose the combination of S_3, S_5^1, and S_5^2 to verify whether the corresponding 8-bit S-boxes satisfy criterion 3. During the search, we found more than 8,000 candidates for

S_8. We selected the one (with no fixed point) that leads to the best resistance to differential and linear attacks when combined with the linear layer of PIPO (refer to section 4.4). The final selected input/output values of S_8 are presented in Table 3; its bitsliced implementation is given in Appendix C.

We also found many of lightweight S-boxes with both DBN and LBN of 3 by using Theorems 1∼6 of the Feistel, Lai-Massey, and unbalanced-MISTY structures. Furthermore, the unbalanced-Bridge structure enabled us to construct more than 1,000 S-boxes with DBN of 4 and LBN of 3. They were found by using the aforementioned additional conditions, but there is one entry of −128 in each of their LATs that might cause ciphers weakened by LC. Appendix C includes a bitsliced implementation of a representative S-box found from each of the four structures. Table 1 compares their cryptographic properties and operations with those of other bitslice 8-bit S-boxes built from smaller three S-boxes.

Table 1. Comparison of bitslice 8-bit S-boxes with respect to cryptographic properties and numbers of operations

	New1	New2	New3	New4	PIPO	FLY	Fantomas	Robin	LILLIPUT
DBN	3	3	3	4	**3**	3	2	2	2
LBN	3	3	3	3	**3**	3	2	2	2
Differential uniformity	16	16	16	64	**16**	16	16	16	8
Non-linearity	96	96	96	0	**96**	96	96	96	96
Algebraic degree	6	5	5	5	**5**	5	5	6	6
#(Fixed points)	16	1	0	2	**0**	1	0	16	1
#(Nonlinear operations)**	12	12	11	8	**11**	12	11	12	12
#(Linear operations)	30	31	24	29	**23**	24	27	24	27
Construction method	Feistel	Lai-Massey	U-MISTY*	U-Bridge	U-**Bridge**	Lai-Massey	U-MISTY	MISTY	Feistel
Reference	**This paper**					[48]	[40]	[40]	[1]

*'U-' represents 'Unbalanced-'.
**Nonlinear (resp. linear) operations represent AND, OR (resp. XOR, NOT).

Designing new 6 and 7-bit S-boxes. Sarkar et al. proposed algorithms to search for 5 and 6-bit S-boxes with DBN and LBN greater than 2, and presented several such S-boxes [61]. They have good cryptographic properties. However, they are not efficient in a bitslice manner, since their search algorithms are based on the algebraic methods. Meanwhile, 7-bit S-boxes have been used in KASUMI and MISTY, but DBN and LBN of 7-bit S-boxes have not been studied.

With minor modifications, the theorems presented in Section 2 can be applied not only to the 6-bit S-boxes but also to the 7-bit S-boxes. We were able to find 6-bit S-boxes with DBN and LBN of 3 using three 3-bit S-boxes in the Feistel structure. Using two 4-bit S-boxes and a 3-bit S-box in the unbalanced-MISTY structure, we were able to find 7-bit S-boxes with DBN and LBN of 3. Since these are based on 3 and 4-bit small S-boxes, it is easy to find their efficient bitsliced implementations (some are described in Appendix C). The 6 and 7-bit S-boxes we found are compared with published ones in Table 2.

Table 2. Comparison of 6 and 7-bit S-boxes with respect to cryptographic properties and numbers of operations

	6-bit S-boxes			7-bit S-boxes	
	Sakar's S_6	Sakar's S_6'	New S_6	MISTY, KASUMI	New S_7
DBN	3	3	3	2	3
LBN	3	3	3	2	3
Differentiality	4	4	4	2	8
Non-linearity	8	8	8	8	16
Algebraic degree	3	2	4	3	4
#(Fixed points)	2	4	2	1	0
#(Nonlinear operations)*	167	36	9	104	11
#(Linear operations)	119	54	12	77	24
Construction method	Cubic function	Toeplitz matrix	Feistel	$A \bullet x^{\alpha}$ over $GF(2^7)$	U-MISTY
Reference	[61]	[61]	Listing 1.8	[37,57]	Listing 1.9

*For the previously published 6 and 7-bit S-boxes the numbers of operations used in their algebraic nomal forms are indicated.

4 Specification of PIPO and Its Design Choices

4.1 Encryption Algorithm

The PIPO block cipher accepts a 64-bit plaintext and either a 128 or 256-bit key, generating a 64-bit ciphertext. It performs 13 rounds for a 128-bit key and 17 rounds for a 256-bit key. Each round is composed of a nonlinear layer denoted as the S-layer, a linear layer denoted as the R-layer, and round key and constant XOR additions. The overall structure of PIPO is depicted on the left side of Fig. 2. Here, RK_0 is a whitening key and $RK_1, RK_2, \cdots, RK_r$ are round keys, where $r = 13$ (128-bit key) or 17 (256-bit key). The i-th round constant c_i is i (the round counter) which is XORed with RK_i. During the enciphering process, the intermediate state is regarded as an 8×8 array of bits, as shown on the right side of Fig. 2, where $X[i]$ represents the i-th row byte for $i = 0 \sim 7$. The S-layer executes eight identical 8-bit S-boxes (denoted as S_8) in parallel. The S_8 is applied to each column of the 8×8 array of bits, where the uppermost bit is the least significant. Table 3 shows the S_8. The R-layer rotates the bits in each row by a given offset (Fig. 3).

4.2 Key Schedule

The key schedule of PIPO is very simple. For PIPO-64/128, split a master key K into two 64-bit subkeys K_0 and K_1, *i.e.*, $K = K_1||K_0$. The whitening and round keys are then defined as $RK_i = K_{i \ mod \ 2}$, where $i = 0, 1, 2, \cdots, 13$. Similarly, for PIPO-64/256, a master key K is divided into four 64-bit subkeys K_0, K_1, K_2, and K_3, *i.e.*, $K = K_3||K_2||K_1||K_0$. Some test vectors for PIPO are provided in Appendix A. Note that resistance to related-key attacks was not considered when designing the PIPO cipher. This aspect will be discussed further in Section D.12.

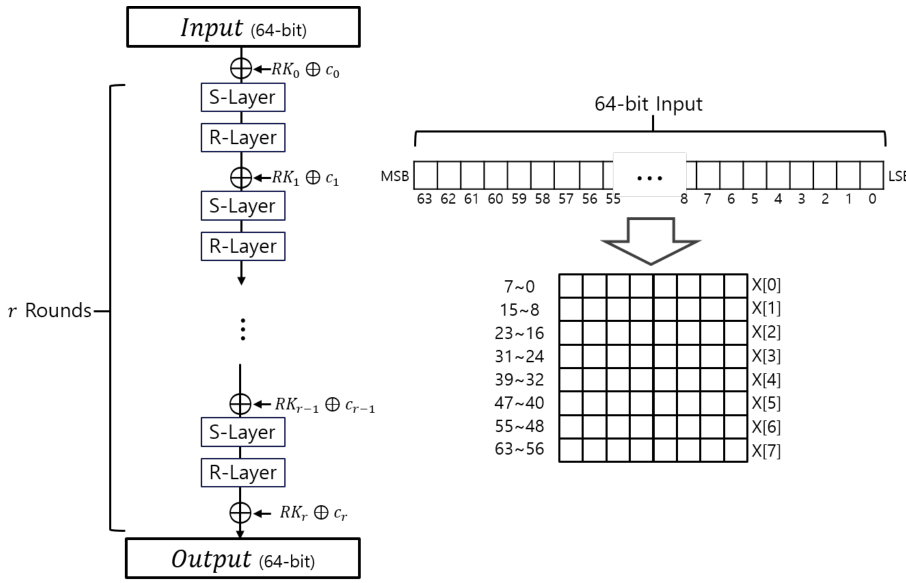

Fig. 2. Overall structure (left) and intermediate state (right) of PIPO

Table 3. 8-bit S-box of PIPO in hexadecimal notation: For example, S_8(31)=86.

$S_8(x\|\|y)$		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	5E	F9	FC	00	3F	85	BA	5B	18	37	B2	C6	71	C3	74	9D
	1	A7	94	0D	E1	CA	68	53	2E	49	62	EB	97	A4	0E	2D	D0
	2	16	25	AC	48	63	D1	EA	8F	F7	40	45	B1	9E	34	1B	F2
	3	B9	86	03	7F	D8	7A	DD	3C	E0	CB	52	26	15	AF	8C	69
	4	C2	75	70	1C	33	99	B6	C7	04	3B	BE	5A	FD	5F	F8	81
	5	93	A0	29	4D	66	D4	EF	0A	E5	CE	57	A3	90	2A	09	6C
	6	22	11	88	E4	CF	6D	56	AB	7B	DC	D9	BD	82	38	07	7E
	7	B5	9A	1F	F3	44	F6	41	30	4C	67	EE	12	21	8B	A8	D5
	8	55	6E	E7	0B	28	92	A1	CC	2B	08	91	ED	D6	64	4F	A2
	9	BC	83	06	FA	5D	FF	58	39	72	C5	C0	B4	9B	31	1E	77
	A	01	3E	BB	DF	78	DA	7D	84	50	6B	E2	8E	AD	17	24	C9
	B	AE	8D	14	E8	D3	61	4A	27	47	F0	F5	19	36	9C	B3	42
	C	1D	32	B7	43	F4	46	F1	98	EC	D7	4E	AA	89	23	10	65
	D	8A	A9	20	54	6F	CD	E6	13	DB	7C	79	05	3A	80	BF	DE
	E	E9	D2	4B	2F	0C	A6	95	60	0F	2C	A5	51	6A	C8	E3	96
	F	B0	9F	1A	76	C1	73	C4	35	FE	59	5C	B8	87	3D	02	FB

4.3 Choice of S-Layer

The S-layer was chosen to be efficient implementations on byte-level operations, without any table lookup. As mentioned before, S_8 offers an efficient bitsliced implementation including only 11 nonlinear and 23 linear bitwise operations.

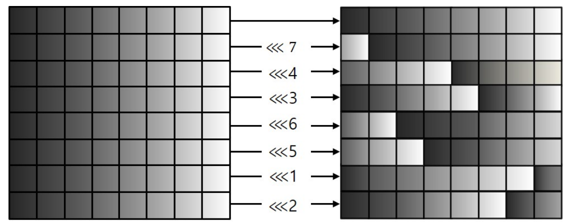

Fig. 3. R-layer

Therefore, it enables the S-layer to be implemented with the same number of byte-level operations, since eight identical S_8s are performed in parallel.

4.4 Choice of R-Layer

To ensure efficient hardware and software implementations, we chose the R-layer to be a bit permutation which only uses bit-rotations in bytes. Listing 1.1 presents the bitsliced implementation of our R-layer, which is free for hardware implementations. During the design of the R-layer, the following criteria were considered.

Listing 1.1. Bitsliced implementation of R-layer (in C code)

```c
//Input: (MSB) X[7], X[6], X[5], X[4], X[3], X[2], X[1], X[0] (LSB)
X[1] = ((X[1] << 7)) | ((X[1] >> 1));
X[2] = ((X[2] << 4)) | ((X[2] >> 4));
X[3] = ((X[3] << 3)) | ((X[3] >> 5));
X[4] = ((X[4] << 6)) | ((X[4] >> 2));
X[5] = ((X[5] << 5)) | ((X[5] >> 3));
X[6] = ((X[6] << 1)) | ((X[6] >> 7));
X[7] = ((X[7] << 2)) | ((X[7] >> 6));
//Output: (MSB) X[7], X[6], X[5], X[4], X[3], X[2], X[1], X[0] (LSB)
```

1. The number of rounds to achieve full diffusion – through which any input bit can affect the entire output bits – should be minimized.
2. Combining the R-layer with the S-layer should enable the cipher to have the best resistance to DC and LC (among all bit permutations satisfying the first criterion).

To meet the first criterion, we adopted a bit permutation that enables PIPO to achieve full diffusion in two rounds by using rotation offsets $0 \sim 7$ for all rows. The second criterion was taken into account when deciding which rotation to use for which row. We applied all 5,040(=7!) R-layers (except for all rotation equivalences) to the S-layer and selected one with the lowest probabilities of 6 and 7-round best differential and linear trails. Table 4 presents the highest probabilities of differential and linear trails according to some of the rotation offset selections[2] (the first row represents the rotation offsets selected for the R-layer). Our analysis found that the selected combination of the S and R layers provides superior resistance to DC and LC than any other combinations even when other S-boxes among the aforementioned candidates were chosen. Note that most combinations of S and R layers candidates could not provide best 7-round differential and linear trails with less than probability 2^{-64}.

Table 4. Best probabilities of differential and linear trails according to rotation offset selections

Rotations	2-round		3-round		4-round		5-round		6-round		7-round	
	DC	LC	DC	LC	DC	LC	DC	LC	DC	LC	DC	LC
(0,7,4,3,6,5,1,2)	8	8	16	16	26.8	24	40.4	38	54.4	52	65	66
(0,1,2,3,4,5,6,7)	8	8	16	16	26.8	24	38.4	36.8	44.8	48.8	52.8	60
(0,2,1,5,3,4,6,7)	8	8	16	16	26.8	24	38	38	50.4	48.8	59	58
⋮												

*The numbers in the table are the values of $-\log_2 p$, where p is the probability of the best differential trail for the DC column, and p is the correlation potential of best linear trail for the LC column.

An important design strategy in PIPO is to perform an exhaustive search for the R-layer. All R-layer candidates that achieve full diffusion in minimal rounds have been examined based on the resistance of DC and LC. This approach to the selection of the linear layer differs from or improves on other state-of-the-art bit permutation-based designs. The linear layer of GIFT was chosen to be a BOGI (Bad Output must go to Good Input) bit permutation, whereas a regular bit permutation was used as the linear layer of PRESENT and those with full diffusion after minimal numbers of rounds were chosen in RECTANGLE and FLY. Our design strategy eventually allowed us to adopt fewer rounds in PIPO.

5 Security and Performance Evaluations of PIPO

5.1 Security Evaluation

Table 5 shows the maximum numbers of rounds of characteristics and key recovery attacks that we found for each attack [4,17,18,20,56,62,69]. In addition to the cryptanalysis shown in Table 5, we conducted algebraic attack [27], integral attack [73], statistical saturation attack [31], invariant subspace attack [52,53], nonlinear invariant attack [67] and slide attack [24], but they were not applied

[2] Our program to search for the best differential and linear trails can be downloaded from GitHub (`https://github.com/PIPO-Blockcipher`).

more effectively than DC or LC. Detailed analysis of all the attacks can be found in Appendix D.

Table 5. The numbers of rounds of the best characteristics for each cryptanalysis

Key length	Cryptanalysis	Best characteristic	Key recovery attack
128-bit	Differential	6-round	9-round
	Linear	6-round	9-round
	Impossible differential	4-round	6-round
	Boomerang/Rectangle	6-round	8-round
	Meet-in-the-Middle	6-round	6-round
256-bit	Differential	6-round	11-round
	Linear	6-round	11-round
	Impossible differential	4-round	8-round
	Boomerang/Rectangle	6-round	10-round
	Meet-in-the-Middle	10-round	10-round

One of the major design considerations for PIPO is to adopt a compact number of rounds (not enough rounds to guarantee security that is (too) high) based on thorough security analyses. We discovered that the best attacks applied to PIPO are DC and LC. An exhaustive search (based on the branch and bound technique [58]) for the DC and LC distinguishers was performed, in which the best reaches 6 rounds. Our analyses could recover the key of up to 9 and 11 rounds of PIPO-64/128 and PIPO-64/256, respectively.

Assume that SM = (FR−AR)/FR, where SM, FR, and AR represent security margin, number of full rounds, and number of attacked rounds (key recovery in the single key setting), respectively. The PIPO's SM is then 0.31, compared with those of the other ciphers listed in Table 6. We stress that the best DC and LC distinguishers of PIPO were searched exhaustively, whereas they were analyzed by upper bounds for their probabilities in several other ciphers [25,48,63]. The latter method might require more rounds (whose distinguishers' probabilities are upper bounded by random probability) than $r+1$ rounds, where r is the number of rounds for the actual best distinguishers. It might lead to several redundant extra rounds being used, causing some loss of efficiency.

In general, there is a trade-off between a cipher's security margin and efficiency. The greater (resp. the smaller) security margin the cipher has, the lower (resp. the higher) efficiency it achieves. Unlike general-purpose ciphers, lightweight ciphers tend to be designed with efficiency first because of limited resources. Considering high efficiency and moderate security levels, we believe that the security margin of PIPO is reasonable.

Table 6. Comparison of ciphers' security margins*

Block cipher	FR	Proposal/State-of-the-art			
		AR	SM	Methods	Refs.
PIPO-64/128	13	9	0.31	DC, LC	This work
PRIDE-64/128	20	NA/19	NA/0.05	NA/DC	[2]/[66]
PRESENT-64/128	31	NA/27	NA/0.13	NA/LC	[25]/[26]
SPECK-64/128	27	NA/20	NA/0.26	NA/DC	[12]/[65]
RECTANGLE-64/128	25	18/18	0.28/0.28	DC/DC	[74]/[74]
SIMON-64/128	44	NA/31	NA/0.30	NA/LC	[12]/[32]
Piccolo-64/128	31	NA/21	NA/0.32	NA/MITM	[63]/[35]
CRAFT-64/128	32	NA/19	NA/0.41	NA/DC	[13]/[41]
SKINNY-64/128	36	16/20	0.56/0.44	IDC, Integral/IDC	[14]/[68]
PIPO-64/256	17	11	0.35	DC, LC	This work

*All the ciphers compared here are from implementation Tables 8, 10, and 11. The best key recovery attack of RoadRunneR has not been presented in literature.

5.2 Software Implementations

In the near future, the growth of the Internet of Things (IoT) is expected to be very rapid. Thus, billions of sensors, actuators, and smart devices, many of which are battery-powered (*e.g.*, wireless sensor nodes), are expected to be used [29,72]. Therefore, any progress in the lightweight block cipher for 8-bit processors (*i.e.*, low-end platform) carries the potential to advance the whole field of IoT security.

The AVR embedded processor is a typical 8-bit microcontroller [5]. It has a RISC architecture with 32 general-purpose registers, of which 6 (`R26`$\sim$`R31`) serve as special pointers for indirect address mode, whereas the remaining 26 are available to users. In general, one arithmetic instruction requires one clock cycle, whereas memory access and 8-bit multiplication instructions require two clock cycles. The details of the instructions used in this paper are available in [5].

The PIPO block cipher consists of permutation (R-layer) and S-box (S-layer) computations. First, the permutation routine is performed in 8-bit rotation operations; our implementation uses the optimized 8-bit rotation operations shown in Table 7. We minimized the number of clock cycles required by converting left rotations to right rotations and vice versa: for example, we converted a 7-bit left rotation to a 1-bit right rotation. To compute the S-box, we used the most optimal method (in terms of logical operations), which requires 22 XOR, 6 AND, 5 OR, 1 COM and 24 MOV instructions. This uses a total of 21 general-purpose registers: six for temporal storage, one for a zero constant, eight for a plaintext, four for address pointers and two for counter variables.

Low-end IoT devices are considered to be resource-constrained platforms, in terms of execution time, code size (*i.e.*, ROM) and RAM. Consequently, cryptographic implementations on low-end devices need to meet not only throughput

Table 7. 8-bit rotations on 8-bit AVR

$\lll 1$	$\lll 2$	$\lll 3$	$\lll 4$	$\lll 5$	$\lll 6$	$\lll 7$

`LSL X1`
`ADC X1, ZERO`
	`LSL X1`
	`ADC X1, ZERO`
	`LSL X1`
	`ADC X1, ZERO`
		`SWAP X1`
		`BST X1, 0`
		`LSR X1`
		`BLD X1, 7`
			`SWAP X1`	`SWAP X1`
				`LSL X1`
				`ADC X1, ZERO`
					`SWAP X1`
					`LSL X1`
					`ADC X1, ZERO`
					`LSL X1`
					`ADC X1, ZERO`
						`BST X1, 0`
						`LSR X1`
						`BLD X1, 7`

2 cycles	4 cycles	4 cycles	1 cycle	3 cycles	5 cycles	3 cycles

targets but also code size and RAM usage ones. The developers of SIMON and SPECK have proposed a new metric to measure overall performance on low-end devices, namely RANK [11]. This is calculated as follows:

$$RANK = (10^6/CPB)/(ROM + 2 \times RAM).$$

In this metric, higher values of RANK correspond to better performance. Table 8 compares results for several block ciphers on an 8-bit AVR platform. Here, we used Atmel Studio 6.2, and compiled all implementations with optimization level 3. The target processor was an ATmega128 running at 8MHz. PIPO requires 320 bytes of code, 31 bytes of RAM and an execution time of 197 CPB. We used the RANK metric to compare the ciphers' overall performances, finding that PIPO achieved the highest score among block ciphers with the same parameter lengths.

Table 8. Comparison of block ciphers on 8-bit AVR*

Block cipher	Code size (bytes)	RAM (bytes)	Execution time (cycles per byte)	RANK
PIPO-64/128	320	31	197	13.31
SIMON-64/128 [11]	290	24	253	11.69
RoadRunneR-64/128 [10]	196	24	477	8.59
RECTANGLE-64/128 [34]	466	204	403	2.84
PRIDE-64/128 [34]	650	47	969	1.39
SKINNY-64/128 [34]	502	187	877	1.30
PRESENT-64/128 [36]	660	280	1,349	0.61
CRAFT-64/128 [13]	894	243	1,504	0.48
PIPO-64/256	320	47	253	9.54

*The code size represents ROM, and RAM metric includes STACK.

5.3 Hardware Implementations

We implemented PIPO in Verilog, and synthesized the proposed architectures using the Synopsys Design Compiler with 130nm CMOS technology. Fig. 4 shows the datapath of an area-optimized encryption-only PIPO block cipher, which performs one round per clock cycle (*i.e.*, uses a 64-bit-wide datapath). The S-layer uses the same 8-bit S-box 8 times, whereas the R-layer is implemented in wiring. For lightweight key generation, we obtain the round key from the master key, directly. This feature avoids including the key storage. Our implementations require 13 clock cycles to encrypt a 64-bit plaintext.

Table 9 shows the areas required by PIPO-64/128 and PIPO-64/256. Most of the areas are taken up by the S-layer, in order to compute eight 8-bit S-boxes in parallel.[7] The flip-flops are used for storing plaintext and counter, and the other areas consist of MUX and other logical operations.

Table 10 compares the results for several different block ciphers implemented as ASICs. Compared with the other block ciphers using the same parameter lengths, PIPO needs more gates than CRAFT, Piccolo and SIMON but its cycles per block are much lower, resulting in the highest figure of merit FOM (nano bits per clock cycle per GE squared [6,42]). It is obvious that the high FOM of PIPO requires less energy and battery consumption.

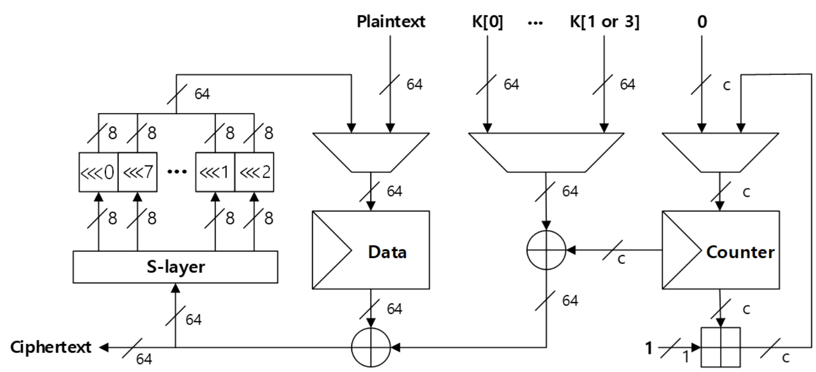

Fig. 4. Datapath of an area-optimized version of PIPO

6 Performance Evaluations of Higher-Order Masking Implementations of PIPO

Side-channel attacks were published by Kocher in 1996 [51] and can reveal secret information by analyzing side-channel leakages, such as power consumption and

[7] The NAND gate is the most basic unit for hardware implementations. In 130nm ASIC library, which was used in our hardware implementations, AND, OR, and XOR operations require 1.66, 1.66, and 2.66 NAND gates, respectively.

Table 9. Area requirement of PIPO-64/128 and PIPO-64/256.

	PIPO-64/128		PIPO-64/256	
Module	GE	%	GE	%
Data and Counter States	341	24	360	22
S-layer	581	40	581	36
Add Round Key	170	12	170	11
Others	354	24	491	31
Total	1,446	100	1,602	100

Table 10. Comparison of round-based and area optimized implementations for block ciphers using 130nm ASIC library.

Block cipher	Area [GE]	Throughput (Kbps@100KHz)	cycles /block	FOM $[\frac{bits \times 10^9}{clk \times GE^2}]$
PIPO-64/128	1,446	492	13	2,355
CRAFT-64/128 [13]	949	200	32	2,221
Piccolo-64/128 [63]	1,197	194	33	1,354
SIMON-64/128 [12]	1,417	133	48	664
RECTANGLE-64/128 [74]	2,064	246	26	578
PIPO-64/256	1,602	376	17	1,467

electromagnetic emission [55]. This information reveal is due to the fact that side-channel leakages depend on data values being manipulated, *i.e.*, intermediate values, while the cryptographic algorithm is running. Thus, to cope with this, randomization techniques, which make side-channel leakages of a cryptographic device independent of the intermediate values of the cryptographic algorithm are generally used. Among them, a higher-order Boolean masking technique is the most popular.

For low spec-devices which have tiny processors, noise is relatively lower and the feasibility of higher-order side-channel attacks increases. Therefore, the main aim of our proposed PIPO is to enable efficient implementations that are secure against high-order side-channel attacks. Thus, we now compare the execution times, for different numbers of shares, when we apply higher-order Boolean masking schemes [44,55].

6.1 Higher-Order Masking

Higher-order masking is a randomization technique, which splits the sensitive intermediate variable x into $d+1$ random variables $x_1, x_2, \cdots, x_{d+1}$ called shares and satisfies $x = x_1 * x_2 * \cdots * x_{d+1}$ for the operation $*$ defined according to the cryptographic algorithm. In this paper, $*$ is considered as the exclusive-or (XOR) operation denoted by $\oplus$. This masking scheme is called Boolean masking, and it is the most generally used. The number of shares is $d + 1$, and the masking order is d.

6.2 Bitsliced Implementations for Efficient Higher-Order Masking

Bitsliced implementations, initially proposed by Biham [16], are known to be efficient when applying Boolean masking, since secure S-box computations can be carried out in parallel [38,39,40,45]. Thus, we used an S-box that can be efficiently implemented in this way, and only involves 11 nonlinear bitwise operations. The number of nonlinear operations is very important for Boolean masking schemes, since they have a quadratic complexity, *i.e.*, $O(d^2)$, compared with the linear complexity, *i.e.*, $O(d)$, for other operations.

We constructed PIPO using higher-order masked S-layer and R-layer, which is shown in Appendix E. The nonlinear operations, logical AND and OR, were replaced by ISW-AND and ISW-OR, respectively. ISW-AND is d-probing secure with a masking order d and has a quadratic complexity for d. There are several variations of ISW-AND [7,8,15], however, in this paper, we apply original ISW-AND. Since logical OR of two inputs a and b satisfies $a \vee b = (a \wedge b) \oplus a \oplus b$, thus, ISW-OR can be calculated by replacing logical AND with ISW-AND. We refreshed one of two inputs of ISW-AND and ISW-OR, which might be linearly related, to guarantee full security by using refresh masking [38]. It is possible to implement higher-order masked logical XOR and rotations by repeating as many as the number of shares, because they are the linear operations. Higher-order masked logical NOT operation can be calculated by taking logical NOT operation on only one of the shares.

We compare our proposed PIPO with PRIDE, RoadRunneR, RECTANGLE, CRAFT, SIMON, PRESENT, and SKINNY [2,10,12,13,14,25,74], which are 64-bit block ciphers with 128-bit keys. All the ciphers compared were implemented using bitslice techniques, and only round constants were precomputed. There is no need to precompute round constants of PIPO, RoadRunneR, and PRESENT, because they are the i or $NR - i$ for $i = 0, 1, \cdots, NR - 1$, where NR is the number of rounds. Therefore, the required ROM for round constants is shown in Table 11. Only CRAFT used an additional 16-byte diffusion table Q for generating tweakeys. The same secure logical operations of PIPO were applied to implement higher-order masking structures.

Fig. 5 shows the execution times for different numbers of shares on an 8-bit AVR processor. Especially, it shows that the more nonlinear operations, the greater increase in execution time with the number of shares (refer to Table 11).[3] PIPO has the smallest number of nonlinear operations.

[3] A family of block ciphers named LowMC, whose main design goal is a low nonlinear operation count, was introduced [3]. However, they are not in our comparison list, because they do not have any 64-bit block/128-bit key version. We also exclude ARX-based ciphers in our comparison Tables 8, 10, and 11 because their side-channel countermeasures are far inferior to those of S-box-based ciphers

Table 11. Comparison of required ROM (bytes) for round constant, number of nonlinear bitwise operations, and linear layers of round functions

Block cipher	Table size	#(nonlinear bitwise operations)	Linear layer
PIPO-64/128	0	1,144	7 bit-rotations in bytes
PRIDE-64/128	80	1,280	MixColumns*
SIMON-64/128	62	1,408	3 bit-rotations in 32-bit words
RoadRunneR-64/128	0	1,536	24 bit-rotations in bytes
RECTANGLE-64/128	25	1,600	3 bit-rotations in 16-bit words
CRAFT-64/128	64	1,984	MixColumns*, PermuteNibbles
PRESENT-64/128	0	1,984	Bit permutation
SKINNY-64/128	62	2,304	ShiftRows, MixColumns*

* : multiply with binary matrix

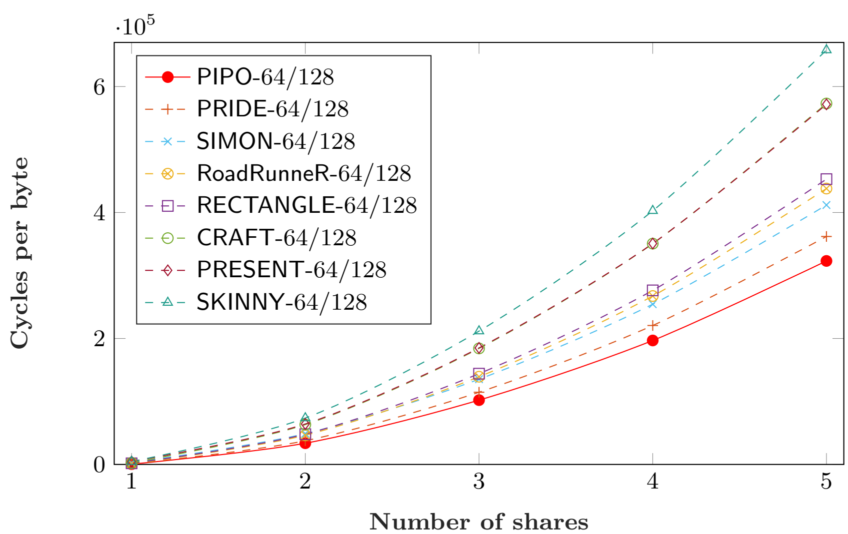

Fig. 5. Execution times of one-block encryptions according to the number of shares in an Atmel AVR XMEGA128 (1 means unprotected)

Moreover, the R-layer of PIPO consists only of seven bit-rotations in bytes, which is efficient compared to the other ciphers as shown in Table 11. Thus, it can be inferred that PIPO has the lowest time complexity. Here, the execution time of PIPO increases more slowly with the number of shares compared with the other ciphers. As a result, PIPO does not need ROM for precomputed table and offers excellent performance in 8-bit AVR software implementations while providing security against side-channel attacks.

7 Conclusion and Future Work

In this paper, we presented a widely applicable method for constructing lightweight S-boxes with DBN and LBN greater than 2, from smaller S-boxes. Using existing structures such as Feistel, Lai-Massey, unbalanced-MISTY as well as the proposed unbalanced-Bridge structure, we were able to find many lightweight S-boxes with both DBN and LBN of at least 3. Among them, the most efficient and secure 8-bit S-box was selected to create new lightweight versatile block cipher PIPO suitable for diverse resource-constrained environments. In particular, PIPO exhibits exelent performance in both side-channel protected and unprotected environments on 8-bit microcontrollers, and fast round-based hardware implementations as well. Furthermore, a thorough security analysis of PIPO was conducted.

For future work, it would be interesting to investigate the following research questions.

- Are there any other 8-bit S-boxes that have the same level of cryptographic properties as S_8 (Table 1) but require fewer nonlinear operations?
- Are there secure and efficient 8-bit S-boxes with both DBN and LBN of 4?

We believe that our proposed method can help cipher designers build lightweight S-boxes with high DBN and LBN, and that the PIPO cipher can be used for data condentiality in a wide range of low-end IoT environments (*e.g.* wireless sensors/-passive RFID tags and their hubs, Underwater Acoustic Networks (UAVs) which may only ask that 64-bit quantities be encrypted [23,46,59]).

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00520, Development of SCR-Friendly Symmetric Key Cryptosystem and Its Application Modes)

References

1. Adomnicai, A., Berger, T. P., Clavier, C., Francq, J., Huynh, P., Lallemand, V., Gouguec, K. L., Minier, M., Reynaud, L., Thomas, G., *Lilliput-AE: a New Lightweight Tweakable Block Cipher for Authenticated Encryption with Associated Data*, Submission to the NIST Lightweight Cryptography Standardization Process, 2019.
2. Albrecht, M. R., Driessen, B., Kavun E. B., Leander., G., Paar, C., Yalçin, T., *Block Ciphers - Focus on the Linear Layer (feat. PRIDE)*, CRYPTO 2014, LNCS 8616, pp. 57–76, Springer, 2014.
3. Albrecht, M. R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M., *Ciphers for MPC and FHE*, EUROCRYPT 2015, LNCS 9056, pp. 430-454, Springer, 2015.
4. Aoki, K., Sasaki, Y., *Preimage attacks on one-block MD4, 63-step MD5 and more*, Selected Areas in Cryptography 2008, LNCS 5381, Springer, 2008.

5. Atmel Corporation, ATmega128(L) Datasheet, `www.microchip.com/wwwproducts/en/ATmega128`, Visited on April 23, 2019.
6. Badel, S., Dagtekin, N., Nakahara, J. Jr., Ouafi, K., Reffé, N., Sepehrdad, P. Susil, P., Vaudenay, S., *ARMADILLO: A Multi-purpose Cryptographic Primitive Dedicated to Hardware*, CHES 2010, LNCS 6225, pp. 398–412, Springer, 2010.
7. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F., Strub, P., *Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model*, EUROCRYPT 2017, LNCS 10210, pp. 535–566, Springer, 2017.
8. Battistello, A., Coron, J., Prouff, E., Zeitoun, R., *Horizontal Side-Channel Attacks and Countermeasures on the ISW Masking Scheme*, CHES 2016, LNCS 9813, pp. 23–39, Springer, 2016.
9. Banik, S., Pandey, S. K., Peyrin, T., Sasaki, Y., Sim, S. M., Todo, Y., *GIFT: A Small Present Towards Reaching the Limit of Lightweight Encryption*, CHES 2017, LNCS 10529, pp. 321–345, Springer, 2017.
10. Baysal, A., Sahin, S., *RoadRunneR: A Small And Fast Bitslice Block Cipher For Low Cost 8-bit Processors*, LightSec 2015, LNCS 9542, pp. 58–76, Springer, 2016.
11. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L., *The SIMON and SPECK block ciphers on AVR 8-bit microcontrollers*, LightSec 2014, LNCS 8898, pp. 3–20, Springer, 2014.
12. Beaulieu, R., Shors D., Smith J., Treatman-Clark, S., Weeks, B., Wingers, L., *The SIMON and SPECK families of lightweight block ciphers*, Cryptology ePrint Archive, 2013.
13. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S., *CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks*, IACR Trans. Symmetric Cryptol. 2019(1), pp. 5–45, 2019.
14. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S. M., *The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS*, CRYPTO 2016, LNCS 9815, pp. 123–153, Springer, 2016.
15. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D., *Randomness Complexity of Private Circuits for Multiplication*, EUROCRYPT 2016, LNCS 9666, pp. 616–648, Springer, 2016.
16. Biham, E., *A Fast New DES Implementation in Software*, FSE 1997, LNCS 1267, pp. 360–272, Springer, 1997.
17. Biham, E., Biryukov, A., Shamir, A., *Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials*, EUROCRYPT 1999, LNCS 1592, pp. 12–23, Springer, 1999.
18. Biham, E., Dunkelman, O., Keller, N., *The Rectangle Attack - Rectangling the Serpent*, EUROCRYPT 2001, LNCS 2045, pp. 340–357, Springer, 2001.
19. Biham, E., Dunkelman, O., Keller, N., *Related-Key Boomerang and Rectangle Attacks*, EUROCRYPT 2005, LNCS 3494, pp. 507–525, Springer, 2005.
20. Biham, E., Shamir, A., *Differential Cryptanalysis of DES-like Cryptosystems*, CRYPTO 1990, LNCS 537, pp. 2–21, Springer, 1991.
21. Biham, E., *New Types of Cryptanalytic Attacks Using Related Keys*, J. Cryptology, 7(4), pp. 229–246, 1994.
22. Biryukov, A., Khovratovich, D., *Related-key Cryptanalysis of the Full AES-192 and AES-256*, ASIACRYPT 2009, LNCS 5912, pp. 1–18, Springer, 2009.
23. Biryukov, A., Perrin, L., *State of the Art in Lightweight Symmetric Cryptography*, IACR Cryptology ePrint Archive, pp. 511, 2017.
24. Biryukov, A., Wagner, D., *Advanced Slide Attacks*, EUROCRYPT 2000, LNCS 1807, pp. 589–606, Springer, 2000.

25. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C., *PRESENT: An Ultra-Lightweight Block Cipher*, CHES 2007, LNCS 4727, pp. 450–466, Springer, 2007.
26. Bogdanov, A., Tischhauser, E., Vejre, P. S., *Multivariate Linear Cryptanalysis: The Past and Future of PRESENT*, IACR Cryptology ePrint Archive, 2016, pp. 667, 2016.
27. Boura, C., Canteaut, A., Cannière, C. D., *Higher-Order Differential Properties of Keccak and* Luffa, FSE 2011, LNCS 6733, pp. 252–269, Springer, 2011.
28. Canteaut, A., Duval, S., Leurent, G., *Construction of Lightweight S-Boxes Using Feistel and MISTY Structures*, SAC 2015, LNCS 9566, pp. 373–393, Springer, 2016.
29. Cheng, H., Großschädl, J., Rønne, P. B., Ryan, P. Y., *A Lightweight Implementation of NTRUEncrypt for 8-bit AVR Microcontrollers*, Second PQC Standardization Conference, 2019.
30. Cid, C., Murphy, S., Robshaw, M., *Algebraic aspects of the advanced encryption standard*, Springer Science & Business Media, 2006.
31. Collard, B., Standaert, F. X., *A Statistical Saturation Attack against the Block Cipher PRESENT*, CT-RSA 1909, LNCS 5473, pp. 195–210, Springer, 2009.
32. Chen, H., Wang, X., *Improved Linear Hull Attack on Round-Reduced Simon with Dynamic Key-Guessing Techniques*, FSE 2016, LNCS 9783, pp. 428–449, Springer, 2016.
33. Daemen, J., Rijmen, V., *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer, 2002.
34. Dinu, D., Biryukov, A., Großschädl, J., Khovratovich, D., and Corre, Y. L., Perrin, L., *FELICS–fair evaluation of lightweight cryptographic systems*, NIST Workshop on Lightweight Cryptography, 2015.
35. Isobe, T., Shibutani, K., *Security Analysis of the Lightweight Block Ciphers XTEA, LED and Piccolo*, ACISP 2012, pp. 71–86, Springer, 2012.
36. Engels, S., and Kavun, E. B., Paar, C., Yalçin, T., Mihajloska, H., *A nonlinear/linear instruction set extension for lightweight ciphers*, IEEE 21st Symposium on Computer Arithmetic, pp. 67–75, 2013.
37. ETSI. TS 135 202 V7. 0.0: Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI specification (3GPP TS 35.202 version 7.0. 0 Release 7).
38. Goudarzi, D., Journault, A., Rivain, M., Standaert, F., *Secure Multiplication for Bitslice Higher-Order Masking: Optimisation and Comparison*, COSADE 2018, LNCS 10815, pp. 3–22, Springer, 2018.
39. Goudarzi, D., Rivain, M., *How Fast Can Higher-Order Masking Be in Software?*, EUROCRYPT 2017, LNCS 10210, pp. 567–597, Springer, 2017.
40. Grosso, V., Leurent, G., Standaert, F., Varici, K., *LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations*, FSE 2014, LNCS 8540, pp. 18–37, Springer, 2014.
41. Guo, H., Sun, S., Shi, D., Sun, L., Sun, Y., Hu, L., Wang, M. *Differential Attacks on CRAFT Exploiting the Involutory S-boxes and Tweak Additions*, IACR Trans. Symmetric Cryptol., 2020(3), pp. 119–151, 2020.
42. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M., *The LED Block Cipher*, CHES 2011, LNCS 6917, pp. 326–341, Springer, 2011.
43. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B. Y., Han, D. G., Seo, H., Kim, S., Hong, S., Sung, J., Hong, D., *PIPO: A Lightweight Block Cipher with Efficient Higher-Order Masking Software Implementations*, ICISC 2020, To appear.
44. Ishai, Y., Sahai, A., Wagner, D., *Private Circuits: Securing Hardware against Probing Attacks*, CRYPTO 2003, LNCS 2729, pp. 463–481, Springer, 2003.

45. Journault, A.,Standaert, F., *Very High Order Masking: Efficient Implementation and Security Evaluation*, CHES 2017, LNCS 10529, pp. 623–643, Springer, 2017.
46. Juels, A., Weis, S. A., *Authenticating pervasive devices with human protocols*, CRYPTO 2005, LNCS 3621, pp. 293–308, Springer, 2005.
47. Junod, P., Vaudenay, S., *FOX : A New Family of Block Ciphers*, SAC 2014, LNCS 3357, pp. 114–129, Springer, 2004.
48. Karpman, P., Grégoire, B., *The littlun s-box and the fly block cipher*, Lightweight Cryptography Workshop, 2016.
49. Kim, J., Kim, G., Hong, S., Lee, S., Hong, D., *The Related-Key Rectangle Attack - Application to SHACAL-1*, ACISP 2004, LNCS 3108, pp. 123–136, Springer, 2004.
50. Kim, J., Hong, S., Preneel, B., Biham, E., Dunkelman, O., Keller, N., *Related-Key Boomerang and Rectangle Attacks: Theory and Experimental Analysis*, IEEE Trans. Information Theory, 58(7), pp. 4948–4966, 2012.
51. Kocher, P. C., *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*, CRYPTO 1996, LNCS 1109, pp. 104–113, Springer, 1996.
52. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E., *A cryptanalysis of PRINTcipher: The invariant subspace attack*, CRYPTO 2011, LNCS 6841, pp. 206–221, Springer, 2011.
53. Leander, G., Minaud, B., Rønjom, S., *A generic approach to invariant subspace attacks: Cryptanalysis of Robin, iSCREAM and Zorro*, EUROCRYPT 2015, LNCS 9056, pp. 254–283, Springer, 2015.
54. Li, Y., Wang, M., *Constructing S-boxes for Lightweight Cryptography with Feistel Structure*, CHES 2014, LNCS 8731, pp. 127–146, Springer, 2014.
55. Mangard, S., Oswald, E., Popp, T., *Power analysis attacks - revealing the secrets of smart cards*, Vol. 31. Springer Science & Business Media, 2008.
56. Matsui, M., *Linear Cryptoanalysis Method for DES Cipher* EUROCRYPT 1993, LNCS 765, pp. 386–397, Springer, 1994.
57. Matsui, M., *New Block Encryption Algorithm MISTY*, FSE 1997, LNCS 1267, pp. 54–68, Springer, 1997.
58. Matsui, M., *On Correlation Between the Order of S-boxes and the Strength of DES*, EUROCRYPT 1994, LNCS 950, pp. 366–375, Springer, 1995.
59. Peng, C., Du, X., Li, K., Li, M., *An Ultra-Lightweight Encryption Scheme in Underwater Acoustic Networks*, Journal of Sensors, 2016.
60. Samarati, P., Obaidat, M. S., Cabello, E., *Differential cryptanalysis with SAT solvers*, ICETE 2017, SciTePress, 2017.
61. Sarkar, S., Mandal, K., Saha, D., *On the Relationship Between Resilient Boolean Functions and Linear Branch Number of S-Boxes*, INDOCRYPT 2019, LNCS 11898, Springer, 2019.
62. Sasaki, Y., Aoki, K., *Finding preimages in full MD5 faster than exhaustive search*, EUROCRYPT 2009, LNCS 5479, Springer, 2009.
63. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T., *Piccolo: An Ultra-Lightweight Blockcipher*, CHES 2011, LNCS 6917, pp. 342–357, Springer, 2011.
64. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T., *The 128-Bit Blockcipher CLEFIA (Extended Abstract)*, FSE 2007, LNCS 4593, pp. 181–195, Springer, 2007.
65. Song, L., Huang, Z., Yang, Q., *Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA*, ACISP 2016, LNCS 9723, pp. 379–394, Springer, 2016.

66. Tezcan, C., Okan, G. O., Senol, A., Dogan, E., Yücebas, F., Baykal, N., *Differential Attacks on Lightweight Block Ciphers PRESENT, PRIDE, and RECTANGLE Revisited*, LightSec 2016, LNCS 10098, pp. 18–32, Springer, 2016.
67. Todo, Y., Leander, G., Sasaki, Y., *Nonlinear invariant attack - practical attack on full SCREAM, iSCREAM, and Midori64*, ASIACRYPT 2016, LNCS 10032, pp. 3–33, Springer, 2016.
68. Tolba, M., Abdelkhalek, A., Youssef, A. M., *Impossible Differential Cryptanalysis of Reduced-Round SKINNY*, AFRICACRYPT 2017, LNCS 10239, pp. 117–134, 2017.
69. Wagner, D., *The Boomerang Attack*, FSE 1999, LNCS 1636, pp. 156–170, Springer, 1999.
70. Wang, H., Peyrin, T., *Boomerang Switch in Multiple Rounds*, IACR Trans. Symmetric Cryptol. 2019(1), pp. 142–169, Springer, 2019.
71. Worthman, E., *ChaoLogix: Integrated Security*, Semiconductor Engineering, 13 April 2015.
72. Yan, L., Zhang, Y., Yang, L. T., Ning, H., *The Internet of things: from RFID to the next-generation pervasive networked systems*, Crc Press, 2008.
73. Z'aba, M., R., Raddum, H., Henricksen, M., Dawson, E., *Bit-Pattern Based Integral Attack*, FSE 2008, LNCS 5086, pp. 363–381, Springer, 2008.
74. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I., *RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms*, SCIENCE CHINA Information Sciences 58(12), pp. 1–15, 2015.

A Test Vectors

The following test vectors are represented in big endian representation.[4]

- PIPO-64/128
  - Secret key: `0x6DC416DD_779428D2_7E1D20AD_2E152297`
  - Plaintext: `0x098552F6_1E270026`
  - Ciphertext: `0x6B6B2981_AD5D0327`

- PIPO-64/256
  - Secret key:`0x009A3AA4_76A96DB5_54A71206_26D15633_6DC416DD _779428D2_7E1D20AD_2E152297`
  - Plaintext: `0x098552F6_1E270026`
  - Ciphertext: `0x816DAE6F_B6523889`

B Proofs of Propositions and Theorems

B.1 Proof of Proposition 1

($\Rightarrow$)

If S_3 or S_5^1 is non-bijective, there are two different inputs $X_L||X_R, X'_L||X'_R$ satisfying $(S_5^1(X_L), S_3(X_R)) = (S_5^1(X'_L), S_3(X'_R))$. Then, it is easy to see that

[4] The bitslice and table look-up implementation codes of PIPO can be found on GitHub (`https://github.com/PIPO-Blockcipher`).

$S_8(X_L||X_R) = S_8(X'_L||X'_R)$, and thus two conditions $i)$ and $ii)$ should hold. Assume that the f_y in condition $iii)$ is non-bijective for some $y \in \mathbb{F}_2^3$. Then there should be two different inputs a, a' satisfying $f_y(a) = f_y(a')$. It induces $\tau'_2(S_5^2(y||a)) = \tau'_2(S_5^2(y||a'))$. On the other hand, we can take a pair X_R, X'_R satisfying $\tau_3(S_5^2(y||a)) \oplus S_3(X_R) = \tau_3(S_5^2(y||a')) \oplus S_3(X'_R)$, and thus $C_R = C'_R$. Combining the above two equations yields $S_5^2(y||a) \oplus (S_3(X_R)||0^{(2)}) = S_5^2(y||a') \oplus (S_3(X'_R)||0^{(2)})$. And, we take a pair X_L, X'_L satisfying $S_5^1(X_L) = (y \oplus S_3(X_R))||a$ and $S_5^1(X'_L) = (y \oplus S_3(X'_R))||a'$. Since $a \neq a'$, we have $X_L \neq X'_L$ satisfying $S_8(X_L||X_R) = S_8(X'_L||X'_R)$. Therefore, condition $iii)$ should also hold.

$(\Leftarrow)$

Assume that $X_L \neq X'_L$ and $X_R = X'_R$. If $\tau_3(S_5^1(X_L)) \neq \tau_3(S_5^1(X'_L))$, then $C_L(X_L, X_R) \neq C_L(X'_L, X'_R)$. Let $\tau_3(S_5^1(X_L)) = \tau_3(S_5^1(X'_L))$. It leads to $C_L(X_L, X_R) = C_L(X'_L, X'_R)$, and $\tau'_2(S_5^1(X_L)) \neq \tau'_2(S_5^1(X'_L))$. Because of condition $iii)$, $\tau_2(C_R(X_L, X_R)) \neq \tau_2(C_R(X'_L, X'_R))$. Assume that $X_L = X'_L$ and $X_R \neq X'_R$. Since $S_3(X_R) \neq S_3(X'_R)$, $C_L(X_L, X_R) \neq C_L(X'_L, X'_R)$. Assume that $X_L \neq X'_L$, $X_R \neq X'_R$. If $C_L(X_L, X_R) = C_L(X'_L, X'_R)$, either $\tau'_2(S_5^1(X_L)) \neq \tau'_2(S_5^1(X'_L))$ or $\tau'_2(S_5^1(X_L)) = \tau'_2(S_5^1(X'_L))$. The former case leads to $\tau_2(C_R(X_L, X_R)) \neq \tau_2(C_R(X'_L, X'_R))$, and the latter case leads to $\tau'_3(C_R(X_L, X_R)) \neq \tau'_3(C_R(X'_L, X'_R))$. Therefore, the 8-bit S-box is bijective. ■

B.2 Proof of Proposition 2

All the smaller S-boxes in (C) and (D) should be bijective except for S_5^2 in (D). Condition $iii)$ of Proposition 1 should hold for S_5^2 in order to make the 8-bit S-box bijective. For a fixed $y \in \mathbb{F}_2^3$, the number of functions $S_5^2(y||\cdot)$ is $4! \times 8^4$. Since y can have any value in $\mathbb{F}_2^3$, the number of possible S_5^2 is $(4! \times 8^4)^8 = 98304^8$. ■

B.3 Proof of Theorem 1

As stated earlier, the expression of the C_L and C_R is

$$
\begin{aligned}
C_L(X_L, X_R) &= X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)), \\
C_R(X_L, X_R) &= X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))).
\end{aligned}
$$

We define the following notation for ease of expression.

$$Y = X_R \oplus S_4^1(X_L), \quad Z = X_L \oplus S_4^2(Y).$$

$\underline{(0^{(4)}||\Delta a, 0^{(4)}||\Delta c)}$: This case is ruled out by condition $i)$. It was proved in section 2.

$\underline{(0^{(4)}||\Delta a, \Delta d||0^{(4)})}$: It happens if and only if there exists at least one (X_L, X_R)

satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta 0$. The first equation is expressed as

$$\begin{aligned} &X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)) \oplus X_L \oplus S_4^2(X_R \oplus \Delta a \oplus S_4^1(X_L)) \\ &\quad = S_4^2(X_R \oplus S_4^1(X_L)) \oplus S_4^2(X_R \oplus \Delta a \oplus S_4^1(X_L)) = \Delta d \end{aligned}$$

By applying Y, we have

$$S_4^2(Y) \oplus S_4^2(Y \oplus \Delta a) = \Delta d \tag{3}$$

Similarly, the second equation $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta 0$ is expressed as

$$\begin{aligned} &X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \\ &\quad \oplus X_R \oplus \Delta a \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus \Delta a \oplus S_4^1(X_L))) \\ &= S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \\ &\quad \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L) \oplus \Delta a)) \oplus \Delta a = \Delta 0 \end{aligned}$$

By applying Eq. (3) and using the definition of Z, we obtain

$$S_4^3(Z) \oplus S_4^3(Z \oplus \Delta d) = \Delta a \tag{4}$$

Since the function $(X_L, X_R) \mapsto (Y, Z)$ is bijective, the $(0^{(4)}||\Delta a, \Delta d||0^{(4)})$ case does not happen if and only if there is no (Y, Z) satisfying both Eqs. ((3 and 4)), which is equivalent to condition $ii)$ where $\Delta\alpha = \Delta a$, $\Delta\beta = \Delta d$.

$\underline{(\Delta b||0^{(4)}, 0^{(4)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$. The first equation is expressed as

$$\begin{aligned} &X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)) \oplus X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b)) \\ &\quad = S_4^2(X_R \oplus S_4^1(X_L)) \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b)) \oplus \Delta b = \Delta 0. \end{aligned}$$

It becomes

$$S_4^2(X_R \oplus S_4^1(X_L)) \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b)) = \Delta b. \tag{5}$$

Similarly, the second equation $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$ is expressed as

$$\begin{aligned} &X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \\ &\quad \oplus X_R \oplus S_4^1(X_L \oplus \Delta b) \oplus S_4^3(X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b))) \\ &\quad = S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \\ &\qquad \oplus S_4^1(X_L \oplus \Delta b) \oplus S_4^3(X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b))) \\ &\quad = \Delta c. \end{aligned}$$

By applying Eq. (5), we get

$$S_4^1(X_L) \oplus S_4^1(X_L \oplus \Delta b) = \Delta c. \tag{6}$$

By applying Eq. (6) and using the definition of Y, Eq. (5) is rewritten as

$$S_4^2(Y) \oplus S_4^2(Y \oplus \Delta c) = \Delta b. \tag{7}$$

Since the function $(X_L, X_R) \mapsto (Y, X_R)$ is bijective, the $(\Delta b||0^{(4)}, 0^{(4)}||\Delta c)$ case does not happen if and only if there is no (Y, X_R) satisfying both Eqs. (6) and (7), which is equivalent to condition iii) where $\Delta\alpha = \Delta b$, $\Delta\beta = \Delta c$.

$\underline{(\Delta b||0^{(4)}, \Delta d||0^{(4)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta 0$. The second equation is expressed as

$$\begin{aligned} &X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \\ &\quad \oplus X_R \oplus S_4^1(X_L \oplus \Delta b) \oplus S_4^3(X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b))) \\ &\quad = S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \\ &\qquad \oplus S_4^1(X_L \oplus \Delta b) \oplus S_4^3(X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b))) \\ &\quad = \Delta 0. \end{aligned}$$

It becomes

$$\begin{aligned} &S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \oplus S_4^3(X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b))) \\ &\quad = S_4^1(X_L) \oplus S_4^1(X_L \oplus \Delta b). \end{aligned} \tag{8}$$

Similarly, the first equation $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ is expressed as

$$\begin{aligned} &X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)) \oplus X_L \oplus \Delta b \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b)) \\ &\quad = S_4^2(X_R \oplus S_4^1(X_L)) \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b)) \oplus \Delta b = \Delta d. \end{aligned}$$

It becomes

$$S_4^2(X_R \oplus S_4^1(X_L)) \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus \Delta b)) = \Delta b \oplus \Delta d. \tag{9}$$

Therefore, $(\Delta b||0^{(4)}, \Delta d||0^{(4)})$ case does not happen if and only if there is no (X_L, X_R) satisfying both Eqs. (8) and (9), which is equivalent to condition iv).■

B.4 Proof of Theorem 2

We use C_L, C_R, Y and Z defined in proof B.3.
$\underline{(0^{(4)}||\lambda_a, 0^{(4)}||\lambda_c)}$: This case is ruled out by condition i). It was proved in section 2.

$\underline{(0^{(4)}||\lambda_a, \lambda_d||0^{(4)})}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_R \bullet \lambda_a = C_L(X_L, X_R) \bullet \lambda_d$. The equation is expressed as

$$X_R \bullet \lambda_a = (X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \bullet \lambda_d.$$

It follows

$$(X_R \oplus S_4^1(X_L)) \bullet \lambda_a \oplus S_4^1(X_L) \bullet \lambda_a = (X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \bullet \lambda_d.$$

The equation becomes

$$X_L \bullet \lambda_d \oplus S_4^1(X_L) \bullet \lambda_a = Y \bullet \lambda_a \oplus S_4^2(Y) \bullet \lambda_d \tag{10}$$

by using the definition of Y. Note that the function $(X_L, X_R) \mapsto (X_L, Y)$ is bijective. The $(0^{(4)}||\lambda_a, \lambda_d||0^{(4)})$ case has zero bias if and only if the equation (10) is not biased, which is equivalent to condition *ii*) where $\lambda_\alpha = \lambda_d$, $\lambda_\beta = \lambda_a$.

$\underline{(\lambda_b||0^{(4)}, 0^{(4)}||\lambda_c)}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_L \bullet \lambda_b = C_R(X_L, X_R) \bullet \lambda_c$. The equation is expressed as

$$X_L \bullet \lambda_b = (X_R \oplus S_4^1(X_L) \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)))) \bullet \lambda_c.$$

It follows

$$\begin{aligned}(X_R \oplus S_4^1(X_L)) \bullet \lambda_c \oplus S_4^2(X_R \oplus S_4^1(X_L)) \bullet \lambda_b \\ = (X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \bullet \lambda_b \oplus S_4^3(X_L \oplus S_4^2(X_R \oplus S_4^1(X_L)))) \bullet \lambda_c.\end{aligned}$$

The equation becomes

$$Y \bullet \lambda_c \oplus S_4^2(Y) \bullet \lambda_b = Z \bullet \lambda_b \oplus S_4^3(Z) \bullet \lambda_c \tag{11}$$

by using the definition of Y and Z. Note that the function $(X_L, X_R) \mapsto (Z, Y)$ is bijective. The $(\lambda_b||0^{(4)}, 0^{(4)}||\lambda_c)$ case has zero bias if and only if the equation (11) is not biased, which is equivalent to condition *iii*) where $\lambda_\alpha = \lambda_c$, $\lambda_\beta = \lambda_b$.

$\underline{(\lambda_b||0^{(4)}, \lambda_d||0^{(4)})}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_L \bullet \lambda_b = C_L(X_L, X_R) \bullet \lambda_d$. The equation is expressed as

$$X_L \bullet \lambda_b = (X_L \oplus S_4^2(X_R \oplus S_4^1(X_L))) \bullet \lambda_d.$$

It follows

$$X_L \bullet (\lambda_b \oplus \lambda_c) = S_4^2(X_R \oplus S_4^1(X_L)) \bullet \lambda_d.$$

The equation becomes

$$X_L \bullet (\lambda_b \oplus \lambda_c) = S_4^2(Y) \bullet \lambda_d \tag{12}$$

by using the definition of Y. Since the left side of the equation is always not biased, only need to consider the right side. The Eq. (12) is not biased if and only if

$$0 = S_4^2(Y) \bullet \lambda_d \tag{13}$$

is not biased. The $(\lambda_b||0^{(4)}, \lambda_d||0^{(4)})$ case has zero bias if and only if the equation (13) is not biased, which is equivalent to condition *iv*) where $\lambda_\alpha = \lambda_d$. ■

B.5 Proof of Theorem 3

The expression of the C_L and C_R is

$$C_L(X_L, X_R) = S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)),$$
$$C_R(X_L, X_R) = S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)).$$

We define the following notation for ease of expression.

$$Y = X_L \oplus X_R, \quad Z = X_L \oplus S_4^1(X_L \oplus X_R), \quad W = X_R \oplus S_4^1(X_L \oplus X_R).$$

$\underline{(0^{(4)}||\Delta a, 0^{(4)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta c$. The first equation is expressed as

$$S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^2(X_L \oplus S_4^1(X_L \oplus X_R \oplus \Delta a)) = \Delta 0.$$

By applying $(S_4^2)^{-1}$ and using the definition of Y, we obtain

$$S_4^1(Y) \oplus S_4^1(Y \oplus \Delta a) = \Delta 0. \tag{14}$$

Similarly, the second equation $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta c$ is expressed as

$$S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^3(X_R \oplus \Delta a \oplus S_4^1(X_L \oplus X_R \oplus \Delta a)) = \Delta c.$$

By applying Eq. (14) and using the definition of W, we obtain

$$S_4^3(W) \oplus S_4^3(W \oplus \Delta a) = \Delta c. \tag{15}$$

Since the function $(X_L, X_R) \mapsto (Y, W)$ is bijective, the $(0^{(4)}||\Delta a, 0^{(4)}||\Delta c)$ case does not happen if and only if there is no (Y, W) satisfying both Eqs. (14) and (15), which is equivalent to condition $i)$ where $\Delta\alpha = \Delta a$, $\Delta\beta = \Delta c$.

$\underline{(0^{(4)}||\Delta a, \Delta d||0^{(4)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta 0$. The second equation is expressed as

$$S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^3(X_R \oplus \Delta a \oplus S_4^1(X_L \oplus X_R \oplus \Delta a)) = \Delta 0.$$

By applying $(S_4^3)^{-1}$ and using the definition of Y, we obtain

$$S_4^1(Y) \oplus S_4^1(Y \oplus \Delta a) = \Delta a. \tag{16}$$

Similarly, the first equation $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ is expressed as

$$S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^2(X_L \oplus S_4^1(X_L \oplus X_R \oplus \Delta a)) = \Delta d.$$

By applying Eq. (16) and using the definition of Z, we obtain

$$S_4^2(Z) \oplus S_4^2(Z \oplus \Delta a) = \Delta d. \tag{17}$$

Since the function $(X_L, X_R) \mapsto (Z, Y)$ is bijective, the $(0^{(4)}||\Delta a, \Delta d||0^{(4)})$ case does not happen if and only if there is no (Z, Y) satisfying both Eqs. (16) and (17), which is equivalent to condition $ii)$ where $\Delta\alpha = \Delta a$, $\Delta\beta = \Delta d$.

$\underline{(\Delta b||0^{(4)}, 0^{(4)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$. The first equation is expressed as

$$S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^2(X_L \oplus \Delta b \oplus S_4^1(X_L \oplus \Delta b \oplus X_R)) = \Delta 0.$$

By applying $(S_4^2)^{-1}$ and using the definition of Y, we obtain

$$S_4^1(Y) \oplus S_4^1(Y \oplus \Delta b) = \Delta b. \tag{18}$$

Similarly, the second equation $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$ is expressed as

$$S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^3(X_R \oplus S_4^1(X_L \oplus \Delta b \oplus X_R)) = \Delta c.$$

By applying Eq. (18) and using the definition of W, we obtain

$$S_4^3(W) \oplus S_4^3(W \oplus \Delta b) = \Delta c. \tag{19}$$

Since the function $(X_L, X_R) \mapsto (Y, W)$ is bijective, the $(\Delta b||0^{(4)}, 0^{(4)}||\Delta c)$ case does not happen if and only if there is no (Y, W) satisfying both Eqs. (18) and (19), which is equivalent to condition $iii)$ where $\Delta\alpha = \Delta b$, $\Delta\beta = \Delta c$.

$\underline{(\Delta b||0^{(4)}, \Delta d||0^{(4)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta 0$. The second equation is expressed as

$$S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^3(X_R \oplus S_4^1(X_L \oplus X_R \oplus \Delta b)) = \Delta 0.$$

By applying $(S_4^3)^{-1}$ and using the definition of Y, we obtain

$$S_4^1(Y) \oplus S_4^1(Y \oplus \Delta b) = \Delta 0. \tag{20}$$

Similarly, the first equation $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ is expressed as

$$S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \oplus S_4^2(X_L \oplus \Delta b \oplus S_4^1(X_L \oplus \Delta b \oplus X_R)) = \Delta d.$$

By applying Eq. (20) and using the definition of Z, we obtain

$$S_4^2(Z) \oplus S_4^2(Z \oplus \Delta b) = \Delta d. \tag{21}$$

Since the function $(X_L, X_R) \mapsto (Z, Y)$ is bijective, the $(\Delta b||0^{(4)}, \Delta d||0^{(4)})$ case does not happen if and only if there is no (Z, Y) satisfying both Eqs. (20) and (21), which is equivalent to condition $iv)$ where $\Delta\alpha = \Delta b$, $\Delta\beta = \Delta d$. ■

B.6 Proof of Theorem 4

We use C_L, C_R, Y and Z defined in proof B.5.

$(0^{(4)}||\lambda_a, 0^{(4)}||\lambda_c)$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_R \bullet \lambda_a = C_R(X_L, X_R) \bullet \lambda_c$. The equation is expressed as

$$X_R \bullet \lambda_a = S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_c.$$

It follows

$$S_4^1(X_L \oplus X_R) \bullet \lambda_a = (X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_a \oplus S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_c.$$

The equation becomes

$$S_4^1(Y) \bullet \lambda_a = W \bullet \lambda_a \oplus S_4^3(W) \bullet \lambda_c \tag{22}$$

by using the definition of Y and W. Note that the function $(X_L, X_R) \mapsto (Y, W)$ is bijective. The $(0^{(4)}||\lambda_a, 0^{(4)}||\lambda_c)$ case has zero bias if and only if the equation (22) is not biased, which is equivalent to condition $i)$ where $\lambda_\alpha = \lambda_a$, $\lambda_\beta = \lambda_c$.

$(0^{(4)}||\lambda_a, \lambda_d||0^{(4)})$:Its bias can be calculated by the number of (X_L, X_R) satisfying $X_R \bullet \lambda_a = C_L(X_L, X_R) \bullet \lambda_d$. The equation is expressed as

$$X_R \bullet \lambda_a = S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_d.$$

It follows

$$\begin{aligned}&(X_L \oplus X_R) \bullet \lambda_a \oplus S_4^1(X_L \oplus X_R) \bullet \lambda_a \\ &\quad = (X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_a \oplus S_4^2(X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_d.\end{aligned}$$

The equation becomes

$$Y \bullet \lambda_a \oplus S_4^1(Y) \bullet \lambda_a = W \bullet \lambda_a \oplus S_4^2(W) \bullet \lambda_d \tag{23}$$

by using the definition of Y and W. Note that the function $(X_L, X_R) \mapsto (Y, W)$ is bijective. The $(0^{(4)}||\lambda_a, \lambda_d||0^{(4)})$ case has zero bias if and only if the equation (23) is not biased, which is equivalent to condition $ii)$ where $\lambda_\alpha = \lambda_a$, $\lambda_\beta = \lambda_d$.

$(\lambda_b||0^{(4)}, 0^{(4)}||\lambda_c)$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_L \bullet \lambda_b = C_R(X_L, X_R) \bullet \lambda_c$. The equation is expressed as

$$X_L \bullet \lambda_b = S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_c.$$

It follows

$$\begin{aligned}&(X_L \oplus X_R) \bullet \lambda_b \oplus S_4^1(X_L \oplus X_R) \bullet \lambda_b \\ &\quad = (X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_b \oplus S_4^3(X_R \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_c.\end{aligned}$$

The equation becomes

$$Y \bullet \lambda_b \oplus S_4^1(Y) \bullet \lambda_b = W \bullet \lambda_b \oplus S_4^3(W) \bullet \lambda_c \tag{24}$$

by using the definition of Y and W. Note that the function $(X_L, X_R) \mapsto (Y, W)$ is bijective. The $(\lambda_b||0^{(4)}, 0^{(4)}||\lambda_c)$ case has zero bias if and only if the equation (24) is not biased, which is equivalent to condition $iii)$ where $\lambda_\alpha = \lambda_b$, $\lambda_\beta = \lambda_c$.

$\underline{(\lambda_b||0^{(4)}, \lambda_d||0^{(4)})}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_L \bullet \lambda_b = C_L(X_L, X_R) \bullet \lambda_d$. The equation is expressed as

$$X_L \bullet \lambda_b = S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_d.$$

It follows

$$S_4^1(X_L \oplus X_R) \bullet \lambda_b = (X_L \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_b \oplus S_4^2(X_L \oplus S_4^1(X_L \oplus X_R)) \bullet \lambda_d.$$

The equation becomes

$$S_4^1(Y) \bullet \lambda_b = Z \bullet \lambda_b \oplus S_4^3(Z) \bullet \lambda_d \tag{25}$$

by using the definition of Y and Z. Note that the function $(X_L, X_R) \mapsto (Z, Y)$ is bijective. The $(\lambda_b||0^{(4)}, \lambda_d||0^{(4)})$ case has zero bias if and only if the equation (25) is not biased, which is equivalent to condition $iv)$ where $\lambda_\alpha = \lambda_b$, $\lambda_\beta = \lambda_d$.$\blacksquare$

B.7 Proof of Theorem 5

The expression of the C_L and C_R is

$$C_L(X_L, X_R) = S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}),$$
$$C_R(X_L, X_R) = \tau_3(S_5^1(X_L)) \oplus X_R \oplus S_3(X_R).$$

We define the following notation for ease of expression.

$$Y = S_5^1(X_L), \quad Z = S_5^1(X_L) \oplus X_R||0^{(2)},$$
$$A = \tau_2'(Y) = \tau_2'(Z), \quad Y = Y'||A, \quad Z = Z'||A.$$

$\underline{(0^{(5)}||\Delta a, 0^{(5)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta c$. The first equation is expressed as

$$S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}) \oplus S_5^2(S_5^1(X_L) \oplus (X_R \oplus \Delta a)||0^{(2)}) = \Delta 0.$$

By applying $(S_5^2)^{-1}$, we obtain

$$\Delta a||0^{(2)} = \Delta 0.$$

Since the equation is impossible, the $(0^{(5)}||\Delta a, 0^{(5)}||\Delta c)$ case dose not happen.

$\underline{(0^{(5)}||\Delta a, \Delta d||0^{(3)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta 0$. The second equation is expressed as

$$\tau_3(S_5^1(X_L)) \oplus X_R \oplus S_3(X_R) \oplus \tau_3(S_5^1(X_L)) \oplus X_R \oplus \Delta a \oplus S_3(X_R \oplus \Delta a) = \Delta 0.$$

Clearly,

$$S_3(X_R) \oplus S_3(X_R \oplus \Delta a) = \Delta a. \tag{26}$$

Similarly, the first equation $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ is expressed as

$$S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}) \oplus S_5^2(S_5^1(X_L) \oplus (X_R \oplus \Delta a)||0^{(2)}) = \Delta d.$$

By using the definition of Z, we obtain

$$S_5^2(Z) \oplus S_5^2(Z \oplus \Delta a||0^{(2)}) = \Delta d. \tag{27}$$

Since the function $(X_L, X_R) \mapsto (Z, X_R)$ is bijective, the $(0^{(5)}||\Delta a, \Delta d||0^{(3)})$ case does not happen if and only if there is no (Z, X_R) satisfying both Eqs. (26) and (27), which is equivalent to condition $i)$ where $\Delta\alpha = \Delta a$, $\Delta\beta = \Delta d$.

$\underline{(\Delta b||0^{(3)}, 0^{(5)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$. The second equation is expressed as

$$\tau_3(S_5^1(X_L)) \oplus X_R \oplus S_3(X_R) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) \oplus X_R \oplus S_3(X_R) = \Delta c.$$

Clearly,

$$\tau_3(S_5^1(X_L)) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) = \Delta c. \tag{28}$$

Similarly, the first equation $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ is expressed as

$$S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}) \oplus S_5^2(S_5^1(X_L \oplus \Delta b) \oplus X_R||0^{(2)}) = \Delta 0.$$

By applying $(S_5^2)^{-1}$, we obtain

$$S_5^1(X_L) \oplus S_5^1(X_L \oplus \Delta b) = \Delta 0. \tag{29}$$

Since Eqs. (28) and (29) cause contradiction, the $(\Delta b||0^{(3)}, 0^{(5)}||\Delta c)$ case dose not happen.

$\underline{(\Delta b||0^{(3)}, \Delta d||0^{(3)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta 0$. The second equation is expressed as

$$\tau_3(S_5^1(X_L)) \oplus X_R \oplus S_3(X_R) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) \oplus X_R \oplus S_3(X_R) = \Delta 0.$$

Clearly,
$$\tau_3(S_5^1(X_L)) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) = \Delta 0.$$
Since S_5^1 is bijection, for a non-zero difference $\Delta\omega \in \mathbb{F}_2^2$, the above equation becomes
$$S_5^1(X_L) \oplus S_5^1(X_L \oplus \Delta b) = \Delta\omega. \tag{30}$$
By applying $(S_5^1)^{-1}$, we get
$$X_L \oplus \Delta b = (S_5^1)^{-1}(S_5^1(X_L) \oplus \Delta\omega).$$
By using the definition of Y, we obtain
$$(S_5^1)^{-1}(Y) \oplus (S_5^1)^{-1}(Y \oplus \Delta\omega) = \Delta b. \tag{31}$$
Similarly, the first equation $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ is expressed as
$$S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}) \oplus S_5^2(S_5^1(X_L \oplus \Delta b) \oplus X_R||0^{(2)}) = \Delta d.$$
By applying Eq. (30) and using the definition of Y, we obtain
$$S_5^2(Y) \oplus S_5^2(Y \oplus \Delta\omega) = \Delta d. \tag{32}$$
For each A, the Eqs. (31) and (32) are equivalent to
$$\mathfrak{F}_A^2(Y') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^2(Y') = \Delta b, \tag{33}$$
$$\mathfrak{F}_A^1(Z') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^1(Z') = \Delta d. \tag{34}$$
Here, $\Delta\omega$ is arbitrary nonzero 2-bit difference, and thus we can define $B = A \oplus \Delta\omega$ *i.e.*, $B \neq A$. Since the function $(X_L, X_R) \mapsto (Y', A, Z')$ is bijective, the $(\Delta b||0^{(3)}, \Delta d||0^{(3)})$ case does not happen if and only if there is no (Y', A, Z') satisfying both Eqs. (33) and (34) for all $B(\neq A)$, which is equivalent to condition $ii)$ where $\Delta\alpha = \Delta b$, $\Delta\beta = \Delta d$. ■

B.8 Proof of Theorem 6

We use C_L, C_R, Y and Z defined in Appendix B.7.

$\underline{(0^{(5)}||\lambda_a, 0^{(5)}||\lambda_c)}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_R \bullet \lambda_a = C_R(X_L, X_R) \bullet \lambda_c$. The equation is expressed as
$$X_R \bullet \lambda_a = (\tau_3(S_5^1(X_L)) \oplus X_R \oplus S_3(X_R)) \bullet \lambda_c.$$
It follows
$$X_R \bullet (\lambda_a \oplus \lambda_c) \oplus S_3(X_R) \bullet \lambda_c = \tau_3(S_5^1(X_L)) \bullet \lambda_c.$$
Clearly,
$$X_R \bullet (\lambda_a \oplus \lambda_c) \oplus S_3(X_R) \bullet \lambda_c = S_5^1(X_L) \bullet \lambda_c||0^{(2)}.$$

Since S_5^1 is bijective, the $(0^{(5)}||\lambda_a, 0^{(5)}||\lambda_c)$ case has zero bias.

$\underline{(0^{(5)}||\lambda_a, \lambda_d||0^{(3)})}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_R \bullet \lambda_a = C_L(X_L, X_R) \bullet \lambda_d$. The equation is expressed as

$$X_R \bullet \lambda_a = S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}) \bullet \lambda_c.$$

The equation becomes

$$X_R \bullet \lambda_a = S_5^2(Z) \bullet \lambda_c$$

by using the definition of Z. Since left side is not biased, the $(0^{(5)}||\lambda_a, \lambda_d||0^{(3)})$ case has zero bias.

$\underline{(\lambda_b||0^{(3)}, 0^{(5)}||\lambda_c)}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_L \bullet \lambda_b = C_R(X_L, X_R) \bullet \lambda_c$. The equation is expressed as

$$X_L \bullet \lambda_b = (\tau_3(S_5^1(X_L)) \oplus X_R \oplus S_3(X_R)) \bullet \lambda_c.$$

It follows

$$X_R \bullet \lambda_c \oplus S_3(X_R) \bullet \lambda_c = X_L \bullet \lambda_b \oplus \tau_3(S_5^1(X_L)) \bullet \lambda_c.$$

Clearly,

$$X_R \bullet \lambda_c \oplus S_3(X_R) \bullet \lambda_c = X_L \bullet \lambda_b \oplus S_5^1(X_L) \bullet \lambda_c||0^{(2)}. \tag{35}$$

The $(\lambda_b||0^{(3)}, 0^{(5)}||\lambda_c)$ case has zero bias if and only if the equation (35) is not biased, which is equivalent to condition $i)$ where $\lambda_\alpha = \lambda_b$, $\lambda_\beta = \lambda_c$.

$\underline{(\lambda_b||0^{(3)}, \lambda_d||0^{(3)})}$: Its bias can be calculated by the number of (X_L, X_R) satisfying $X_L \bullet \lambda_b = C_L(X_L, X_R) \bullet \lambda_d$. The equation is expressed as

$$X_L \bullet \lambda_b = S_5^2(S_5^1(X_L) \oplus X_R||0^{(2)}) \bullet \lambda_d.$$

The equation becomes

$$(S_5^1)^{-1}(Y) \bullet \lambda_b = S_5^2(Z) \bullet \lambda_d$$

by using the definition of Y and Z. For definition of A, the above equation is equivalent to

$$f_A^1(Y') \bullet \lambda_b = f_A^2(Z') \bullet \lambda_d. \tag{36}$$

The $(\lambda_b||0^{(3)}, \lambda_d||0^{(3)})$ case has zero bias if and only if the equation (36) is not biased, which is equivalent to condition $ii)$ where $\lambda_\alpha = \lambda_b$, $\lambda_\beta = \lambda_d$. ■

B.9 Proof of Theorem 7

We define the following notation for ease of expression.

$Y = S_5^1(X_L)$, $Z = S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)})$, $A = \tau_2'(Y) = \tau_2'(Z)$, $Y = Y'||A$, $Z = Z'||A$.

Then, the expression of the C_L and C_R is

$$C_L(X_L, X_R) = \tau_3(Y) \oplus S_3(X_R) = \tau_3(Z),$$
$$C_R(X_L, X_R) = \rho_c(S_5^2(Y \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R) = \rho_c(Z) \oplus S_3(X_R).$$

For convenience, we do not write 0 paddings on MSBs of smaller-bit data operating with larger-bit data; here, the 5-bit operand $S_3(X_R)$ represents $0^{(2)}||S_3(X_R)$.

$\underline{(0^{(5)}||\Delta a, 0^{(3)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta c$. The first equation is expressed as

$$\tau_3(Y) \oplus S_3(X_R) \oplus \tau_3(Y) \oplus S_3(X_R \oplus \Delta a) = S_3(X_R) \oplus S_3(X_R \oplus \Delta a) = \Delta 0.$$

Since S_3 is bijective, the $(0^{(5)}||\Delta a, 0^{(3)}||\Delta c)$ case dose not happen.

$\underline{(0^{(5)}||\Delta a, \Delta d||0^{(5)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L, X_R \oplus \Delta a) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta 0$. The first equation is expressed as

$$\tau_3(Y) \oplus S_3(X_R) \oplus \tau_3(Y) \oplus S_3(X_R \oplus \Delta a) = S_3(X_R) \oplus S_3(X_R \oplus \Delta a) = \Delta d. \quad (37)$$

Similarly, the second equation $C_R(X_L, X_R) \oplus C_R(X_L, X_R \oplus \Delta a) = \Delta 0$ is expressed as

$$\begin{aligned} &\rho_c(S_5^2(Y \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R) \\ &\quad \oplus \rho_c(S_5^2(Y \oplus (S_3(X_R \oplus \Delta a)||0^{(2)}))) \oplus S_3(X_R \oplus \Delta a) \\ &\quad = \rho_c(S_5^2(Y \oplus (S_3(X_R)||0^{(2)}))) \oplus \rho_c(S_5^2(Y \oplus ((S_3(X_R) \oplus \Delta d)||0^{(2)}))) \oplus \Delta d = \Delta 0. \end{aligned}$$

By applying ρ_c^{-1}, we have

$$S_5^2(Y \oplus (S_3(X_R)||0^{(2)})) \oplus S_5^2(Y \oplus ((S_3(X_R) \oplus \Delta d)||0^{(2)})) = \Delta d||0^{(2)}.$$

By applying Z, we obtain

$$S_5^2(Z) \oplus S_5^2(Z \oplus (\Delta d||0^{(2)})) = \Delta d||0^{(2)}. \quad (38)$$

Since the function $(X_L, X_R) \mapsto (Z, X_R)$ is bijective, the $(0^{(5)}||\Delta a, \Delta d||0^{(5)})$ case does not happen if and only if there is no (Z, X_R) satisfying both Eqs. (37) and (38), which is equivalent to condition $i)$ where $\Delta\alpha = \Delta a$, $\Delta\beta = \Delta d$.

$\underline{(\Delta b||0^{(3)}, 0^{(3)}||\Delta c)}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta 0$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$. The first equation is expressed as

$$\tau_3(S_5^1(X_L)) \oplus S_3(X_R) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) \oplus S_3(X_R) = \tau_3(S_5^1(X_L)) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) = \Delta 0.$$

Since S_5^1 is bijective, for a non-zero difference $\Delta\omega \in \mathbb{F}_2^2$, the above equation becomes

$$S_5^1(X_L) \oplus S_5^1(X_L \oplus \Delta b) = \Delta\omega.$$

The equation is rewritten as

$$S_5^1(X_L \oplus \Delta b) = S_5^1(X_L) \oplus \Delta\omega.$$

By applying $(S_5^1)^{-1}$, we obtain

$$X_L \oplus \Delta b = (S_5^1)^{-1}(S_5^1(X_L) \oplus \Delta\omega).$$

By using the variables Y, Y' and A, we have

$$(S_5^1)^{-1}(Y) \oplus (S_5^1)^{-1}(Y \oplus \Delta\omega) = \Delta b,$$

$$(S_5^1)^{-1}(Y'||A) \oplus (S_5^1)^{-1}(Y'||(A \oplus \Delta\omega)) = \Delta b. \tag{39}$$

And the second equation $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta c$ is expressed as

$$\begin{aligned}
&\rho_c(S_5^2(S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R) \\
&\quad \oplus \rho_c(S_5^2(S_5^1(X_L \oplus \Delta b) \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R) \\
&\quad = \rho_c(S_5^2(Z)) \oplus \rho_c(S_5^2(Z \oplus \Delta\omega)) = \Delta c.
\end{aligned}$$

By applying ρ_c^{-1}, we obtain

$$S_5^2(Z) \oplus S_5^2(Z \oplus \Delta\omega) = \rho_c^{-1}(\Delta c).$$

This gives the equation

$$S_5^2(Z'||A) \oplus S_5^2(Z'||(A \oplus \Delta\omega)) = \rho_c^{-1}(\Delta c). \tag{40}$$

For each A, the above Eqs. (39) and (40) are equivalent to

$$\mathfrak{F}_A^1(Y') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^1(Y') = \Delta b, \tag{41}$$

$$\mathfrak{F}_A^2(Z') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^2(Z') = \rho_c^{-1}(\Delta c). \tag{42}$$

Here, $\Delta\omega$ is arbitrary nonzero 2-bit difference, and thus we can define $B = A \oplus \Delta\omega$ *i.e.*, $B \neq A$. Since the function $(X_L, X_R) \mapsto (Y', A, Z')$ is bijective, the $(\Delta b||0^{(3)}, 0^{(3)}||\Delta c)$ case does not happen if and only if there is no (Y', A, Z') satisfying both Eqs. (41) and (42) for all $B(\neq A)$, which is equivalent to condition *ii*) where $\Delta\alpha = \Delta b$, $\Delta\beta = \rho_c^{-1}(\Delta c)$.

$\underline{(\Delta b||0^{(3)}, \Delta d||0^{(5)})}$: It happens if and only if there exists at least one (X_L, X_R) satisfying both $C_L(X_L, X_R) \oplus C_L(X_L \oplus \Delta b, X_R) = \Delta d$ and $C_R(X_L, X_R) \oplus C_R(X_L \oplus \Delta b, X_R) = \Delta 0$. The first equation is expressed as

$$\tau_3(S_5^1(X_L)) \oplus S_3(X_R) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) \oplus S_3(X_R) = \tau_3(S_5^1(X_L)) \oplus \tau_3(S_5^1(X_L \oplus \Delta b)) = \Delta d.$$

For a difference $\varDelta\omega \in \mathbb{F}_2^2$, the above equation becomes

$$S_5^1(X_L) \oplus S_5^1(X_L \oplus \varDelta b) = \varDelta d || \varDelta\omega.$$

As in Eq. (39), we obtain

$$(S_5^1)^{-1}(Y'||A) \oplus (S_5^1)^{-1}((Y' \oplus \varDelta d)||(A \oplus \varDelta\omega)) = \varDelta b. \tag{43}$$

And the second equation is expressed as

$$\begin{aligned}\rho_c(S_5^2(S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R)& \\ \oplus\, \rho_c(S_5^2(S_5^1(X_L \oplus \varDelta b) \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R)& \\ = \rho_c(S_5^2(Z)) \oplus \rho_c(S_5^2(Z \oplus (\varDelta d||\varDelta\omega))) = \varDelta 0.&\end{aligned}$$

Clearly,

$$S_5^2(Z) \oplus S_5^2(Z \oplus (\varDelta d||\varDelta\omega)) = \varDelta 0.$$

It becomes

$$S_5^2(Z'||A) \oplus S_5^2((Z' \oplus \varDelta d)||(A \oplus \varDelta\omega)) = \varDelta 0. \tag{44}$$

For each A, the above Eqs. (43) and (44) are equivalent to

$$\mathfrak{F}_A^1(Y') \oplus \mathfrak{F}_{A\oplus\varDelta\omega}^1(Y' \oplus \varDelta d) = \varDelta b, \tag{45}$$

$$\mathfrak{F}_A^2(Z') \oplus \mathfrak{F}_{A\oplus\varDelta\omega}^2(Z' \oplus \varDelta d) = \varDelta 0. \tag{46}$$

Similarly to the case above, we define $B = A \oplus \varDelta\omega$. In this time, B can be either A or not, since $\varDelta\omega$ can be a zero difference. The $(\varDelta b||0^{(3)}, \varDelta d||0^{(5)})$ case does not happen if and only if there is no (Y', A, Z') satisfying both Eqs. (45) and (46) for all B, which is equivalent to condition $iii)$ where $\varDelta\alpha = \varDelta d$, $\varDelta\beta = \varDelta b$. ■

B.10 Proof of Theorem 8

We use C_L, C_R, Y, Y', Z, Z', and A defined in proof B.9.

$\underline{(0^{(5)}||\lambda_a, 0^{(3)}||\lambda_c)}$: This case is expressed as $X_R \bullet \lambda_a = C_R(X_L, X_R) \bullet \lambda_c$. It follows $X_R \bullet \lambda_a = (\rho_c(S_5^2(S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R)) \bullet \lambda_c$. By applying the variable Z, the equation becomes $X_R \bullet \lambda_a \oplus S_3(X_R) \bullet \lambda_c = \rho_c(S_5^2(Z)) \bullet \lambda_c$. Note that the function $(X_L, X_R) \mapsto (Z, X_R)$ is bijective. Suppose $\tau_2(\lambda_c) \neq 0$. Then, the equation becomes $X_R \bullet \lambda_a = \rho_c(S_5^2(Z)) \bullet \lambda_c$. This should have zero bias because the equation $X_R \bullet \lambda_a = 0$ has zero bias, and Z and X_R are independent variables. Now, suppose $\tau_2(\lambda_c) = 0$. The equation $X_R \bullet \lambda_a \oplus S_3(X_R) \bullet \lambda_c = \rho_c(S_5^2(Z)) \bullet \lambda_c$ has zero bias if and only if at least one of the entries $(\lambda_a, \tau_3'(\lambda_c))$ in LAT of S_3 and $(0, \tau_3'(\lambda_c)||0^{(2)})$ in LAT of S_5^2 is zero. This is due to the fact that Z is independent of X_R. It is equivalent to condition $i)$

$\underline{(0^{(5)}||\lambda_a, \lambda_d||0^{(5)})}$: This case is expressed as $X_R \bullet \lambda_a = C_L(X_L, X_R) \bullet \lambda_d$. It follows

$X_R \bullet \lambda_a = (\tau_3(S_5^1(X_L)) \oplus S_3(X_R)) \bullet \lambda_d$. The equation becomes $X_R \bullet \lambda_a = \tau_3(Z) \bullet \lambda_d$ by using the definition of Z. So, this case has zero bias, because $\tau_3(Z)$ is independent of X_R.

$(\lambda_b||0^{(3)}, 0^{(3)}||\lambda_c)$: This case is expressed as $X_L \bullet \lambda_b = C_R(X_L, X_R) \bullet \lambda_c$. It follows $X_L \bullet \lambda_b = (\rho_c(S_5^2(S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)}))) \oplus S_3(X_R)) \bullet \lambda_c$. We can replace the equation to

$$\begin{aligned} &X_L \bullet \lambda_b \oplus S_5^1(X_L) \bullet \lambda_t \\ &\quad = (S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)})) \bullet \lambda_t \oplus \rho_c(S_5^2(S_5^1(X_L) \oplus (S_3(X_R)||0^{(2)}))) \bullet \lambda_c, \end{aligned}$$

where $\lambda_t = \tau_3'(\lambda_c)||0^{(2)}$ (here, $0^{(2)}$ can be replaced by $01, 10$ or $1^{(2)}$). By applying the variables of Y and Z, this becomes equivalent to the following equations

$$(S_5^1)^{-1}(Y) \bullet \lambda_b \oplus Y \bullet \lambda_t = Z \bullet \lambda_t \oplus (\rho_c(S_5^2(Z))) \bullet \lambda_c,$$

$$(S_5^1)^{-1}(Y'||A) \bullet \lambda_b \oplus (Y'||A) \bullet \lambda_t = (Z'||A) \bullet \lambda_t \oplus (\rho_c(S_5^2(Z'||A))) \bullet \lambda_c.$$

For all $A \in \mathbb{F}_2^2$, we have

$$\mathfrak{F}_A^1(Y') \bullet \lambda_b \oplus (Y'||A) \bullet \lambda_t = (Z'||A) \bullet \lambda_t \oplus (\rho_c(\mathfrak{F}_A^2(Z'))) \bullet \lambda_c.$$

Clearly,

$$\mathfrak{F}_A^1(Y') \bullet \lambda_b \oplus Y' \bullet \tau_3(\lambda_t) = Z' \bullet \tau_3(\lambda_t) \oplus (\rho_c(\mathfrak{F}_A^2(Z'))) \bullet \lambda_c.$$

A collection of (Y', Z') that satisfies the above equation is equivalent to

$$\begin{aligned} &\{Y'|0 = \mathfrak{F}_A^1(Y') \bullet \lambda_b \oplus Y' \bullet \tau_3(\lambda_t)\} \times \{Z'|0 = Z' \bullet \tau_3(\lambda_t) \oplus (\rho_c(\mathfrak{F}_A^2(Z'))) \bullet \lambda_c\} \\ &\quad \cup \{Y'|1 = \mathfrak{F}_A^1(Y') \bullet \lambda_b \oplus Y' \bullet \tau_3(\lambda_t)\} \times \{Z'|1 = Z' \bullet \tau_3(\lambda_t) \oplus (\rho_c(\mathfrak{F}_A^2(Z'))) \bullet \lambda_c\} \end{aligned}$$

Then the number of the above set is $(4 + a_A)(4 + b_A) + (4 - a_A)(4 - b_A) = 32 + 2a_A b_A$, where a_A and b_A are the entries of $(\tau_3(\lambda_t), \lambda_b)$ and $(\tau_3(\lambda_t), \rho_c^{-1}(\lambda_c))$ in LAT of $\mathfrak{F}_A^1$ and $\mathfrak{F}_A^2$, respectively. The above equation has zero bias if and only if

$$\sum_{A \in \mathbb{F}_2^2} (32 + 2a_A b_A) = 2(\sum_{A \in \mathbb{F}_2^2} a_A b_A) + 128 = 128$$

It leads to $\sum_{A \in \mathbb{F}_2^2} a_A b_A = 0$. Because $\tau_3(\lambda_t) = \tau_3'(\lambda_c)$, it is equivalent to condition ii) (when $\tau_3'(\lambda_c) \neq 0$) and condition iii) (when $\tau_3'(\lambda_c) = 0$).

$(\lambda_b||0^{(3)}, \lambda_d||0^{(5)})$: This case is expressed as $X_L \bullet \lambda_b = C_L(X_L, X_R) \bullet \lambda_d$. It follows $X_L \bullet \lambda_b = (\tau_3(S_5^1(X_L)) \oplus S_3(X_R)) \bullet \lambda_d$. The equation becomes $X_L \bullet \lambda_b = Z' \bullet \lambda_d$ by using the definition of Z'. We note that the function $(X_L, X_R) \mapsto (X_L, Z')$ is bijective, and X_L and Z' are independent variables. So, this equation has zero bias. ■

C Bitsliced Implementations of New S-Boxes

Listing 1.2 is the bitsliced implementation of the S_8.[5] The bitsliced implementation of the inverse S_8 cannot be obtained by reversing the bitsliced implementation of the S_8 because the input bits of S_5^2 are not all given. The Listing 1.3 shows how to implement the inverse S_8 with the given input bits. Since the S_8 applies each column of 8×8 array of bits depicted in Fig. 2, we can implement the S-layer by replacing bit $x[i]$ with byte $X[i]$ which represents the i-th row value, where $i = 0, 1, 2, \cdots, 7$. Listings 1.4∼1.9 represent bitsliced implementations of other new S-boxes.

Listing 1.2. The bitsliced implementation of the S_8 (in C code)

```
//(MSb: x[7], LSb: x[0]) :"b" represents bit
// Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
// S5_1
x[5] ^= (x[7] & x[6]);
x[4] ^= (x[3] & x[5]);
x[7] ^= x[4];
x[6] ^= x[3];
x[3] ^= (x[4] | x[5]);
x[5] ^= x[7];
x[4] ^= (x[5] & x[6]);
// S3
x[2] ^= x[1] & x[0];
x[0] ^= x[2] | x[1];
x[1] ^= x[2] | x[0];
x[2] = ~x[2];
// Extend XOR
x[7] ^= x[1]; x[3] ^= x[2]; x[4] ^= x[0];
//S5_2
t[0] = x[7]; t[1] = x[3]; t[2] = x[4];
x[6] ^= (t[0] & x[5]);
t[0] ^= x[6];
x[6] ^= (t[2] | t[1]);
t[1] ^= x[5];
x[5] ^= (x[6] | t[2]);
t[2] ^= (t[1] & t[0]);
// truncate XOR and swap
x[2] ^= t[0]; t[0] = x[1] ^ t[2]; x[1] = x[0]^t[1];
x[0] = x[7]; x[7] = t[0];
t[1] = x[3]; x[3] = x[6]; x[6] = t[1];
t[2] = x[4]; x[4] = x[5]; x[5] = t[2];
// Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

[5] For a higher resistance against DC and LC, swapping bits is additionally conducted in the S_8 design.

Listing 1.3. The bitsliced implementation of the inverse S_8 (in C code)

```
//(MSb: x[7], LSb: x[0]) :"b" represents bit
// Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
t[0] = x[7]; x[7] = x[0]; x[0] = x[1]; x[1] = t[0];
t[0] = x[7]; t[1] = x[6]; t[2] = x[5];
// S52 inv
x[4] ^= (x[3] | t[2]);
x[3] ^= (t[2] | t[1]);
t[1] ^= x[4];
t[0] ^= x[3];
t[2] ^= (t[1] & t[0]);
x[3] ^= (x[4] & x[7]);
//  Extended XOR
x[0] ^= t[1]; x[1] ^= t[2]; x[2] ^= t[0];
t[0] = x[3]; x[3] = x[6]; x[6] = t[0];
t[0] = x[5]; x[5] = x[4]; x[4] = t[0];
//  Truncated XOR
x[7] ^= x[1]; x[3] ^= x[2]; x[4] ^= x[0];
// Inv_S5_1
x[4] ^= (x[5] & x[6]);
x[5] ^= x[7];
x[3] ^= (x[4] | x[5]);
x[6] ^= x[3];
x[7] ^= x[4];
x[4] ^= (x[3] & x[5]);
x[5] ^= (x[7] & x[6]);
// Inv_S3
x[2] = ~x[2];
x[1] ^= x[2] | x[0];
x[0] ^= x[2] | x[1];
x[2] ^= x[1] & x[0];
// Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

Listing 1.4. The bitsliced implementation of the S-box with both DBN and LBN of 3 constructed by the Feistel structure (in C code)

```
//(MSb: x[7], LSb: x[0]) :"b" represents bit
// Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
t[0] = x[4];  t[1] = x[5];  t[2] = x[6];  t[3] = x[7];
//S4
t[4] = x[6];
x[7] ^= (x[6] | x[5]);
x[6] = (x[5] ^ (x[6] & x[7]));
x[5] = (x[4] ^ x[7]);
x[4] = (x[7] ^ (x[6] | x[5]));
x[7] = (t[4] ^ x[4]);
x[4] ^= (x[7] & x[5]);
//XOR and Swap
x[4] ^= x[0]; x[5] ^= x[1]; x[6] ^= x[2]; x[7] ^= x[3];
```

```
x[0] = t[0];  x[1] = t[1];  x[2] = t[2];  x[3] = t[3];
t[0] = x[4];  t[1] = x[5];  t[2] = x[6];  t[3] = x[7];
//S4
t[4] = x[6];
x[7] ^= (x[6] | x[5]);
x[6] = (x[5] ^ (x[6] & x[7]));
x[5] = (x[4] ^ x[7]);
x[4] = (x[7] ^ (x[6] | x[5]));
x[7] = (t[4] ^ x[4]);
x[4] ^= (x[7] & x[5]);
//XOR and Swap
x[4] ^= x[0]; x[5] ^= x[1]; x[6] ^= x[2]; x[7] ^= x[3];
x[0] = t[0];  x[1] = t[1];  x[2] = t[2];  x[3] = t[3];
t[0] = x[4];  t[1] = x[5];  t[2] = x[6];  t[3] = x[7];
//S4
t[4] = x[6];
x[7] ^= (x[6] | x[5]);
x[6] = (x[5] ^ (x[6] & x[7]));
x[5] = (x[4] ^ x[7]);
x[4] = (x[7] ^ (x[6] | x[5]));
x[7] = (t[4] ^ x[4]);
x[4] ^= (x[7] & x[5]);
//XOR and Swap
x[0] ^= x[4]; x[1] ^= x[5]; x[2] ^= x[6]; x[3] ^= x[7];
x[4] = t[0];  x[5] = t[1];  x[6] = t[2];  x[7] = t[3];
// Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

Listing 1.5. The bitsliced implementation of the S-box with both DBN and LBN of 3 constructed by the Lai-Massey structure (in C code)

```
//(MSb: x[7], LSb: x[0]) :"b" represents bit
// Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
// XOR
t[0]=x[4]^x[0];t[1]=x[5]^x[1];t[2]=x[6]^x[2];t[3]=x[7]^x[3];
// S5_1
t[4] = t[2];
t[3] ^= (t[2] | t[1]);
t[2] = (t[1] ^ (t[2] & t[3]));
t[1] = (t[0] ^ t[3]);
t[0] = (t[3] ^ (t[2] | t[1]));
t[3] = (t[4] ^ t[0]);
t[0] ^= (t[3] & t[1]);
// XOR
x[4]^=t[0]; x[5]^=t[1]; x[6]^=t[2]; x[7]^=t[3];
// S5_2
t[4] = x[6];
x[7] ^= (x[6] | x[5]);
x[6] = (x[5] ^ (x[6] & x[7]));
x[5] = (x[4] ^ x[7]);
x[4] = (x[7] ^ (x[6] | x[5]));
```

```
x[7] = (t[4] ^ x[4]);
x[4] ^= (x[7] & x[5]);
// XOR
x[0]^=t[0]; x[1]^=t[1]; x[2]^=t[2]; x[3]^=t[3];
// S5_3
x[2] ^= (x[1]& x[0]);
x[0] ^= x[2];
x[1] ^= x[3];
x[2] ^= (x[3] | x[1]);
x[3] ^= x[0];
x[0] ^= (x[2]| x[1]);
x[1] ^= (x[2]& x[0]);
// Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

Listing 1.6. The bitsliced implementation of the S-box with both DBN and LBN of 3 constructed by the unbalanced-MISTY structure (in C code)

```
//(MSb: x[7], LSb: x[0]) :"b" represents bit
// Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
// S5_1
x[6]^=(x[7] & x[3]);
x[7]^=x[6];
x[4]^=(x[7] & x[5]);
x[5]^=x[4];
x[7]^=(x[3] | x[4]);
x[4]^=x[6];
x[3]^=(x[6] | x[5]);
// Extend XOR
x[7] ^= x[0];x[6] ^= x[2];x[5] ^= x[1];
// S3
x[1] = ~x[1];
x[1] ^= x[0] & x[2];
x[0] ^= x[2] | x[1];
x[2] ^= x[0] & x[1];
//  Truncated XOR
x[2] ^= x[7];x[1] ^= x[6];x[0] ^= x[5];
// S5_2
x[4] ^= (x[7] & x[5]);
x[7] ^= x[3];
x[3] ^= x[4];
x[6] ^= (x[4] & x[7]);
x[5] ^= x[4];
x[3] ^= (x[6] & x[5]);
x[5] ^= (x[3] | x[6]);
// Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

Listing 1.7. The bitsliced implementation of the S-box with DBN of 4 and LBN of 3 constructed by the unbalanced-Bridge (in C code)

```
//(MSb: x[7], LSb: x[0]) :"b" represents bit
```

```
// Input: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
// S5_1
t[0] = x[7] ^ x[5];
t[1] = x[6] ^ t[0];
t[2] = x[3] ^ x[4];
t[3] = x[7] ^ (t[0] | t[1]);
t[4] = x[5] ^ (x[7] & t[1]);
x[5] = t[3] ^ x[6] ^ t[2];
x[6] = t[1] ^ (x[4] | x[3]);
x[3] = x[4];
x[7] = t[2] ^ x[6];
x[4] = t[4];
// S3
t[0] = x[1] ^ x[2];
t[1] = x[0] ^ t[0];
t[2] = t[1] | x[1];
t[3] = t[1] & t[0];
x[1] = t[3] ^ t[2];
x[0] = x[2] ^ t[3];
x[2] = t[1];
// XOR
x[7] ^= x[2];x[6] ^= x[1];x[5] ^= x[0];
// S5_2
t[0] = x[6] ^ x[7];
t[1] = t[0] ^ x[3];
t[2] = t[1] ^ (x[5] | x[6]);
t[3] = x[4] ^ (t[2] & x[3]);
t[4] = x[6] ^ t[3];
t[1] ^= (x[4] & x[5]);
x[3] = x[5] ^ t[4];
x[4] = x[3] ^ t[2];
t[2] = t[1] ^ x[5];
t[0] ^= x[5];
// XOR
x[2] ^= t[2];x[1] ^= t[1];x[0] ^= t[0];
// Output: x[7], x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

Listing 1.8. The bitsliced implementation of the 6-bit S-box with both DBN and LBN of 3 constructed by the Feistel structure (in C code)

```
//(MSb: x[5], LSb: x[0]) :"b" represents bit
// Input: x[5], x[4], x[3], x[2], x[1], x[0]
// S3_1
t[2] = x[4] ^ x[5];
t[1] = x[5] ^ x[3];
t[0] = x[4] | x[3];
t[0] = t[1] ^ t[0];
t[1] = t[1] | t[2];
t[2] = t[2] & x[3];
// XOR
```

```
x[0]^=t[0]; x[1]^=t[1]; x[2]^=t[2];
// S3_2
t[2] = x[0] & x[1];
t[2] = t[2] ^ x[2];
t[0] = x[1] | x[2];
t[0] = t[0] ^ x[0];
t[1] = x[2] & t[0];
t[1] = t[1] ^ x[1];
// XOR
x[3]^=t[0]; x[4]^=t[1]; x[5]^=t[2];
// S3_3
t[2] = x[4] & x[3];
t[1] = t[2] ^ x[5];
t[2] = x[5] | x[4];
t[2] = x[3] ^ t[2];
t[0] = t[2] ^ x[4];
t[0] = x[5] & t[0];
// XOR
x[0]^=t[0]; x[1]^=t[1]; x[2]^=t[2];
// Output: x[5], x[4], x[3], x[2], x[1], x[0]
```

Listing 1.9. The bitsliced implementation of the 7-bit S-box with both DBN and LBN of 3 constructed by unbalanced-MISTY structure (in C code)

```
//(MSb: x[6], LSb: x[0]) :"b" represents bit
// Input: x[6], x[5], x[4], x[3], x[2], x[1], x[0]
// S4_1
x[4] ^= x[5] & x[3];
x[5] ^= x[4];
x[3] ^= x[6];
x[4] ^= x[6] | x[3];
x[6] ^= x[5];
x[5] ^= x[3] | x[4];
x[3] ^= x[5] & x[4];
T[0]=x[6];  x[6] = x[3];  x[3] = T[0];
// Extend XOR
x[4]^=x[0]; x[5]^=x[1]; x[6]^=x[2];
// S3
T[0] = x[1] | x[2];
T[2] = x[1];
x[1] = T[0] ^ x[0];
T[1] = ~x[2];
T[0] = x[1] & x[2];
x[2] = T[2] ^ T[0];
T[0] = T[2] | x[1];
x[0] = T[0] ^ T[1];
// Truncated XOR
x[0]^=x[4]; x[1]^=x[5]; x[2]^=x[6];
// S4_2
x[5] ^= x[6]& x[4];
```

```
x[6] ^= x[5];
x[4] ^= x[3];
x[5] ^= x[3] | x[4];
x[3] ^= x[6];
x[6] ^= x[4] | x[5];
x[4] ^= x[6] & x[5];
T[0] = x[4];  x[4] = x[3];  x[3] = T[0];
// Output: x[6], x[5], x[4], x[3], x[2], x[1], x[0]
```

D Detailed Security Analysis of PIPO

We provide a security analysis of PIPO against relevant and powerful attacks.

D.1 Differential Cryptanalysis

Differential Cryptanalysis [20] (DC) is one of the most powerful attacks on block ciphers. After examining all possible differential trails using the branch and bound technique [58], we found the minimum numbers of differential active S-boxes and probabilities of the best differential trails for up to 7 rounds (Table 12). The best of these differential trails reaches 6 rounds with a probability of $2^{-54.4}$, and 18,944 such 6-round trails were found, each with different input and output differences. One of them is given in Fig. 6.

Table 12. Minimum numbers of differential active S-boxes and probabilities of best differential trails

	Rounds						
	1	2	3	4	5	6	7
#(Active S-box)	1	2	4	6	9	11	13
Prob. of best trail	2^{-4}	2^{-8}	2^{-16}	$2^{-26.8}$	$2^{-40.4}$	$2^{-54.4}$	2^{-65}

In order to obtain a differential probability, we need to investigate all differential trails with the same input and output differences and sum up their probabilities. For the best 6 and 7-round differential trails mentioned above, we repeatedly searched for the next-best possible differential trails until these trails made only negligible contributions to the differential probability. This search showed that the best differential probabilities for 6 and 7-round PIPO are not greater than 2^{-54} and 2^{-64}, respectively. We could append three rounds and five rounds to the best 6-round differentials as the key recovery of PIPO-64/128 and PIPO-64/256, respectively. A detailed attack on 9-round PIPO-64/128 (together with the computation of differential probabilities) is described below.

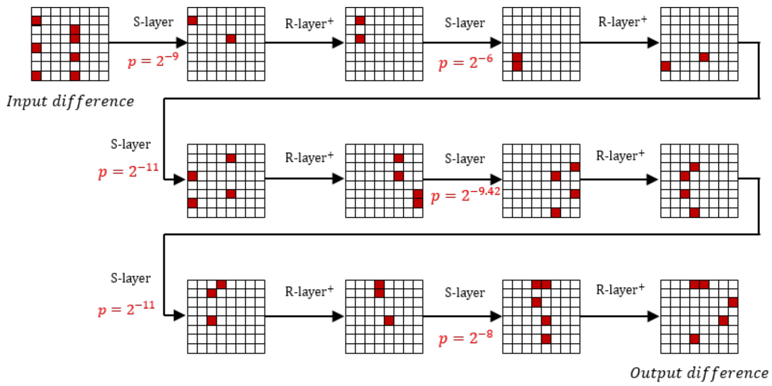

Fig. 6. 6-round differential trail with probability $2^{-54.4}$ (R-layer^{+} : R-layer followed by round key and constant-XOR)

9-Round Differential Attack on* PIPO*-64/128. As stated in Section D.1, the best differential trails reach 6 rounds with probability $2^{-54.4}$, and the number of such trails we found is 18,944. The number of these trails is reduced to 2,368 except for all rotation equivalences. In order to consider the differential effect, we repeatedly searched for the next-best possible 6-round differential trails whose probabilities are between $2^{-54.4}$ and $2^{-64.4}$. Our simulations demonstrate that at most 4 differential trails contribute to a differential. Consequently, each summation of the relevant probabilities ranges from $2^{-54.3729}$ to $2^{-54.415}$. Refer to Table 13 for more details.

Based on the differential trail depicted in Fig. 6, we could find the 6-round differential ($\varDelta 8800088008088000 \rightarrow \varDelta 0010000200010018$) with probability $2^{-54.4087}$. For a better understanding of our differential attack, each state is re-ordered with S-box input-wise (column-wise) representation (*e.g.*, $\varDelta 8800088008088000 \xrightarrow{re-order} \varDelta 92000000AC000000$ and $\varDelta 0010000200010018 \xrightarrow{re-order} \varDelta 0000004101001004$). Hereinafter, we consider re-ordered differentials and values. Adding one and two rounds at the beginning and the end of the differential respectively, we could attack 9-round PIPO. The following notation is used to describe our differential attack.

- $\varDelta S^r$: The difference in outputs of the r-th round's S-layer.
- $\varDelta R^r$: The difference in outputs of the r-th round's R-layer.
- $\varDelta K^r$: The difference in outputs of the r-th round's key and constant-XOR.
- $\varDelta K^0$: The difference in outputs of the whitening key-XOR.
- S^{-1} : The inverse S-box.
- $\mathbb{S}$: The inverse S-layer.
- $\mathbb{R}^{-1}$: The inverse R-layer.
- $Y[i]$: The 8-bit value in the i-th column of a 64-bit Y (i starts from the right).

Table 13. 6-round differentials and their probabilities

Probabilities of differential trails contributing to a differential	Differential Prob.	Number of differentials
$2^{-54.415}, 2^{-60.0}, 2^{-61.8301}, 2^{-62.8301}$	$2^{-54.3729}$	8
$2^{-54.415}, 2^{-60.0}, 2^{-62.8301}, 2^{-63.8301}$	$2^{-54.3791}$	16
$2^{-54.415}, 2^{-60.0}, 2^{-64.0}, 2^{-64.0}$	$2^{-54.3816}$	16
$2^{-54.415}, 2^{-60.0}$	$2^{-54.3853}$	88
$2^{-54.415}, 2^{-61.0}, 2^{-61.8301}, 2^{-62.8301}$	$2^{-54.3876}$	4
$2^{-54.415}, 2^{-61.0}, 2^{-62.8301}, 2^{-63.8301}$	$2^{-54.3938}$	8
$2^{-54.415}, 2^{-61.8301}, 2^{-62.0}, 2^{-62.8301}$	$2^{-54.3949}$	8
$2^{-54.415}, 2^{-61.0}, 2^{-64.0}, 2^{-64.0}$	$2^{-54.3963}$	8
$2^{-54.415}, 2^{-61.0}$	$2^{-54.4001}$	44
$2^{-54.415}, 2^{-62.0}, 2^{-62.8301}, 2^{-63.8301}$	$2^{-54.4012}$	16
$2^{-54.415}, 2^{-61.8301}, 2^{-62.8301}$	$2^{-54.4024}$	128
$2^{-54.415}, 2^{-62.0}, 2^{-64.0}, 2^{-64.0}$	$2^{-54.4038}$	16
$2^{-54.415}, 2^{-62.0}$	$2^{-54.4075}$	88
$2^{-54.415}, 2^{-62.8301}, 2^{-63.8301}$	$2^{-54.4087}$	256
$2^{-54.415}, 2^{-64.0}, 2^{-64.0}$	$2^{-54.4112}$	88
$2^{-54.415}$	$2^{-54.415}$	1,576
	Total	2,368

- $Y[i, j, ..., k]$: The concatenation of $Y[i]$, $Y[j]$, ... , and $Y[k]$.
- RRK_i : The re-ordered state of $RK_i \oplus c_i$ where RK_i and c_i are the i-th round key and constant.
- RRK'_i : $\mathbb{R}^{-1}(RRK_i)$.

The 9-round differential attack is outlined in Table 14. Note that the 20-bit of $RRK'_8[0,1,3,4]$ can be derived from $RRK_0[0,1,5,6,7]$ since the whitening key RK_0 and the 8-th round key RK_8 equal as K_0 according to the key schedule for PIPO (128-bit master key $K = K_1||K_0$).

Data Collection. We establish structures consisting of 2^{40} plaintexts which have all distinct values on 0, 1, 5, 6, and 7-th columns and a fixed value on the other columns. Since plaintexts in each structure have all distinct values on the corresponding columns, we can match 2^{39} pairs in a structure whose differences all satisfy ΔS^1 after guessing the re-ordered whitening key $RRK_0[0,1,5,6,7]$. As the 7-th round output difference of such a pair has a probability of $2^{-54.4087}$ to satisfy ΔK^7 with the right key, each structure is expected to have $2^{-15.4087}$ right pairs with the right key guess. So as to expect the number of the right pairs to be four, we chose to establish $2^{17.4087}$ structures. Thus the total data complexity for our attack is $2^{17.4087} \times 2^{40} = 2^{57.4087}$.

Key Recovery. Our key recovery includes the key guess for partial 52-bit of K_0 and all 64-bit of K_1. Totally, we need 2^{116} counters for the guessed keys. Algorithm 1 presents our key recovery procedure in detail. Taking advantage of the early abort technique at ΔK^8 and ΔK^7, the time complexity is about $2^{17.4087} \times$

Table 14. The 9-round differential attack on PIPO : col. means column, and "0" and "1" are one-bit differences 0 and 1, respectively, while the "?" denotes an undetermined one-bit difference.

		7-th col.	6-th col.	5-th col.	4-th col.	3-rd col.	2-nd col.	1-st col.	0-th col.	prob.
	ΔK^0	01??1???	????????	1?????1?	00000000	00000000	00000000	0???????	????????	1
1R	ΔS^1	00000100	00100000	10000000	00000000	00000000	00000000	10010000	00001010	
	ΔR^1	10010010	00000000	00000000	00000000	10101100	00000000	00000000	00000000	
	ΔK^1	10010010	00000000	00000000	00000000	10101100	00000000	00000000	00000000	
2R	ΔS^2					$\vdots$				$2^{-54.4087}$
$\sim$	$\cdots$									
7R	ΔK^7	00000000	00000000	00000000	01000001	00000001	00000000	00010000	00000100	
8R	ΔS^8	00000000	00000000	00000000	????????	??1?????	00000000	???????1	?1??????	1
	ΔR^8	000????0	?0???000	???00?00	0?00??0?	?000?0??	??0?00?0	01??0001	00100???	
	ΔK^8	000????0	?0???000	???00?00	0?00??0?	?000?0??	??0?00?0	01??0001	00100???	
9R	ΔS^9	????????	????????	????????	????????	????????	????????	????????	????????	
	ΔR^9	????????	????????	????????	????????	????????	????????	????????	????????	
	ΔK^9	????????	????????	????????	????????	????????	????????	????????	????????	

$$2^{40} \times (2^{40} \times 5 + \underbrace{(2^{47} + 2^{49} + 2^{52} + 2^{56} + \cdots + 2^{72} + 2^{71} + 2^{66} + 2^{62} + 2^{58}) \times 2)}_{\text{the early abort technique}}$$

$\approx 2^{131.0717}$ S-box look-ups, equivalently about $2^{124.9017}$ 9-round PIPO encryptions.

Algorithm 1: Key recovery procedure on 9-round PIPO

for each of the prepared $2^{17.4087}$ structures **do**
  `// A structure consists of` 2^{40} `of` $(P_i,\ C_i)$
  for each guess for $RRK_0[0,1,5,6,7]$ **do**
    $rrk_0[0,1,5,6,7] \leftarrow$ the 40-bit guess
    for each plaintext(P_i) in a structure **do**
      $P'_i[0,1,5,6,7] \leftarrow \mathbb{S}(P_i[0,1,5,6,7] \oplus rrk_0[0,1,5,6,7])$
    end
    Match all (C_i, C_j) where $P'_i[0,1,5,6,7] \oplus P'_j[0,1,5,6,7] = \Delta S^1[0,1,5,6,7]$.
    `//` 2^{39} `distinct pairs` (C_i, C_j) `are matched in each structure.`
    `// The following applies the early abort technique for` RRK'_9 `and` $RRK'_8[0,1,3,4]$.
    for each of the matched pairs (C_i, C_j) **do**
      $C'_i \leftarrow \mathbb{R}^{-1}(C_i)$, $C'_j \leftarrow \mathbb{R}^{-1}(C_j)$
      `// By the order 1,0,2,3,4,5,6,7-th columns of` ΔK^8
      `// 8-bit guess and 6-bit filtering`
      for each guess for $RRK'_9[1]$ **do**
        $rrk'_9[1] \leftarrow$ the 8-bit guess
        $k_i^9[1] \leftarrow S^{-1}(C'_i[1] \oplus rrk'_9[1])$, $k_j^9[1] \leftarrow S^{-1}(C'_j[1] \oplus rrk'_9[1])$
        if $(k_i^9[1] \oplus k_j^9[1]) \neq \Delta K^8[1]$ **then** break
        `// 8-bit guess and 5-bit filtering`
        for each guess for $RRK'_9[0]$ **do**
          $\ddots$
          `// 8-bit guess and 4-bit filtering`
          for each guess for $RRK'_9[7]$ **do**
            $rrk'_9[7] \leftarrow$ the 8-bit guess
            $k_i^9[7] \leftarrow S^{-1}(C'_i[7] \oplus rrk'_9[7])$, $k_j^9[7] \leftarrow S^{-1}(C'_j[7] \oplus rrk'_9[7])$
            if $(k_i^9[7] \oplus k_j^9[7]) \neq \Delta K^8[7]$ **then** break
            $C''_i \leftarrow \mathbb{R}^{-1}(k_i^9)$, $C''_j \leftarrow \mathbb{R}^{-1}(k_j^9)$
            `// By the order 0,1,3,4-th columns of` ΔK^7
            `// 3-bit guess and 8-bit filtering`
            for each possible guess for $RRK'_8[0]$ **do**
              $rrk'_8[0] \leftarrow$ the 3-bit guess and 5-bit derivation from $rrk_0[0,1,5,6,7]$
              $\Delta k^7[0] \leftarrow S^{-1}(C''_i[0] \oplus rrk'_8[0]) \oplus S^{-1}(C''_j[0] \oplus rrk'_8[0])$
              if $\Delta k^7[0] \neq \Delta K^7[0]$ **then** break
              $\ddots$
              `// 3-bit guess and 7-bit filtering`
              for each possible guess for $RRK'_8[4]$ **do**
                $rrk'_8[4] \leftarrow$ the 3-bit guess and 5-bit derivation from $rrk_0[0,1,5,6,7]$
                $\Delta k^7[4] \leftarrow S^{-1}(C''_i[0] \oplus rrk'_8[4]) \oplus S^{-1}(C''_j[4] \oplus rrk'_8[4])$
                if $(\Delta k^7[4] = \Delta K^7[4]$ **then**
                  Increase the corresponding 116-bit key counter.
                end
              end
            end
          end
        end
      end
    end
  end
end
Derive partial 52-bit of K_0 and 64-bit of K_1 from the max-counted re-ordered key.

D.2 Linear Cryptanalysis

Linear Cryptanalysis [56] (LC), along with DC, is a powerful attack on block ciphers. Given a linear trail (linear approximation) with probability p, the bias ϵ is defined as $p - \frac{1}{2}$ and the correlation potential [33] as $4\epsilon^2$. For LC to work on an n-bit block cipher, the correlation potential should be greater than 2^{-n}.

We investigated all possible linear trails for up to 7 rounds, in order to find the minimum numbers of linear active S-boxes and the correlation potentials of the best linear trails (Table 15). The best of these linear trails reaches 6 rounds with a correlation potential of 2^{-52}, and 768 such 6-round trails were found, each with different input and output masks. An example trail is presented in Fig. 7.

Table 15. Minimum numbers of linear active S-boxes and best correlation potentials of linear trails

	Rounds						
	1	2	3	4	5	6	7
#(Active S-box)	1	2	4	6	9	11	13
Best correlation potential	2^{-4}	2^{-8}	2^{-16}	2^{-24}	2^{-38}	2^{-52}	2^{-66}

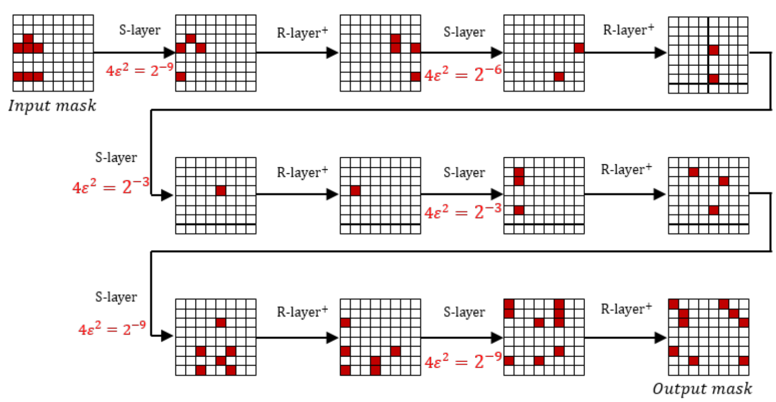

Fig. 7. 6-round linear trail with correlation potential 2^{-52}

The average correlation potential, which is a more accurate metric for LC, is the sum of the correlation potentials of all linear trails with the same input and output masks [33,74]. To calculate this, we searched for the next-best linear trails with the same input and output masks used by the best 6 and 7-round trails. However, we found that only a few linear trails improved the correlation potential, so we concluded that the best average correlation potentials for 6 and 7-round PIPO are not greater than 2^{-51} and 2^{-64}, respectively. Similarly to DC, a LC based key recovery attack could be applied up to 9-round PIPO-64/128 and 11-round PIPO-64/256.

D.3 Impossible Differential Attack

Impossible differential cryptanalysis [17] exploits impossible differentials. When a differential has probability zero, the differential is called an impossible differential. To search for impossible differentials, we developed a SAT-based finder that collects zero-probability differentials with given input and output differences for a reduced-round PIPO [60]. We investigated whether there are impossible differentials satisfying the following conditions which are expected to go through the longest rounds: the input difference activates one S-box, and the output difference activates one S-box.

In total, there are $8 \times 255 = 2,040$ differences for input and output, which satisfy the above conditions, creating a search pool of $(2,040)^2 = 4,161,600$ pairs of input and output differences. After testing whether any of these 4,161,600 choices yielded impossible differentials for a 4 or 5-round PIPO, we found 52,856 4-round impossible differentials, and no 5-round impossible differentials. Using these impossible differentials we could not design any shortcut attack on more than 6 rounds of PIPO-64/128 or 8 rounds of PIPO-64/256.

D.4 Boomerang and Rectangle Attacks

The boomerang and rectangle attacks [18,69] exploit a variety of two independent differentials. These attacks are effective when an n-bit cipher satisfies $\hat{p} \times \hat{q} \leq 2^{-n/2}$, where

$$\hat{p} = \sqrt{\sum_{\beta} Pr^2[\alpha \to \beta]}, \text{ and } \hat{q} = \sqrt{\sum_{\gamma} Pr^2[\gamma \to \delta]}.$$

Based on the best 3 and 4-round differential trails (Table 12), we computed $\hat{p}$ and $\hat{q}$. For 3 rounds, we investigated all differential trails with probabilities in the range $2^{-24} \sim 2^{-16}$, obtaining approximate values of $\hat{p} = 2^{-12.11}$ and $\hat{q} = 2^{-13.86}$. For 4 rounds, we investigated all differential trails with probabilities in the range $2^{-32} \sim 2^{-24}$, obtaining approximate values of $\hat{p} = 2^{-22.94}$ and $\hat{q} = 2^{-22.23}$. For more details, see Table 16 (note that differential trails with probabilities less than the minimum probabilities in Table 16 have minor contributions to $\hat{p}$ and $\hat{q}$).

These results indicate that PIPO has 6-round boomerang and rectangle distinguishers that allow for key recovery attacks on at most 8 rounds of PIPO-64/128 and 10 rounds of PIPO-64/256 (unlike DC, these attacks are hard to have filtering effects of partially decrypted data for each guessed key). We also confirmed that advanced techniques such as boomerang switch [22,70] are not applicable to PIPO. Thus, we believe that PIPO cannot be compromised by boomerang or rectangle attacks.

Table 16. Numbers of 3 and 4-round differential trails with respect to probabilities

3 rounds	for $\hat{p}$	for $\hat{q}$	4 rounds	for $\hat{p}$	for $\hat{q}$
Prob.	Number of trails		Prob.	Number of trails	
$2^{-16} = p$	64	32	$2^{-24} = p$	0	0
$2^{-16} > p \geq 2^{-17}$	512	0	$2^{-24} > p \geq 2^{-25}$	0	0
$2^{-17} > p \geq 2^{-18}$	904	64	$2^{-25} > p \geq 2^{-26}$	0	0
$2^{-18} > p \geq 2^{-19}$	5,024	0	$2^{-26} > p \geq 2^{-27}$	56	128
$2^{-19} > p \geq 2^{-20}$	7,380	512	$2^{-27} > p \geq 2^{-28}$	688	576
$2^{-20} > p \geq 2^{-21}$	12,560	0	$2^{-28} > p \geq 2^{-29}$	2,176	960
$2^{-21} > p \geq 2^{-22}$	7,488	1,546	$2^{-29} > p \geq 2^{-30}$	1,598	2,816
$2^{-22} > p \geq 2^{-23}$	4,416	2,395	$2^{-30} > p \geq 2^{-31}$	3,088	5,472
$2^{-23} > p \geq 2^{-24}$	6,656	4,847	$2^{-31} > p \geq 2^{-32}$	5,000	19,936
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

*In this table, p is the probability of differential trails.

D.5 Algebraic Attack

The S-boxes S_3, S_5^1, and S_5^2 used in PIPO are described by 14, 25, and 25 quadratic equations and 6, 10 and 10 variables over $GF(2)$, respectively. Since PIPO uses eight S_8s per round, it can be expressed by $64 \times 8 \times 13$ quadratic equations in $26 \times 8 \times 13$ variables. Therefore, it requires 6,656 quadratic equations and 2,704 variables, more than those required by AES (consisting of 6,400 equations in 2,560 variables [30]). This indicates that PIPO provides a high level of security against algebraic attacks.

D.6 Integral Attack

Using the method presented in [27], we found the cumulative algebraic degrees of several PIPO rounds (Table 17). The cumulative algebraic degree is calculated over plaintext and key variables. Since PIPO encrypts 64-bit data blocks and has a cumulative algebraic degree of 63 after 5 rounds, it would be difficult to create an r-round integral distinguisher for $r \geq 5$. Thus, we believe that PIPO is resistant to the integral attack.

Table 17. Cumulative algebraic degrees of PIPO

# of rounds	1	2	3	4	5	6	7	$\cdots$
Cumulative algebraic degrees	5	25	57	62	63	63	63	$\cdots$

D.7 Statistical Saturation Attack

For 4 selected S-box positions, 16 out of 32 bits are directed to the same positions even after the R-layer is applied, as indicated in Fig. 8. This weak R-layer diffusion can be targeted by the statistical saturation attack [31].

A series of simulations were performed to test the statistical saturation attack on PIPO. These simulations can be classified into 5 sets. Each set is independent of the others (*i.e.*, they all use randomly generated different keys), it uses a single key, and it includes 10 experiments from which the average squared Euclidean distance is calculated. In each experiment, a squared Euclidean distance between a uniform distribution and a 16-bit distribution (black cells in Fig. 8) after 2^{32} plaintexts were computed. These cells, which are all fixed by the same 32-bit value in colored cells and receive all values in the white cells (on the left side of Fig. 8), are encrypted by 2∼4 rounds of PIPO. Simulation results are presented in Table 18.

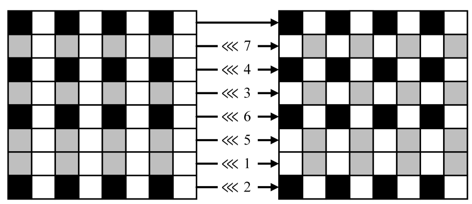

Fig. 8. A weak diffusion of the R-layer on 4 selected S-boxes

Table 18. Experimental results on the average squared Euclidean distances with 2^{32} plaintexts

	2-round	3-round	4-round
Simulation 1	$2^{-12.580}$	$2^{-20.900}$	$2^{-30.783}$
Simulation 2	$2^{-12.529}$	$2^{-20.977}$	$2^{-30.656}$
Simulation 3	$2^{-12.358}$	$2^{-20.908}$	$2^{-30.902}$
Simulation 4	$2^{-12.645}$	$2^{-20.766}$	$2^{-30.712}$
Simulation 5	$2^{-12.492}$	$2^{-20.888}$	$2^{-30.622}$

The above simulation results indicate that the addition of a round multiplies the distance by a factor of approximately 2^{-9}. Assuming the distance continues to decrease by a similar factor, PIPO with more than 7 rounds would have no statistical saturation distinguisher. Thus, we believe that PIPO is sufficiently resistant to the statistical saturation attack.

D.8 Invariant Subspace Attack

The invariant subspace attack exploits a subspace A and constants u, v such that $F(u \oplus A) = v \oplus A$, where F is a round transformation of a block cipher [52,53]. For the round key $rk \in A \oplus u \oplus v$, $F \oplus rk$ maps the subspace $u \oplus A$ onto itself, because $F(u \oplus A) \oplus rk = v \oplus A \oplus rk = u \oplus A$. However, we can avoid this invariant subspace by using appropriate round constants; recall that PIPO uses a round constant (counter) that is XORed with the least-significant byte of the state at the end of each round.

We investigated all possible invariant subspace transitions in the S_8, finding a total of 124 invariant subspace transitions (excluding dimension 8); 120 and 4 such transitions exist in dimentions 1 and 2, respectively. One such example is $\{0\text{x}00, 0\text{x}6F\} \oplus 0\text{x}25 \xrightarrow{S_8} \{0\text{x}00, 0\text{x}6F\} \oplus 0\text{x}BE$. If we disregard the R-layer and round constant, and the corresponding round key byte is in the $\{0\text{x}00, 0\text{x}6F\} \oplus 0\text{x}9B$, then we can use this invariant subspace transition again in the next round since $0\text{x}BE \oplus 0\text{x}9B = 0\text{x}25$.

However, XORing a different constant with the state in each round breaks all the invariant subspaces, even though we can bypass the R-layer by applying the same input subspace to all 8 S-boxes in the S-layer. We confirmed by simulation that there are no invariant subspaces in PIPO.

D.9 Nonlinear Invariant Attack

The nonlinear invariant attack [67] exploits nonlinear invariant equations through ciphers (for some weak-key classes). This attack can be mounted when 1) the S-box has at least one nonlinear invariant equation with probability one and 2) the equations generated by each round can be XORed to produce an equation whose variables consist purely of plaintext, ciphertext, and round key bits.

PIPO uses different rotations for different rows to send all the 8 output bits of an S_8 to the inputs of different S_8's in the next round, breaking the second condition. Thus, PIPO is secure against the nonlinear invariant attack.

D.10 Meet-in-the-Middle Attack

We here present a key recovery attack against 6-round PIPO-64/128 using meet-in-the-middle (MITM) attack with splice-and-cut and initial-structure (IS) techniques [4,62]. In this analysis, 6-round PIPO-64/128 is separated into 5 chunks, as shown in Table 19.

Table 19. Chunk separation for 6-round MITM attack on PIPO-64/128

Roundkey	RK_0	RK_1	RK_2	RK_3	RK_4	RK_5	RK_6
Subkey	K_0	K_1	K_0	K_1	K_0	K_1	K_0
Chunk	$\leftarrow$	IS		$\rightarrow$	PM		$\leftarrow$

Since PIPO-64/128 achieves full diffusion in 2 rounds and uses the round keys alternately, if more than 2 rounds are allocated to the IS or partial match (PM) process, the propagation of the neutral bit is bound to overlap. In the whole steps of MITM analysis, K_1 is used for the forward computations whereas K_0 is used for computation in the opposite direction. The IS and PM porcesses are illustrated in Figures 9 and 10.

By carefully setting 10 neutral bits for each of K_0 and K_1 (colored in blue and red, respectively), the propagations of neutral bits in the forward and backward computation do not overlap. It is assumed that bits other than the 20 neutral bits are fixed. In the analysis, we use the notation S_r^I, S_r^S and S_r^R to denote the initial state of round, the state after S-layer, and the state after R-layer in round r, respectively. In IS, we fix 32 state bits in S_1^R and 32 state bits in S_3^I (colored in green) which are not affected by the backward and forward computations, respectively. Then, one can compute the value of S_1^S (resp. S_3^S) in the backward (resp. forward) computation for each of the 2^{10} choices of neutral bits in K_0 (resp. K_1).

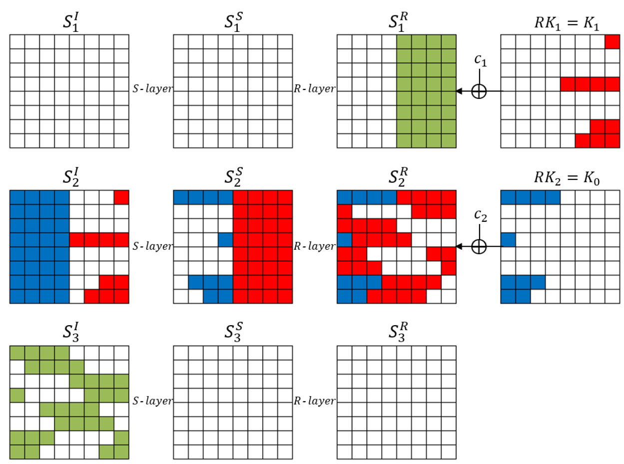

Fig. 9. 2-round initial structure for MITM attack

After IS, only one round of forward computation is possible because RK_4 is K_0 (which is the backward computation key). For each choice of neutral bits in K_1 (resp, K_0), one can compute 54 (resp, 32) bits of S_5^I, where 31 bits can be used for matching (colored in yellow in Fig. 10).

Then $2^{10} \times 2^{10} = 2^{20}$ of candidates are filtered out to 2^{-11} by probability 2^{-31} of partial matching. By repeating this process for each of the 108 values of keys not chosen as neutral bits, a total of $2^{108} \times 2^{-11} = 2^{97}$ candidates are

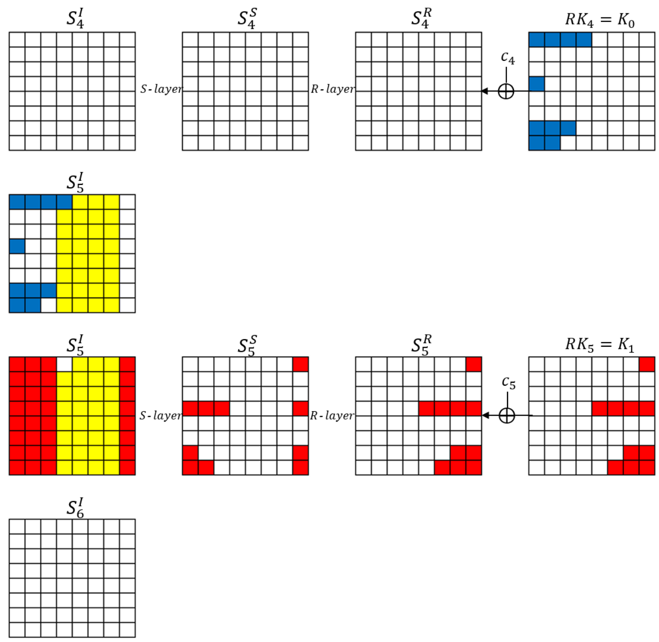

Fig. 10. 2-round partial matching for MITM attack

expected. Therefore, the time and memory complexity are $2^{108} \times 2^{10} + 2^{97} \approx 2^{118}$ and 2^{10}, respectively. The data complexity is 2^{64} because the 2^{108} queries require the knowledge of the full codebook.

We found that a key recovery attack against 10-round PIPO-64/256 is also possible by applying the same method. In the MITM attack on PIPO-64/256, K_3 is used for forward computation and K_0 is used for computation in the opposite direction, but they use the same neutral bits setting as in the 128-bit version attack. In this attack, 10-round PIPO-64/256 is separated as in Table 20.

Table 20. Chunk separation for 10-round MITM attack on PIPO-64/256

Roundkey	RK_0	RK_1	RK_2	RK_3	RK_4	RK_5	RK_6	RK_7	RK_8	RK_9	RK_{10}
Subkey	K_0	K_1	K_2	K_3	K_0	K_1	K_2	K_3	K_0	K_1	K_2
Chunk		$\leftarrow$		IS		$\rightarrow$		PM		$\leftarrow$	

D.11 Slide Attack

The slide attack exploits round functions that have self similarities [24]. Round-dependent constant-XORs in PIPO simply break self similarities in sliding round functions. Therefore, the slide attack does not apply to PIPO.

D.12 Attacks Using Related-Keys

The simple key schedule of PIPO enables us to make several related-key differential trails containing a few active S-boxes. However, as noted earlier, the resistance of PIPO against attacks using related keys, such as related-key differential [21] or related-key boomerang/rectangle attacks [19,49,50], is not considered. This is due to the fact that these kinds of attacks are unrealistic in most of resource-constrained environments. There have been many lightweight block ciphers that do not claim the related-key security [2,3,9,10,13,40,42].

E Bitsliced Implementations of Higher-Order Masked S-Layer and R-Layer

Listing 1.10. The bitsliced implementation of higher-order masked S-layer (in C code)

```
// ISW_AND(out,in1,in2): out=in1&in2, ISW_OR(out,in1,in2): out=in1|in2
// MSB: X[7][SHARES], LSB: X[0][SHARES]
// Input: X[i][SHARES], 0<=i<=7
// S5_1
Mask_refreshing(X[7]);
ISW_AND(T[3], X[7], X[6]);
for (i = 0; i < SHARES; i++) X[5][i] ^= T[3][i];
Mask_refreshing(X[3]);
ISW_AND(T[3], X[3], X[5]);
for (i = 0; i < SHARES; i++)
{X[4][i] ^= T[3][i]; X[7][i] ^= X[4][i]; X[6][i] ^= X[3][i];}
Mask_refreshing(X[4]);
ISW_OR(T[3], X[4], X[5]);
for (i = 0; i < SHARES; i++) {X[3][i] ^= T[3][i]; X[5][i] ^= X[7][i];}
Mask_refreshing(X[5]);
ISW_AND(T[3], X[5], X[6]);
for (i = 0; i < SHARES; i++) X[4][i] ^= T[3][i];
// S3
Mask_refreshing(X[1]);
ISW_AND(T[3], X[1], X[0]);
for (i = 0; i < SHARES; i++) X[2][i] ^= T[3][i];
Mask_refreshing(X[2]);
ISW_OR(T[3], X[2], X[1]);
for (i = 0; i < SHARES; i++) X[0][i] ^= T[3][i];
Mask_refreshing(X[2]);
ISW_OR(T[3], X[2], X[0]);
```

```
for (i = 0; i < SHARES; i++) X[1][i] ^= T[3][i];
X[2][0] = ~X[2][0];
// Extend XOR
for (i = 0; i < SHARES; i++)
{X[7][i] ^= X[1][i]; X[3][i] ^= X[2][i]; X[4][i] ^= X[0][i];}
// S5_2
for (i = 0; i < SHARES; i++)
{T[0][i] = X[7][i]; T[1][i] = X[3][i]; T[2][i] = X[4][i];}
Mask_refreshing(T[0]);
ISW_AND(T[3], T[0], X[5]);
for (i = 0; i < SHARES; i++) {X[6][i] ^= T[3][i]; T[0][i] ^= X[6][i];}
Mask_refreshing(T[2]);
ISW_OR(T[3], T[2], T[1]);
for (i = 0; i < SHARES; i++) {X[6][i] ^= T[3][i]; T[1][i] ^= X[5][i];}
Mask_refreshing(X[6]);
ISW_OR(T[3], X[6], T[2]);
for (i = 0; i < SHARES; i++) X[5][i] ^= T[3][i];
Mask_refreshing(T[1]);
ISW_AND(T[3] T[1] T[0]);
for (i = 0; i < SHARES; i++) T[2][i] ^= T[3][i];
// Truncate XOR
for (i = 0; i < SHARES; i++)
{X[2][i] ^= T[0][i];
T[0][i] = X[1][i] ^ T[2][i]; X[1][i] = X[0][i] ^ T[1][i];
X[0][i] = X[7][i]; X[7][i] = T[0][i]; T[1][i] = X[3][i];
X[3][i] = X[6][i]; X[6][i] = T[1][i]; T[2][i] = X[4][i];
X[4][i] = X[5][i]; X[5][i] = T[2][i];}
// Output: X[i][SHARES], 0<=i<=7
```

Listing 1.11. The bitsliced implementation of higher-order masked R-layer (in C code)

```
// MSB: X[7][SHARES], LSB: X[0][SHARES]
// Input: X[i][SHARES], 0<=i<=7
for(i=0;i<SHARES;i++)
{
X[1][i] = ((X[1][i] << 7)) | ((X[1][i] >> 1));
X[2][i] = ((X[2][i] << 4)) | ((X[2][i] >> 4));
X[3][i] = ((X[3][i] << 3)) | ((X[3][i] >> 5));
X[4][i] = ((X[4][i] << 6)) | ((X[4][i] >> 2));
X[5][i] = ((X[5][i] << 5)) | ((X[5][i] >> 3));
X[6][i] = ((X[6][i] << 1)) | ((X[6][i] >> 7));
X[7][i] = ((X[7][i] << 2)) | ((X[7][i] >> 6));
}
// Output: X[i][SHARES], 0<=i<=7
```