Enhancing the Related-Key Security of PIPO through New Key Schedules*

Seungjun Baek[1], Giyoon Kim[1], Yongjin Jeon[1], and Jongsung Kim[1,2]

[1] Department of Financial Information Security, Kookmin University, Republic of Korea
{hellosj3,gi0412,idealtop18,jskim}@kookmin.ac.kr
[2] Department of Information Security, Cryptology, and Mathematics, Kookmin University, Republic of Korea

Abstract. In this paper, we present new key schedules for the PIPO block cipher that enhance its security in the related-key setting. While PIPO has demonstrated noteworthy resistance against attacks in the single-key setting, its security in the related-key setting is very vulnerable owing to its simple key schedule. Given the lightweight property of PIPO, we tweak the key schedule algorithm of PIPO by applying computation only within a single register or from one register to another in key states. By adopting our new key schedules, the tweaked version of PIPO achieves better resistance to related-key attacks and demonstrates competitive implementation results in an 8-bit AVR environment. We expect that this paper will contribute to a better understanding of the PIPO block cipher.

Keywords: symmetric-key cryptography · PIPO · block cipher · related-key attacks · key schedule

1 Introduction

Plug-In Plug-Out (PIPO) [11], proposed at ICISC 2020, is a lightweight block cipher with a substitution permutation network (SPN) structure that supports 64-bit block size and 128- and 256-bit keys. PIPO was designed to be suitable for the AVR embedded processor, which is a typical 8-bit microcontroller. PIPO-128 achieved the highest speed in an 8-bit AVR environment among lightweight block ciphers such as SIMON [3], CRAFT [5], PRIDE [1], and RECTANGLE [19]. PIPO is also a block cipher standard that was approved by the Telecommunications Technology Association (TTA) of Korea in 2022 [16]. Since PIPO was developed, its security has been scrutinized by several cryptographers, and its full-round security has not yet been broken in the single-key setting.

* 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).

In designing a lightweight block cipher, related-key attacks are often dismissed because, from a practical perspective, they are unlikely to occur. Nevertheless, a block cipher vulnerable to a related-key attack presents some security concerns. It may not be suitable for other cryptographic primitives that use block ciphers as building blocks, e.g., block cipher-based hash functions. A concrete real-world example is the use of a hash function based on the block cipher TEA [10]. Microsoft's Xbox architecture employed a Davies–Meyer hash function instantiated with TEA, and a security vulnerability related to related-key characteristics of TEA was exploited in a hacking [18]. Another security concern arises when secret keys are frequently updated in protocols or when differences can be incorporated through fault attacks.

Recently, several analyses [17,14] of related-key characteristics for PIPO have been proposed. In these analyses, researchers have reported PIPO's full-round characteristics based on iterative characteristics with a high probability. This weakness is attributed to PIPO's simple key schedule. Given that cryptographers repeatedly analyze the related-key security of PIPO, enhancing its resistance against related-key attacks might give them confusions. Furthermore, considering that PIPO is designed for embedded processors, it could also be employed to construct a hash function, which motivates us to scrutinize its security in the context of related-key setting.

Our Contributions. In this paper, we propose tweaks to the key schedule algorithm of PIPO-128. We take into account two conditions for tweaking PIPO-128's key schedule. First, our proposed tweaks must ensure better related-key security than the original PIPO-128. This is achieved by rotating the registers of key states in the key schedule algorithm to break the 2-round iterative related-key differential characteristics that occur. We also add additional bit-rotation within a register to further improve security. Second, we strive to ensure that the tweaked PIPO algorithm has minimal overhead in an 8-bit AVR environment. To inherit the lightweight nature of PIPO-128 while keeping implementation cost low, we completely exclude nonlinear operators, such as AND or OR gates, in the proposed tweaks. Instead, we mainly apply computation within a single register or from one register to another.

We evaluate the related-key security of our tweaks in terms of the number of active S-boxes in a characteristic. We first construct a Mixed Integer Linear Programming (MILP) model for PIPO-128 and evaluate the number of active S-boxes. Comparing our tweak to the original PIPO-128, we achieve more than twice the number of active S-boxes in characteristics with large rounds. For example, the 10-round characteristic of the original PIPO-128 had five active S-boxes, while ours has 11. While this measurement may not yield the characteristic with the lowest probability, it is sufficient to demonstrate the related-key security of PIPO-128. We also examine the implementation efficiency of our tweaks in an 8-bit AVR environment. Even though our tweaks involve slightly more computation compared to the original PIPO-128, their overhead is minimal. Thus, we

preserve the lightweight property of PIPO-128. Furthermore, we confirm that our tweaks are useful for PIPO-256 as well.

Paper Organization. Section 2 describes the specifications of PIPO and related-key differential attacks. Section 3 describes our new tweaks for PIPO's key schedule. Section 4 describes security analysis for the tweaked PIPO in the related-key setting. Section 5 describes our implementation results for the tweaked PIPO in an 8-bit AVR environment. Section 6 presents our conclusion.

2 Preliminaries

2.1 Description of PIPO

Figure 1 depicts the process of PIPO [11,12]. The internal state of PIPO is represented by an 8×8 bit matrix. In the bit matrix, the least significant bit (LSB) is located at the top right and is filled from right to left. When one row is filled, the next row is filled again from the right.

The plaintext is XORed with the whitening key and then undergoes a sequence of r rounds. For PIPO-128, r is 13, while for PIPO-256, r is 17. Each round consists of three layers: S-layer, R-layer, and round key and constant XOR additions.

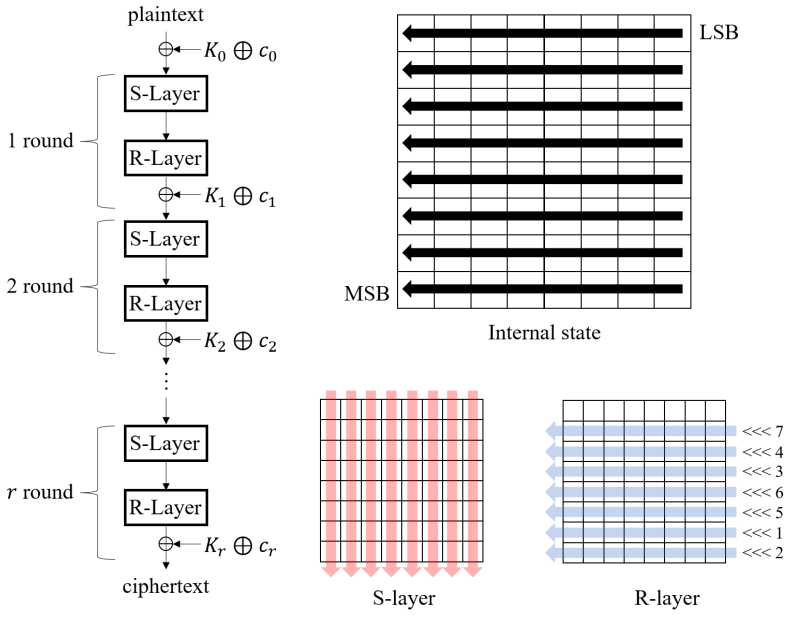

Fig. 1. Description of PIPO

S-Layer (SL). PIPO uses the defined 8-bit S-box as shown in Table 1. Each column in the state is independently substituted using eight S-boxes. The top bit of the state becomes the LSB of the S-box input value.

Table 1. PIPO S-box

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
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

R-Layer (RL). RL rotates each row of the state to the left. The rotation values from the top row to the bottom row are 0, 7, 4, 3, 6, 5, 1, and 2, respectively.

Round Key and Constant XOR Additions. This layer XORs round constants and the round keys to the internal state. We denote the i-th round key as K_i. We also denote the j-th row of K_i is k_j^i, i.e., $K_i = k_7^i||k_6^i||\cdots||k_0^i$. In PIPO, there is a whitening key, and we treat it as the 0-th round key K_0.

c_i is the i-th round constant, defined as $c_i = i$. This definition includes the case of $i = 0$ (i.e., $c_0 = 0$). Since c_i cannot be higher than 19, the constant XOR addition only affects the 0-th row of the internal state.

Key Schedule. For PIPO-128, the master key MK is split into two 64-bit states and used alternately (see Figure 2). Let $MK = MK_1||MK_0$ for 64-bit values MK_0 and MK_1. The i-th round key K_i is defined by $K_i = MK_{i \pmod 2}$.

For PIPO-256, the master key MK is split into four 64-bit states and used in sequence. That is, $K_i = MK_{i \pmod 4}$ where $MK = MK_3||MK_2||MK_1||MK_0$ for 64-bit values MK_0, MK_1, MK_2, and MK_3.

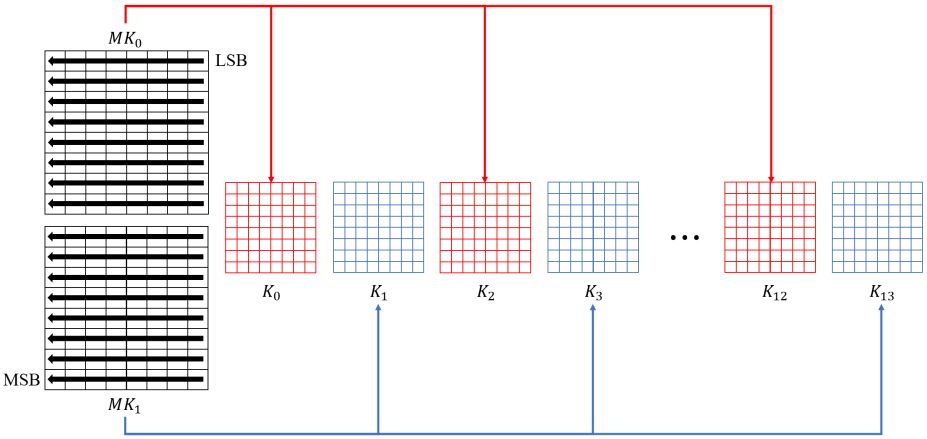

Fig. 2. Key schedule of PIPO-128

2.2 Related-Key Differential Attack

Related-key attack, independently introduced by Biham [6] and Knudsen [13], is a powerful cryptanalytic tool for the analysis of block ciphers. In this attack, the adversary can obtain the encryption of plaintexts under several related keys, where the relationship between the keys is known to (or can be chosen by) the adversary. Kelsey et al. [9] introduced the related-key differential attack. The adversary can ask for the encryption of plaintext pairs with a chosen difference of α, using unknown keys that have a difference of ΔK in a manner that is known or chosen by the adversary. To attack an n-bit cipher, the adversary exploits a related-key differential characteristic $\alpha \rightarrow \beta$ for target (sub-)cipher E with a probability p larger than 2^{-n}, i.e.,

$$Pr_{(P,K)}[E_K(P) \oplus E_{K \oplus \Delta K}(P \oplus \alpha) = \beta] = p > 2^{-n},$$

where P represents a plaintext. Here, the adversary's task is to find a related-key characteristic with as high a probability as possible. This attack is based on the key schedule and on the encryption/decryption algorithms, so a cipher with a weak key schedule may be vulnerable to this kind of attack.

3 New Key Schedules of PIPO

In this section, we propose new key schedules of PIPO to enhance the security in the related-key setting. We first observe the existing iterative related-key differential characteristics for PIPO-128 and PIPO-256. Here, we omit the constant addition, as it is not relevant to our analysis.

3.1 Related-Key differential characteristics of PIPO

Yadav and Kumar [17] showed a 2-round iterative related-key differential characteristic with probability 2^{-4} and constructed a full-round characteristic with probability 2^{-24} for PIPO-128. Soon after, Sun and Wang [14] reported full-round differential characteristics of PIPO-256 for the first time. Due to the simple key schedule of PIPO, we can construct several related-key differential characteristics containing only a few active S-boxes. Concretely, 2-round iterative related-key differential characteristics can be found straightforwardly (see Figure 3).

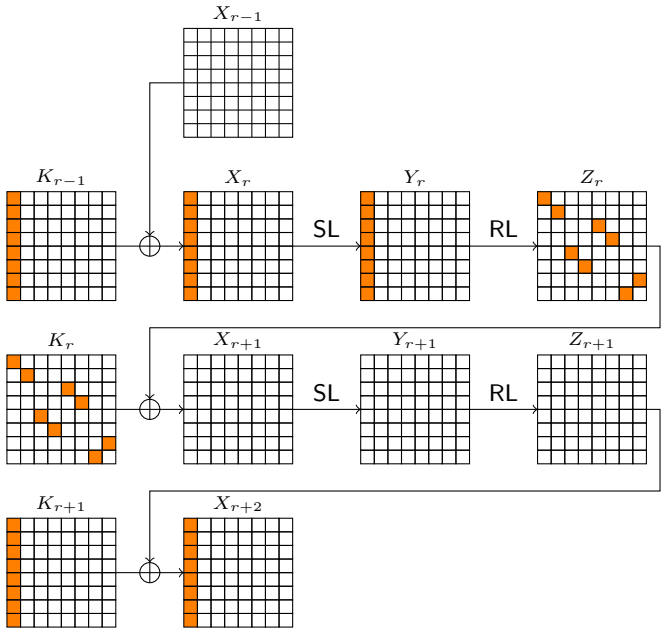

Fig. 3. 2-round iterative related-key differential characteristics of PIPO-128

In the transition $X_r \xrightarrow{\mathsf{SL}} Y_r \xrightarrow{\mathsf{RL}} Z_r$, the 2-round characteristic is constructed by setting ΔX_r and ΔZ_r as iterative keys. Considering the differential distribution table (DDT) of PIPO, there are 224 entries with probability 2^{-4} (see Table 2). Since the difference of X_r can also be placed in the remaining seven columns, there are a total of $224 \times 8 = 1792$ characteristics.

Table 2. Distribution of non-trivial probabilities in DDT of PIPO's S-box

DDT value	2	4	6	8	10	12	16
# of entries	12552	6226	651	951	9	7	224
probability	2^{-7}	2^{-6}	$2^{-5.415}$	2^{-5}	$2^{-4.678}$	$2^{-4.415}$	2^{-4}

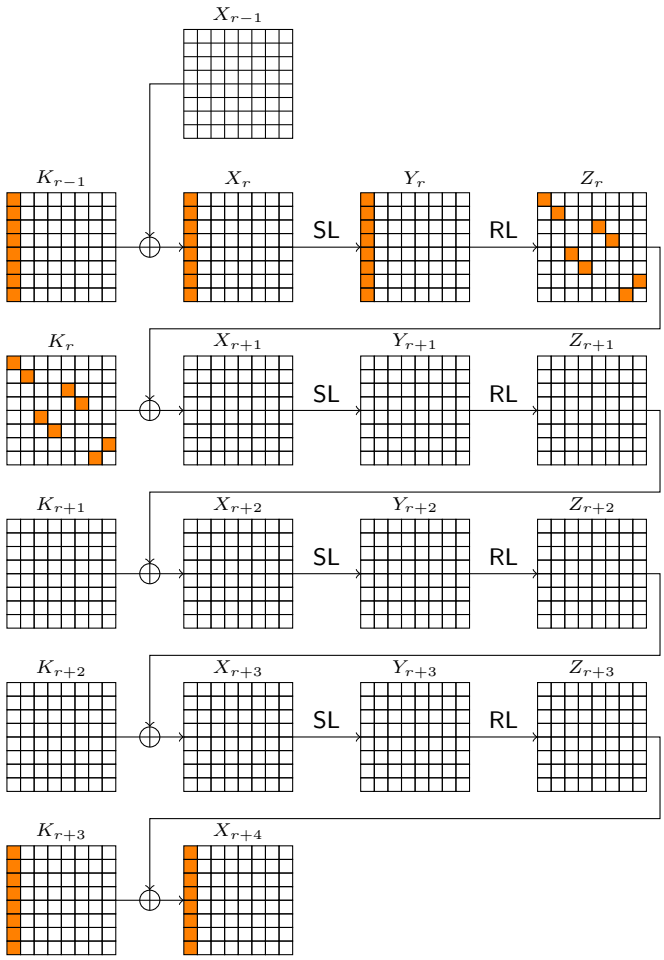

Fig. 4. 4-round iterative related-key differential characteristics of PIPO-256

Similarly, there exists a full-round differential characteristic with probability 2^{-16} for PIPO-256 based on the 4-round iterative differential characteristic (see Figure 4).

3.2 Introducing New Key Schedules: KS1 and KS2

We propose new key schedules for PIPO-128. There are two factors to consider in order to simultaneously satisfy the related-key security and implementation efficiency of our key schedules. Note that we are not considering changing the entire algorithm of PIPO-128; we are only changing the key schedule. Our considerations for these tweaks are as follows:

1. **Increasing resistance against related-key differential attacks** - In PIPO-128's key schedule, the master key is divided into two 64-bit key states, and the attacker determines the difference of the two states by selecting the difference of the master key. Given this simple key schedule, the initially selected difference remains fixed within the two key states and is XORed throughout the entire algorithm every two rounds. Ultimately, there are 2-round iterative differential characteristics resulting from this property, so our main goal is to prevent such characteristics from occurring. To increase resistance against related-key differential attacks, we induce diffusion within the key schedule in the column-wise as well as row-wise directions. Specifically, we measure the minimum number of active S-boxes using the MILP tool. This number enables us to establish the bounds on the probability of differential characteristics, considering the best probability of 2^{-4} from the PIPO S-box's DDT table.
2. **Achieving minimal overhead** - When considering tweaks to the key schedule, one might choose to apply various operators to induce diffusion of differences within key states. Recall that PIPO-128 is a block cipher optimized for 8-bit microcontrollers, so it primarily relies on computations in terms of register level throughout the encrypting/decrypting process. To inherit this advantage, we strictly limit our key schedule tweaks to computing within a single register or from one register to another. Specifically, to preserve the low implementation cost, we completely exclude nonlinear operators such as AND or OR gates. Finally, we tweak the key schedule in a way that ensures security while minimizing the overhead in an 8-bit AVR environment.

Now we introduce two new key schedules of PIPO-128, which we refer to as KS1 and KS2. We refer to the original key schedule of PIPO-128 as KS0. The evaluation of their related-key security is discussed in Section 4.

KS1. KS1 is our first proposal for PIPO-128's key schedule. Our aim is to eliminate the 2-round iterative characteristics of PIPO-128 (see Figure 3) in KS1. To do this, we simply rotate each row register within each of the two key states by 1 in the upward direction. For the first two rounds, two key states are input as MK_0 and MK_1, but from then on, rotation is applied to each register every two rounds. If we apply this operation to the key schedule, a 2-round iterative pattern is easily broken due to the RL of PIPO-128. In Figure 5, we describe one example demonstrating our claim. We distinguish the differences between two key states: one represented by the color orange and the other by the color blue. In addition, we use hatch patterns to denote all possible differences. Here, we can see that the difference cancellation does not occur in the transition $X_2 \xrightarrow{\mathsf{SL}} Y_2 \xrightarrow{\mathsf{RL}} Z_2 \rightarrow X_3$. This is because the possible differences caused by ΔX_2 are not canceled out, mainly due to the rotation of the key state. In this way, difference cancellation patterns are prevented by applying rotations of registers. Therefore, KS1 can amplify the diffusion of key differences more than the original

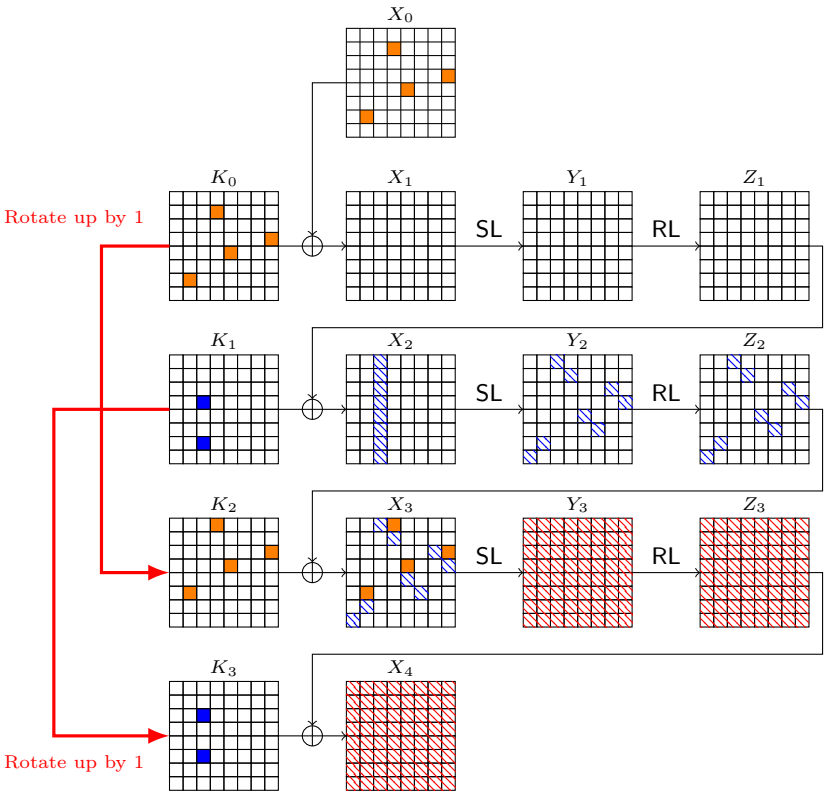

Fig. 5. Breaking the occurrence of the 2-round iterative characteristic of PIPO-128

one. KS1 is represented as follows:

$$(K_{r-1},K_r) = (k_7^{r-1}||k_6^{r-1}||\cdots||k_0^{r-1}, k_7^r||k_6^r||\cdots||k_0^r)$$
$$\xrightarrow{2\ rounds} (K_{r+1}, K_{r+2}) = (k_0^{r+1}||k_7^{r+1}||\cdots||k_1^{r+1}, k_0^{r+2}||k_7^{r+2}||\cdots||k_1^{r+2}).$$

In the rotation of two key states, one may consider rotating or changing only a few registers in each state. Let the attacker choose a key difference in the unchanged registers in one key state, typically one bit, and then set the difference determined by the RL operation in the unchanged registers in the other state. Since the differences in the unchanged registers are fixed, a 2-round characteristic occurs repeatedly. This is not a desirable proprety for us, so we do not adopt this method.

KS2. While KS1 offers better related-key security compared to KS0, there is still room for further improvement in security. The focus of KS2 is to improve the related-key security of KS1 by applying bit-rotation to one register in each key state. In KS1, there is no row-wise directional diffusion of the key difference,

allowing us to consider bit-rotation within the registers. Since our goal is to minimize overhead in an 8-bit AVR environment, we consider the optimized 8-bit rotation operations presented in [11] based on [7]. We mainly consider 1-bit and 4-bit left rotations, which require 2 and 1 clock cycles, respectively (see Table 3). The remaining bit-rotation operations require 3 to 5 clock cycles, so we do not take into account other cases. We also apply bit-rotation to only one upper register to keep the implementation efficient. Surprisingly, according to our examinations, applying 4-bit rotation yields better results than 1-bit rotation, even though 4-bit rotation is less expensive. Thus, we adopt the 4-bit rotation for `KS2`. `KS2` is represented as follows:

$$(K_{r-1}, K_r) = (k_7^{r-1}||k_6^{r-1}||\cdots||k_0^{r-1}, k_7^r||k_6^r||\cdots||k_0^r)$$
$$\xrightarrow{2\ rounds} (K_{r+1}, K_{r+2}) = (k_0^{r+1}||k_7^{r+1}||\cdots||(k_1^{r+1} \lll 4), k_0^{r+2}||k_7^{r+2}||\cdots||(k_1^{r+2} \lll 4)).$$

Table 3. 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

4 MILP-based Search for Related-Key Characteristics for PIPO with New Key Schedules

Now, we present a security analysis for new key schedules for PIPO-128. We adopt the MILP framework for bit-oriented ciphers proposed by Sun et al. [15] and describe the MILP model for PIPO-128. To optimize the model, we use the Gurobi MILP solver. We utilized the MILES tool [17] for generating linear inequalities of the PIPO-128 S-box. Finally, we apply our MILP model to search for related-key characteristics for PIPO-128 with new key schedules and present the results. We also present some results for PIPO-256.

4.1 MILP model for PIPO

Generating Linear Inequalities of S-box. Yadav and Kumar [17] proposed the MILES tool to minimize the number of linear inequalities for large S-boxes. Minimizing the number of inequalities directly affects the efficiency of MILP model. Thus, we utilize the MILES tool to generate linear inequalities of the PIPO S-box. As described in [17], we obtain 4474 linear inequalities for the S-box, and 35792 inequalities are needed for the SL of one round of PIPO.

Variables and Constraints. We represent the difference of all cells in each round as a set of binary variables x_i. Each variable x_i can take on values 0 or 1, signifying inactive and active bits, respectively. The binary variables $x_0, x_1, \cdots, x_{63}$ represent a 64-bit plaintext difference, and the difference for the next round state is updated as $x_{64}, x_{65}, \cdots, x_{127}$, and so on. To reduce the number of variables in the MILP model, the output bits of SL in the first round are set to the variables in the next round with RL^{-1} applied, and the process is repeated for each subsequent round. This process in the first round is represented as follows:

$$\begin{bmatrix} x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \\ x_{15} & x_{14} & x_{13} & x_{12} & x_{11} & x_{10} & x_9 & x_8 \\ x_{23} & x_{22} & x_{21} & x_{20} & x_{19} & x_{18} & x_{17} & x_{16} \\ x_{31} & x_{30} & x_{29} & x_{28} & x_{27} & x_{26} & x_{25} & x_{24} \\ x_{39} & x_{38} & x_{37} & x_{36} & x_{35} & x_{34} & x_{33} & x_{32} \\ x_{47} & x_{46} & x_{45} & x_{44} & x_{43} & x_{42} & x_{41} & x_{40} \\ x_{55} & x_{54} & x_{53} & x_{52} & x_{51} & x_{50} & x_{49} & x_{48} \\ x_{63} & x_{62} & x_{61} & x_{60} & x_{59} & x_{58} & x_{57} & x_{56} \end{bmatrix} \xrightarrow[\text{1-round}]{\mathsf{SL}} \begin{bmatrix} x_{71} & x_{70} & x_{69} & x_{68} & x_{67} & x_{66} & x_{65} & x_{64} \\ x_{78} & x_{77} & x_{76} & x_{75} & x_{74} & x_{73} & x_{72} & x_{79} \\ x_{83} & x_{82} & x_{81} & x_{80} & x_{87} & x_{86} & x_{85} & x_{84} \\ x_{90} & x_{89} & x_{88} & x_{95} & x_{94} & x_{93} & x_{92} & x_{91} \\ x_{101} & x_{100} & x_{99} & x_{98} & x_{97} & x_{96} & x_{103} & x_{102} \\ x_{108} & x_{107} & x_{106} & x_{105} & x_{104} & x_{111} & x_{110} & x_{109} \\ x_{112} & x_{119} & x_{118} & x_{117} & x_{116} & x_{115} & x_{114} & x_{113} \\ x_{121} & x_{120} & x_{127} & x_{126} & x_{125} & x_{124} & x_{123} & x_{122} \end{bmatrix}$$

Here, we construct linear inequalities based on the input and output variables of SL. Since SL is applied column-wise, the linear inequalities are also constructed in such a manner.

To search a related-key differential characteristics, we represent the difference of the master key as a set of binary variables k_i. For PIPO-128, its 128-bit key is represented by $k_0, k_1, \cdots, k_{127}$ and for PIPO-256, the 256-bit key is represented by $k_0, k_1, \cdots, k_{255}$.

In our model, the XOR operation of the difference is used for XORing the internal state and the key to generate a new internal state. x_{in} and k_{in} are the state bit and corresponding key bit, respectively, and x_{out} is the corresponding output bit. The following inequalities are used to describe the XOR operation:

$$\begin{cases} x_{in} + k_{in} - x_{out} \geq 0, \\ x_{in} - k_{in} + x_{out} \geq 0, \\ -x_{in} + k_{in} + x_{out} \geq 0, \\ x_{in} + k_{in} + x_{out} \leq 2. \end{cases}$$

In addition, we use the following set of the inequalities to check the number of active S-boxes of a characteristic:

$$\begin{cases} x_{64\cdot(r-1)+i} + x_{64\cdot(r-1)+i+8} + x_{64\cdot(r-1)+i+16} + x_{64\cdot(r-1)+i+24} \\ +x_{64\cdot(r-1)+i+32} + x_{64\cdot(r-1)+i+40} + x_{64\cdot(r-1)+i+48} + x_{64\cdot(r-1)+i+56} - a_{(r,i)} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+8} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+16} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+24} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+32} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+40} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+48} \geq 0, \\ a_{(r,i)} - x_{64\cdot(r-1)+i+56} \geq 0, \end{cases}$$

where $a_{(r,i)}$ denotes whether the i-th column from the right is active.

Objective Function. Our goal is to minimize the number of active S-boxes of a characteristic. Thus, when finding a r-round characteristic, our objective function is

$$\text{Minimize} \sum_{\text{Round } 1} a_{(1,i)} + \sum_{\text{Round } 2} a_{(2,i)} + \cdots + \sum_{\text{Round } r} a_{(r,i)}.$$

4.2 Results

We apply our MILP model to PIPO-128 with `KS0`, `KS1`, and `KS2`. Due to the large search space, we only compare these results up to 10 rounds of PIPO-128. In the case of `KS2` in round 10, we were unable to prove that this is the best result since the MILP solver did not terminate within a reasonable amount of time. We imposed a one-month time constraint for this case and ran the solver. Our results are summarized in Table 4.

We can observe that in rounds 1 to 2, the results for three key schedules are identical since the first two key states are the same as K_0 and K_1. The change occurs starting from round 5, which is due to the differential diffusion resulting from additional operations on key states. In particular, in `KS0`, there are rounds where active S-boxes do not exist every two rounds, whereas, in `KS1` and `KS2`, after round 3, there is at least one active S-box in each round. In comparing `KS0` and `KS1`, the difference in the number of active S-boxes begins to appear from round 9, and considering the results up to round 10, this difference is expected to increase as the number of rounds increases. This difference is more pronounced when comparing `KS0` and `KS2`. Furthermore, even if the number of active S-boxes in round 10 of `KS2` may not be optimal, we need to consider at least three additional active S-boxes to reach a full-round PIPO. Thus, by adopting `KS2` as the key schedule for the tweaked version of PIPO, we expect that there will be no related-key differential characteristics with a probability higher than 2^{-64}.

Table 4. Comparison of related-key differential characteristics for PIPO-128 according to KS0, KS1, and KS2

Round	KS0		KS1		KS2	
	#(Active S-box)	$-\log_2 p$	#(Active S-box)	$-\log_2 p$	#(Active S-box)	$-\log_2 p$
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	4	1	4	1	4
4	2	8	2	8	2	8
5	2	8	3	13	3	13
6	3	12	4	22.415	4	23
7	3	12	5	29	5	30
8	4	16	6	33.415	6	36.415
9	4	16	7	38.415	8	46.415
10	5	20	8	47.415	11*	60.830

KS0 represents the original key schedule of PIPO-128.
*Number of active S-boxes are not confirmed to be optimal.

On the Results for PIPO-256. We try to apply the approach of the key schedule KS1 to PIPO-256, and we refer to it as KS1*. That is, we simply rotate each row register within each of the four key states by 1 in the upward direction. In the same way as with PIPO-128, four key states are input as MK_0, MK_1, MK_2, and MK_3 in the first four rounds. As a result, we see that even when KS1* is adopted for PIPO-256, we can achieve better related-key security than the original key schedule (see Table 5).

We also attempted to apply the approach of KS2 to PIPO-256. However, due to the larger search space, the MILP solver did not termininate after 14 rounds. Furthermore, the results are either the same as or inferior to those obtained using KS1. Thus, we only present the results adopting KS1*.

5 Implementations

In this section, we compare our implementation results with the original PIPO-128 and other lightweight block ciphers. We used Atmel Studio 6.2 and compiled all implementations with optimization level 3. Our target processor was an ATmega128 running at 8 MHz, as in [11]. Since we could not find a reference assembly code for PIPO-128, we developed the code and analyzed it for a fair comparison. We also adopted a metric to measure overall performance on low-end devices, RANK, which is calculated as

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

where the code size represents ROM. Table 6 compares results for PIPO-128 on 8-bit AVR environment according to key schedules. Results for other block ciphers can be found in [11].

Table 5. Comparison of related-key differential characteristics for PIPO-256 according to KS0*, KS1*

Round	KS0*		KS1*	
	#(Active S-box)	$-\log_2 p$	#(Active S-box)	$-\log_2 p$
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	1	4	1	4
6	1	4	1	4
7	1	4	1	4
8	2	8	2	8
9	2	8	3	13
10	2	8	4	20.415
11	2	8	4	22
12	3	12	6	31.415
13	3	12	8	39
14	3	12	10	51.245
15	3	12	11	60.660
16	4	16	12	67

We refer to the original key schedule of PIPO-256 as KS0.

Table 6. Comparison of PIPO-128 on 8-bit AVR according to key schedules with other lightweight block ciphers

Block cipher	Code size (bytes)	RAM (bytes)	Execution time (cycles per byte)	RANK
PIPO-64/128(KS0)	354	31	197	12.09
SIMON-64/128 [3]	290	24	253	11.69
PIPO-64/128(KS1)	354	31	249	8.85
PIPO-64/128(KS2)	354	31	251	8.78
RoadRunneR-64/128 [2]	196	24	477	8.59
RECTANGLE-64/128 [19]	466	204	403	2.84
PRIDE-64/128 [1]	650	47	969	1.39
SKINNY-64/128 [4]	502	187	877	1.30
PRESENT-64/128 [8]	660	280	1,349	0.61
CRAFT-64/128 [5]	894	243	1,504	0.48

We also implemented PIPO-256 with KS0* and KS1* in the same environment. Both cases require the same code size and RAM: 354 bytes of code and 47 bytes of RAM. With regard to the execution time, PIPO-256 with KS0* requires 253 CPB, whereas with KS1*, it requires 321 CPB. Therefore, the RANK metrics for them are 8.82 and 6.95, respectively.

6 Conclusion

In this paper, we presented two new key schedules, KS1 and KS2, for the PIPO block cipher, aiming to enhance PIPO's related-key security. By applying KS1 and KS2 to PIPO-128, we achieved better related-key security compared to original PIPO-128. We also applied KS1 to PIPO-256 and obtained interesting results regarding security. We obtained comparative implementation results in an 8-bit AVR environment by completely excluding nonlinear operators and only applying computation within a single register or from one register to another. The significance of this study lies in enhancing related-key security of PIPO without significantly increasing the implementation cost.

References

1. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçin, T.: Block ciphers - focus on the linear layer (feat. PRIDE). In: CRYPTO 2014. Lecture Notes in Computer Science, vol. 8616, pp. 57–76. Springer (2014), https://doi.org/10.1007/978-3-662-44371-2_4
2. Baysal, A., Sahin, S.: Roadrunner: A small and fast bitslice block cipher for low cost 8-bit processors. In: LightSec 2015. Lecture Notes in Computer Science, vol. 9542, pp. 58–76. Springer (2015), https://doi.org/10.1007/978-3-319-29078-2_4
3. 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: LightSec 2014. Lecture Notes in Computer Science, vol. 8898, pp. 3–20. Springer (2014), https://doi.org/10.1007/978-3-319-16363-5_1
4. 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. In: CRYPTO 2016. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016), https://doi.org/10.1007/978-3-662-53008-5_5
5. 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), https://doi.org/10.13154/tosc.v2019.i1.5-45
6. Biham, E.: New types of cryptanalytic attacks using related keys. J. Cryptol. **7**(4), 229–246 (1994), https://doi.org/10.1007/BF00203965
7. Corporation, A.: Atmega128(l) datasheet (Visited on October 4, 2023), www.microchip.com/wwwproducts/en/ATmega128
8. Engels, S., Kavun, E.B., Paar, C., Yalçin, T., Mihajloska, H.: A non-linear/linear instruction set extension for lightweight ciphers. In: 21st IEEE Symposium on Computer Arithmetic, ARITH 2013, Austin, TX, USA, April 7-10, 2013. pp. 67–75. IEEE Computer Society (2013), https://doi.org/10.1109/ARITH.2013.36

9. Kelsey, J., Schneier, B., Wagner, D.A.: Key-schedule cryptanalysis of idea, g-des, gost, safer, and triple-des. In: CRYPTO '96. Lecture Notes in Computer Science, vol. 1109, pp. 237–251. Springer (1996), https://doi.org/10.1007/3-540-68697-5_19
10. Kelsey, J., Schneier, B., Wagner, D.A.: Related-key cryptanalysis of 3-way, biham-des, cast, des-x, newdes, rc2, and TEA. In: ICICS'97. Lecture Notes in Computer Science, vol. 1334, pp. 233–246. Springer (1997), https://doi.org/10.1007/BFb0028479
11. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B., Han, D., Seo, H., Kim, S., Hong, S., Sung, J., Hong, D.: PIPO: A lightweight block cipher with efficient higher-order masking software implementations. In: ICISC 2020. Lecture Notes in Computer Science, vol. 12593, pp. 99–122. Springer (2020), https://doi.org/10.1007/978-3-030-68890-5_6
12. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B., Han, D., Seo, H., Kim, S., Hong, S., Sung, J., Hong, D.: A new method for designing lightweight s-boxes with high differential and linear branch numbers, and its application. IEEE Access **9**, 150592–150607 (2021), https://doi.org/10.1109/ACCESS.2021.3126008
13. Knudsen, L.R.: Cryptanalysis of LOKI91. In: AUSCRYPT '92. Lecture Notes in Computer Science, vol. 718, pp. 196–208. Springer (1992), https://doi.org/10.1007/3-540-57220-1_62
14. Sun, L., Wang, M.: Sok: Modeling for large s-boxes oriented to differential probabilities and linear correlations. IACR Transactions on Symmetric Cryptology **2023**(1), 111–151 (Mar 2023), https://tosc.iacr.org/index.php/ToSC/article/view/10310
15. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: ASIACRYPT 2014. Lecture Notes in Computer Science, vol. 8873, pp. 158–178. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_9
16. TTAK.KO-12.0382: 64-bit Block Cipher PIPO. Telecommunications Technology Association of Korea (2022), https://www.tta.or.kr/tta/ttaSearchView.do?key=77&rep=1&searchStandardNo=TTAK.KO-12.0382&searchCate=TTAS
17. Yadav, T., Kumar, M.: Modeling large s-box in MILP and a (related-key) differential attack on full round PIPO-64/128. In: SPACE 2022. Lecture Notes in Computer Science, vol. 13783, pp. 3–27. Springer (2022), https://doi.org/10.1007/978-3-031-22829-2_1
18. ZDNet: New xbox security cracked by linux fans (Visited on October 4, 2023), https://www.zdnet.com/article/new-xbox-security-cracked-by-linux-fans/
19. 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), https://doi.org/10.1007/s11432-015-5459-7