PIPO: A Lightweight Block Cipher with Efficient Higher-Order Masking Software Implementations

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, Seoul, Republic of Korea
jskim@kookmin.ac.kr

[2] Department of Information Security, Cryptology, and Mathematics, Kookmin University, Seoul, Republic of Korea

[3] Division of IT Convergence Engineering, Hansung University, Seoul, Republic of Korea

[4] School of Cyber Security, Korea University, Seoul, Republic of Korea

[5] Department of Mathematics, University of Seoul, Seoul, Republic of Korea

[6] Department of Information Technology and Engineering, Jeonbuk National University, Jeonju, Republic of Korea

Abstract. In this paper, we introduce a new lightweight 64-bit block cipher PIPO (PIPO stands for "Plug-In" and "Plug-Out", representing its use in side-channel protected and unprotected environments, respectively.) supporting a 128 or 256-bit key. It is a byte-oriented and bitsliced cipher that offers excellent performance in 8-bit AVR software implementations. In particular, PIPO allows for efficient higher-order masking implementations, since it uses a minimal number of nonlinear operations. 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. Furthermore, PIPO records competitive round-based hardware implementations.

For the nonlinear layer of PIPO, we have developed a new lightweight 8-bit S-box that provides an efficient bitsliced implementation including only 11 nonlinear bitwise operations. Furthermore, its differential and linear branch numbers are both 3. This characteristic enables PIPO to thwart differential and linear attacks with fewer rounds. The 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

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).

1 Introduction

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 [22] and CLEFIA [44] have been standardized by ISO/IEC. Additionally, a lightweight cryptography standardization project is ongoing at NIST. A variety of lightweight block ciphers have been introduced in the literature, many of which are hardware-friendly systems [9,22,24,33,43]. Others focus on software performance [10,28,31] or both hardware and software performance [8,12,14,32,49].

In 1996, Paul Kocher first proposed side-channel attacks, which extract secret information by analyzing side-channel information [37]. 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 [47], the application of side-channel countermeasures to ciphers has become critical. 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, whose design goal is a low nonlinear operation count, have been proposed [2,28,31]. These are intended for use in side-channel protected environments.

However, most existing lightweight block ciphers are unsuitable for at least one type of hardware, software, or side-channel protected implementations in low resource environments. Consequently, it is challenging to design general-purpose lightweight block ciphers capable of operating in any resource-constrained environment.

In this paper, we propose a new lightweight versatile block cipher PIPO. During the PIPO design process, the focus was on minimizing the number of nonlinear operations because this is the most significant factor for efficient higher-order masking implementations. To construct an 8-bit S-box with a small number of nonlinear operations, we devised a unbalanced-Bridge structure that accepts one 3-bit and two 5-bit S-boxes and produces 8-bit S-boxes. This structure allows us to construct a new 8-bit S-box that offers good cryptographic properties and an efficient bitsliced implementation including only 11 nonlinear bitwise operations. We also present theorems applied to the unbalanced-Bridge structure, which show the conditions that the both differential and linear branch numbers of the S-boxes constructed through the structure are greater than 2. We investigated the linear layer with the highest security against differential and linear cryptanalyses when combined with the new S-box, through which we could reduce the number of rounds of PIPO. Consequently, PIPO achieves fast higher-order masking implementations, 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. Additionally, PIPO records excellent performance on 8-bit microcontrollers and competitive round-based hardware implementations.

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 Specification of PIPO

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. 1. 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. 1, 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. The S_8 is shown in Table 7 of Appendix C.1. The R-layer rotates the bits in each row by a given offset (Fig. 2).

The key schedule of PIPO is very simple. We first split a 128-bit 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, a 256-bit master key K is divided into four 64-bit subkeys K_0, K_1, K_2, and K_3, *i.e.*, $K = K_3 || K_2 || K_1 || K_0$. In this case, $RK_i = K_{i \ mod \ 4}$ where $i = 0, 1, 2, \cdots, 17$. 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.

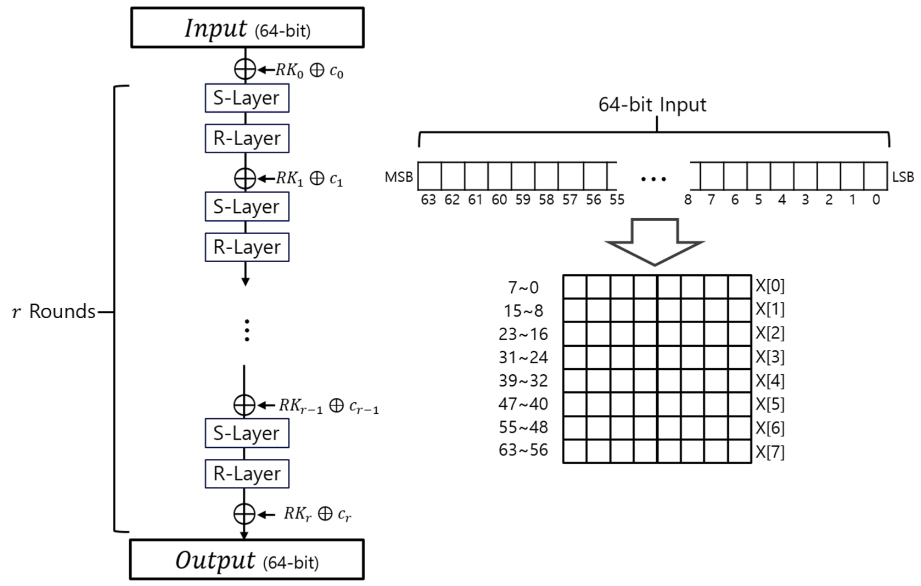

Fig. 1. Overall structure (left) and intermediate state (right) of PIPO

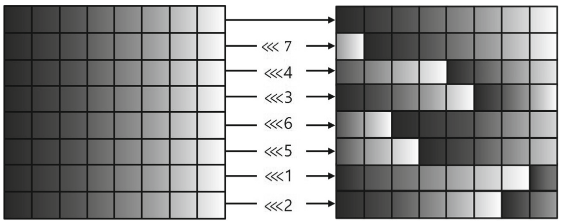

Fig. 2. R-layer

3 Design Rationale of PIPO

3.1 S-Layer

Overall Structure. We focused on the following three criteria when designing our 8-bit S-box, S_8.

1. It should offer an efficient bitsliced implementation including 11 or fewer nonlinear operations.
2. Its differential and linear branch numbers (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).

The Bridge structure was first proposed in [36], and revisited in [15]. In order to construct an S_8 satisfying all the aforementioned three criteria, we employed the unbalanced-Bridge structure depicted in Fig. 3, where S_i^j represents the j-th and i-bit S-box in the structure. This structure has the following three characteristics. First, it uses 3-bit and 5-bit S-boxes instead of 4-bit S-boxes. We observe that 8-bit S-box constructions using three 4-bit S-boxes would have difficulty satisfying criterion 1, even though they conform to criteria 2 and 3. Second, all eight output bits are 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.

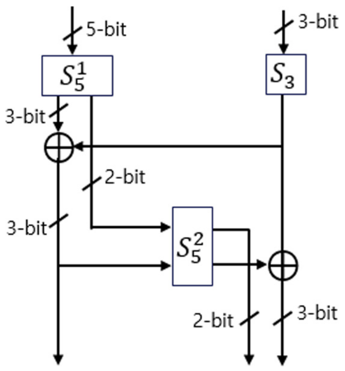

Fig. 3. The unbalanced-Bridge structure

The notation used in this section is introduced below.

$$\begin{aligned}
&\rho_c : \mathbb{F}_2^5 \to \mathbb{F}_2^5,\ \ \rho_c(x||y) = y||x,\ \ \text{for } x \in \mathbb{F}_2^3,\ y \in \mathbb{F}_2^2,\\
&\tau_n : \mathbb{F}_2^5 \to \mathbb{F}_2^n,\ \ \tau_n(x||y) = x,\ \ \text{for } x \in \mathbb{F}_2^n,\ y \in \mathbb{F}_2^{5-n},\ n \in \{1,2,3,4\},\\
&\tau'_n : \mathbb{F}_2^5 \to \mathbb{F}_2^n,\ \ \tau'_n(x||y) = y,\ \ \text{for } x \in \mathbb{F}_2^{5-n},\ y \in \mathbb{F}_2^n,\ n \in \{1,2,3,4\},\\
&\mathfrak{F}_A^1 : \mathbb{F}_2^3 \to \mathbb{F}_2^5,\ \ \mathfrak{F}_A^1(X) = (S_5^1)^{-1}(X||A)\ \ \text{for } A \in \mathbb{F}_2^2,\\
&\mathfrak{F}_A^2 : \mathbb{F}_2^3 \to \mathbb{F}_2^5,\ \ \mathfrak{F}_A^2(X) = S_5^2(X||A)\ \ \text{for } A \in \mathbb{F}_2^2.
\end{aligned}$$

Now we can define an 8-bit S-box constructed by the unbalanced-Bridge. 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 the 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 the 8-bit S-box constructed by the unbalanced-Bridge to be bijective.

Proposition 1. *The 8-bit S-box constructed using the unbalanced-Bridge 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* $f_y : \mathbb{F}_2^2 \rightarrow \mathbb{F}_2^2$.

Proof. Refer to Appendix B.1.

Construction of 8-Bit S-Boxes with DBN> 2 and LBN> 2 and Our S_8 Selection. We present here how to construct 8-bit S-boxes with DBN> 2 and LBN> 2. Our framework is to eliminate all the input and output differences (masks) where the sum of their Hamming weights is 2. During this elimination process, we can obtain some conditions of smaller S-boxes. Theorems 1 and 2 present the necessary and sufficient conditions of smaller S-boxes so that the 8-bit S-boxes constructed by Fig. 3 have both differential and linear branch numbers greater than 2.

Theorem 1. *The DBN of bijective 8-bit S-boxes constructed using the unbalanced-Bridge 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* $X \in \mathbb{F}_2^3$.

Proof. Refer to Appendix B.2.

The following theorem concerning the LBN can be similarly obtained.

Theorem 2. *The LBN of bijective 8-bit S-boxes constructed using the unbalanced-Bridge 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* $\mathfrak{F}_A^2$.

Proof. Refer to Appendix B.3.

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, Theorems 1 and 2. During this process, the search space for the S_8 could be 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-box satisfies criterion 3, and no fixed point. During the search, we found more than 8,000 candidates for the S_8. We selected the one that leads to the best resistance to differential and linear attacks when combined with the linear layer of PIPO (refer to section 3.2). The final selected input/output values of the S_8 are presented in Table 7; its bitsliced implementation is given in Appendix C.2.

Table 1 compares the cryptographic properties and operations with those of other 8-bit S-boxes built from smaller 3 S-boxes.

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

Blockcipher	PIPO	FLY	Fantomas	Robin	LILLIPUT
Differential uniformity	**16**	16	16	16	8
DBN	**3**	3	2	2	2
Non-linearity	**96**	96	96	96	96
LBN	**3**	3	2	2	2
Algebraic degree	**5**	5	5	6	6
#(Fixed points)	**0**	2	0	16	1
#(Nonlinear operations)	**11**	12	11	12	12
#(Linear operations)	**23**	24	27	24	27
Construction method	***U-Bridge**	Lai-Massey	*U-MISTY	MISTY	Feistel
Reference	**This paper**	[35]	[31]	[31]	[1]

*'U-' represents 'Unbalanced-'.

**Nonlinear (resp. linear) operations represent AND, OR (resp. XOR, NOT).

3.2 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. Its bitsliced implementation is given in Listing 1.1. During the design of the R-layer, the following criteria were considered.

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. 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}.

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

```
//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)
```

4 Security Evaluation of PIPO

Table 2 shows the maximum numbers of rounds of characteristics and key recovery attacks that we found for each attack [3,18–20,40,42,46]. In addition to the cryptanalysis shown in Table 2, we conducted algebraic attack [23], integral attack [48], statistical saturation attack [25], invariant subspace attack [38,39], nonlinear invariant attack [45] and slide attack [21], but they were not applied more effectively than DC or LC.

Table 2. 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 [41]) 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.

5 Performance Evaluation of Higher-Order Masking Implementations of PIPO

Bitsliced implementations, initially proposed by Biham [17], are known to be efficient when applying Boolean masking, since secure S-box computations can be carried out in parallel [29–31,34]. 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. There are several variations of ISW-AND [6,7,16], 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.

Table 3. Comparison of required ROM (bytes) for round constant, number of nonlinear bitwise operations, and permutation layers of round functions

Block cipher	Table size	#(nonlinear bitwise operations)	Permutation
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

We compare our proposed PIPO with 64-bit block ciphers with 128-bit keys as shown in Table 3. 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 3. Only CRAFT used an additional 16-byte diffusion table for generating tweakeys. The same secure logical operations of PIPO were applied to implement higher-order masking structures.

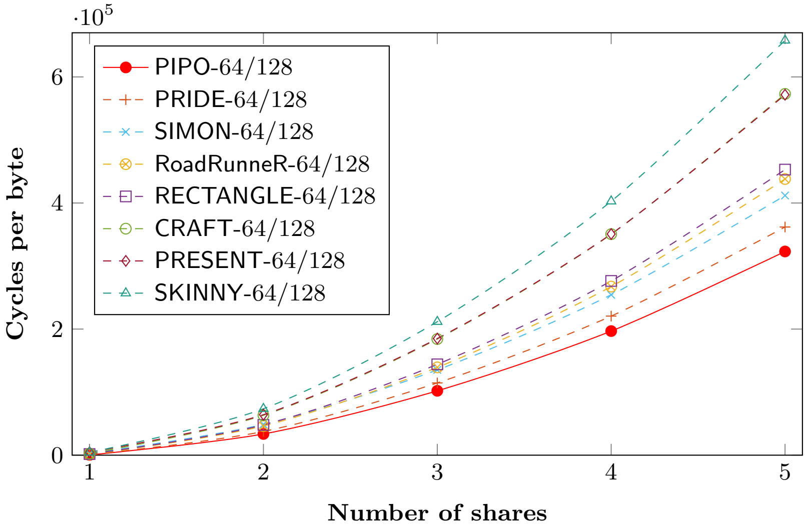

Fig. 4. Execution times of one-block encryptions according to the number of shares in an Atmel AVR XMEGA128 (1 means unprotected)

Figure 4 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 3. PIPO has the smallest number of nonlinear operations.

6 Performance Evaluation of Software and Hardware Implementations of PIPO

6.1 Software Implementations

The PIPO block cipher consists of permutation (R-layer) and S-box (S-layer) computations. The permutation routine is performed in 8-bit rotation operations, and 22 XOR, 6 AND, 5 OR, 1 COM and 24 MOV instructions are used to compute the S-box. 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.

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 4 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 8 MHz [4]. 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 4. 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 [26]	466	204	403	2.84
PRIDE-64/128 [26]	650	47	969	1.39
SKINNY-64/128 [26]	502	187	877	1.30
PRESENT-64/128 [27]	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.

6.2 Hardware Implementations

We implemented PIPO-64/128 and PIPO-64/256 in Verilog, and synthesized the proposed architectures using the Synopsys Design Compiler with 130 nm CMOS technology. Figure 5 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 and 17 clock cycles to encrypt a 64-bit plaintext with 128-bit and 256-bit keys, respectively.

Table 5 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. The flip-flops are used for storing plaintext and counter, and the other areas consist of MUX and other logical operations.

Table 6 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

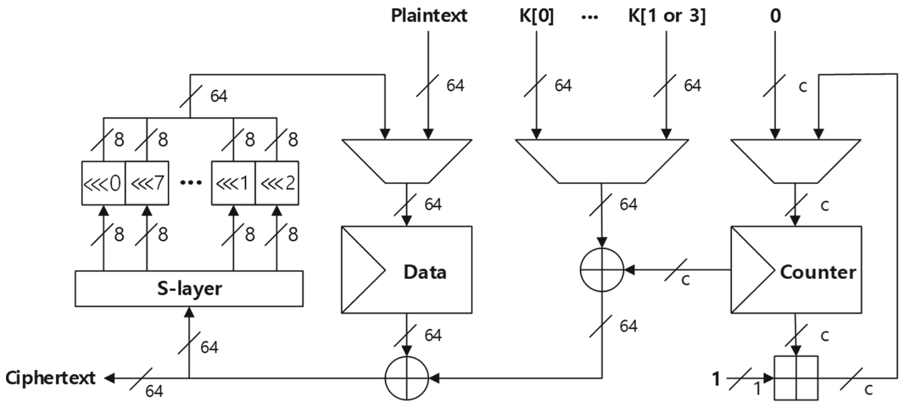

Fig. 5. Datapath of an area-optimized version of PIPO

Table 5. 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 6. Comparison of round-based and area optimized implementations for block ciphers using 130 nm 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 [43]	1,197	194	33	1,354
SIMON-64/128 [12]	1,417	133	48	664
RECTANGLE-64/128 [49]	2,064	246	26	578
PIPO-64/256	1,602	376	17	1,467

bits per clock cycle per GE squared [5,32]). It is obvious that the high FOM of PIPO requires less energy and battery consumption.

7 Conclusion

In this paper, we proposed a new lightweight versatile block cipher PIPO suitable for diverse resource-constrained environments. In particular, PIPO exhibits excellent 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.

A Test Vectors

The following test vectors are represented in big endian representation.

- 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 Proposition 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 $S_8(X_L||X_R) = S_8(X'_L||X'_R)$, 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 Theorem 1

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

$$\tau_3(Y) \oplus S_3(X_R) \oplus \tau_3(Y) \oplus S_3(X_R \oplus \Delta a) = S_3(X_R) \oplus S_3(X_R \oplus \Delta a) = \Delta d. \tag{1}$$

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

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

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. (1) and (2), 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

$$\begin{aligned}
(S_5^1)^{-1}(Y) \oplus (S_5^1)^{-1}(Y \oplus \Delta\omega) &= \Delta b, \\
(S_5^1)^{-1}(Y'||A) \oplus (S_5^1)^{-1}(Y'||(A \oplus \Delta\omega)) &= \Delta b.
\end{aligned} \tag{3}$$

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

$$S_5^2(Z'||A) \oplus S_5^2(Z'||(A \oplus \Delta\omega)) = \rho_c^{-1}(\Delta c). \quad (4)$$

For each A, the above Eqs. (3) and (4) are equivalent to

$$\mathfrak{F}_A^1(Y') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^1(Y') = \Delta b, \quad (5)$$

$$\mathfrak{F}_A^2(Z') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^2(Z') = \rho_c^{-1}(\Delta c). \quad (6)$$

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. (5) and (6) 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 $\Delta\omega \in \mathbb{F}_2^2$, the above equation becomes

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

As in Eq. (3), we obtain

$$(S_5^1)^{-1}(Y'||A) \oplus (S_5^1)^{-1}((Y' \oplus \Delta d)||(A \oplus \Delta\omega)) = \Delta b. \quad (7)$$

And the second equation is expressed as

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

Clearly,

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

It becomes

$$S_5^2(Z'||A) \oplus S_5^2((Z' \oplus \Delta d)||(A \oplus \Delta\omega)) = \Delta 0. \tag{8}$$

For each A, the above Eqs. (7) and (8) are equivalent to

$$\mathfrak{F}_A^1(Y') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^1(Y' \oplus \Delta d) = \Delta b, \tag{9}$$

$$\mathfrak{F}_A^2(Z') \oplus \mathfrak{F}_{A\oplus\Delta\omega}^2(Z' \oplus \Delta d) = \Delta 0. \tag{10}$$

Similarly to the case above, we define $B = A \oplus \Delta\omega$. In this time, B can be either A or not, since $\Delta\omega$ can be a zero difference. The $(\Delta b||0^{(3)}, \Delta d||0^{(5)})$ case does not happen if and only if there is no (Y', A, Z') satisfying both Eqs. (9) and (10) for all B, which is equivalent to condition *iii*) where $\Delta\alpha = \Delta d$, $\Delta\beta = \Delta b$. ■

B.3 Proof of Theorem 2

We use Y, Y', Z, Z', and A defined in proof B.2.

$\underline{(0^{(5)}||\lambda_a, 0^{(3)}||\lambda_c)}$: 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.

$\underline{(\lambda_b||0^{(3)}, 0^{(3)}||\lambda_c)}$: 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 $\tau_3'(\lambda_c) = 0$).

$\underline{(\lambda_b||0^{(3)}, \lambda_d||0^{(5)})}$: 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 8-bit S-box of PIPO, S_8

C.1 Table of the S_8

Table 7 shows the S_8.

Table 7. 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

C.2 Bitsliced Implementations of the S_8 and Its Inverse

Listing 1.2 is the bitsliced implementation of the S_8.[1] 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. 1, we can implement the S-layer by replacing bit $x[i]$ with byte $X[i]$ which represents the i-th row value, where $i = 0, 1, 2, \cdots, 7$.

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];
```

[1] For a higher resistance against DC and LC, swapping bits is additionally conducted in the S_8 design (refer to section 3.2).

```
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]
```

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]
```

References

1. Adomnicai, A., et al.: 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., Yalçın, T.: Block ciphers – focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 57–76. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_4
3. Aoki, K., Sasaki, Yu.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_7
4. Atmel Corporation, ATmega128(L) Datasheet. www.microchip.com/wwwproducts/en/ATmega128. Accessed 23 Apr 2019
5. Badel, S., et al.: ARMADILLO: a multi-purpose cryptographic primitive dedicated to hardware. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 398–412. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_27
6. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.-X., Strub, P.-Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 535–566. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_19
7. Battistello, A., Coron, J.-S., Prouff, E., Zeitoun, R.: Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 23–39. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_2
8. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Yu., Sim, S.M., Todo, Y.: GIFT: a small present. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_16
9. Banik, S., et al.: Midori: a block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 411–436. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_17
10. Baysal, A., Şahin, S.: RoadRunneR: a small and fast bitslice block cipher for low cost 8-bit processors. In: Güneysu, T., Leander, G., Moradi, A. (eds.) LightSec 2015. LNCS, vol. 9542, pp. 58–76. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29078-2_4
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. In: Eisenbarth, T., Öztürk, E. (eds.) LightSec 2014. LNCS, vol. 8898, pp. 3–20. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16363-5_1

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), 5–45 (2019)
14. Beierle, C., et al.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_5
15. Bilgin, B., De Meyer, L., Duval, S., Levi, I., Standaert, F.X.: Low AND depth and efficient inverses: a guide on s-boxes for low-latency masking. IACR Trans. Symmetric Cryptol. **2020**(1), 144–184 (2020)
16. Belaïd, S., Benhamouda, F., Passelègue, A., Prouff, E., Thillard, A., Vergnaud, D.: Randomness complexity of private circuits for multiplication. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 616–648. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_22
17. Biham, E.: A fast new DES implementation in software. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052352
18. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_2
19. Biham, E., Dunkelman, O., Keller, N.: The rectangle attack — rectangling the serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_21
20. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_1
21. Biryukov, A., Wagner, D.: Advanced slide attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 589–606. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_41
22. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
23. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of Keccak and *Luffa*. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_15
24. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_14
25. Collard, B., Standaert, F.-X.: A statistical saturation attack against the block cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00862-7_13
26. Dinu, D., Biryukov, A., Großschädl, J., Khovratovich, D., Corre, Y.L., Perrin, L.: FELICS-fair evaluation of lightweight cryptographic systems. In: NIST Workshop on Lightweight Cryptography (2015)

27. Engels, S., Kavun, E.B., Paar, C., Yalçin, T., Mihajloska, H.: A non-linear/linear instruction set extension for lightweight ciphers. In: IEEE 21st Symposium on Computer Arithmetic, pp. 67–75 (2013)
28. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.-X.: Block ciphers that are easier to mask: how far can we go? In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 383–399. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40349-1_22
29. Goudarzi, D., Journault, A., Rivain, M., Standaert, F.-X.: Secure multiplication for bitslice higher-order masking: optimisation and comparison. In: Fan, J., Gierlichs, B. (eds.) COSADE 2018. LNCS, vol. 10815, pp. 3–22. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89641-0_1
30. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 567–597. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_20
31. Grosso, V., Leurent, G., Standaert, F.-X., Varıcı, K.: LS-designs: bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_2
32. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_22
33. Hong, D., et al.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_4
34. Journault, A., Standaert, F.-X.: Very high order masking: efficient implementation and security evaluation. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 623–643. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_30
35. Karpman, P., Grégoire, B.: The littlun s-box and the fly block cipher. In: Lightweight Cryptography Workshop (2016)
36. Kim, J., Lee, C., Sung, J., Hong, S., Lee, S., Lim, J.: Seven new block cipher structures with provable security against differential cryptanalysis. IEICE Trans. **91-A**(10), 3047–3058 (2008)
37. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
38. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A cryptanalysis of PRINTcipher: the invariant subspace attack. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 206–221. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_12
39. Leander, G., Minaud, B., Rønjom, S.: A generic approach to invariant subspace attacks: cryptanalysis of Robin, iSCREAM and Zorro. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 254–283. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_11
40. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_33
41. Matsui, M.: On correlation between the order of S-boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0053451

42. Sasaki, Yu., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_8
43. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: *Piccolo*: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_23
44. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74619-5_12
45. Todo, Y., Leander, G., Sasaki, Y.: Nonlinear invariant attack - practical attack on full SCREAM, iSCREAM, and Midori64. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 3–33. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_1
46. Wagner, D.: The boomerang attack. In: Knudsen, L. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48519-8_12
47. Worthman, E.: ChaoLogix: integrated security. Semiconductor Eng. (2015)
48. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-pattern based integral attack. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 363–381. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71039-4_23
49. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)