Masked Implementation of PIPO Block Cipher on 8-bit AVR Microcontrollers*

Hyunjun Kim[1], Minjoo Sim[1], Siwoo Eum[1], Kyungbae Jang[1], Gyeongju Song[1], Hyunji Kim[1], Hyeokdong Kwon[1], Wai-Kong Lee[2], and Hwajeong Seo[1][0000-0003-0069-9061]

[1]IT Department, Hansung University, Seoul (02876), South Korea,
{khj930704, minjoos9797, shuraatum, starj1023, thdrudwn98, khj1594012, korlethean, hwajeong84}@gmail.com
[2]Department of Computer Engineering,
Gachon University, Seongnam, Incheon (13120), Korea,
waikonglee@gachon.ac.kr

Abstract. PIPO is a lightweight block cipher and shows better performance than other block cipher algorithms on low-end microcontrollers (e.g. 8-bit AVR). In addition, PIPO block cipher can utilize the efficient masking method by minimizing the number of non-linear operations. Therefore, PIPO block cipher can prevent side-channel attacks, efficiently. In this paper, we propose an efficient first-order masking technique using a 2-byte random mask by taking an advantage of PIPO block cipher. We present a new OR operation masking technique. Among functions of PIPO, the masked S-layer with 23 AND operations, 5 OR operations, and 46 XOR operations is used. Operations of PIPO block cipher are implemented in AVR assembly languages. The proposed implementation showed 1.5× faster performance enhancements compared to the unprotected C implementation in the encryption process and 2.2× faster performance enhancements compared to the unprotected optimized assembly implementation.

Keywords: PIPO Block Cipher · Masked Implementation · 8-bit AVR Microcontrollers · Side Channel Attack.

* This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 25%) and this work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1F1A1048478, 25%) and this work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-00540, Development of Fast Design and Implementation of Cryptographic Algorithms based on GPU/ASIC, 25%) and this work was partly supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (<Q|Crypton>, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity, 25%).

1 Introduction

Side-channel analysis is an attack technique that obtains a secret key through the unintended information, such as power consumption and electromagnetic waves, while cryptographic modules are working. For this reason, the cryptographic engineer should pay attention on countermeasures against side channel attack for secure implementations. A masking technique, known as an effective countermeasure against side-channel attacks, computes a random value on the sensitive information to hide the information. Secure and scalable masking techniques have been proven through several studies. However, there is room to improve the performance in the implementation.

PIPO block cipher is a lightweight cipher developed for efficient masking implementation and it is designed for limited resources in the IoT environment by minimizing the number of non-linear operations [1]. With this nice feature, the PIPO block cipher can efficiently utilize the masking method, which is a side-channel analysis response technique. The PIPO block cipher outperforms other lightweight 64-bit block ciphers using 128-bit keys in terms of 8-bit AVR implementations. Since the cost of masking is lower than that of other 64-bit lightweight ciphers, the increase in execution time is small when the masking technique is utilized. Therefore, PIPO block cipher can apply side-channel response techniques efficiently to the IoT environment with limited resources. It is very costly to generate a random value of high entropy used for masking. Low non-linearity of PIPO block cipher helps to minimize the masking cost. The masking application should optimize the randomness for a specific masking device, such as an AND gate, or make it work efficiently while reducing the amount of new randomness.

1.1 Our Contributions

We aimed to maximize the advantage of PIPO block cipher to utilize the masking. As a first step, we performed a power analysis attack on the PIPO block cipher [1]. The original paper discusses masking techniques, but it does not attack the PIPO before the masking protection. Through this process, the necessity of countermeasures against PIPO side channels is confirmed. To reduce the cost of generating random values, we propose an efficient first-order masking technique using two 8-bit masks. The mask value is generated in the first time. PIPO uses non-linear operations (i.e. AND and OR operations). For the safe and efficient masking of AND and OR operations, ISW masking technique is used for AND operation, and a new technique for the masking OR operation is proposed. The proposed new OR operation masking is based on the technique of Gross et al. and uses fewer instructions than the existing ones. The masked S-layer with 23 AND operations, 5 OR operations, and 46 XOR operations is designed for the PIPO's masking. There is no additional operations in the rest of the functions except for the initial mask value generation, the process of masking the input plaintext and round key, and the S-layer. When the proposed method is implemented in an assembly code, compared to the existing assembly code implementation reference code, the overhead of proposed method is 120%.

2 PIPO Algorithm

PIPO is a lightweight block cipher algorithm announced at ICISC'20 [1]. It consists of S-Layer using a new S-box and bit permutation using bit rotation in units of bytes. PIPO's new S-Box is cryptographically superior and is designed to be efficient. The differential and linear branching numbers of nonlinear functions is designed to be high, allowing safe use of efficient and low-spreading bit permutations in lightweight environments. A bit slicing implementation using only 11 non-linear bitwise operations and 23 linear bitwise operations is also presented. It outperforms existing lightweight block ciphers in side-channel protection and unprotected environments in 8-bit AVR microcontrollers due to byte-unit bit slicing operations and a small number of non-linear bit operations. In the side-channel analysis, the non-linearity of cryptographic algorithms and their actual implementation affect the attack success [2]. In this paper, we utilized masking considering PIPO's new S-box implementation of bit slicing.

3 Masked PIPO

We applied Boolean masking with the goal of defending against the primary side-channel attack. In Boolean masking, non-linear functions require additional techniques to manage mask values. In PIPO, the AND operation and the OR operation correspond to this. We considered safe and efficient AND and OR masking without using a new random mask. $\oplus$ is an XOR operation, $\wedge$ is an AND operation, $\vee$ is an OR operation, $m0$ and $m1$ are random mask values, and $a_1 = m_0$, $b_1 = m_1$, $q = q_0 \oplus q_1$, $m_2 = m_0 \oplus m_1$.

3.1 Masked XOR operation

In Boolean masking, the XOR operation can maintain the masking state without performing additional operations. In the case of XORing two masked values, it can be protected with a mask value $m2$ as shown in the following equation.

$$q_0 = a \oplus m_0,\ q_1 = b \oplus m_1,\ q = q_0 \oplus q_1 = (a \oplus m_0) \oplus (b \oplus m_1) = (a \oplus b) \oplus m_2$$

When two mask values are the same, the state of the mask is removed. And different mask values must be used. In this paper, the same mask state is implemented by using masks of mask states m_0, m_1, and m_2. The mask value is not removed when the same two values are calculated. Since the remaining mask values are obtained by XORing two mask values, the mask can be easily managed even with three mask values.

3.2 Masking of AND operation

The masking of AND operation uses the same technique as the existing PIPO. The formula below is the AND operation masking of the existing PIPO using two masks [3].

$$\begin{aligned} q &= a \wedge b = (a_0 \oplus a1)(b_0 \oplus b_1) \quad = a_0 \wedge b_0 \oplus a_0 \wedge b_1 \oplus a_1 \wedge b_0 \oplus a_1 \wedge b_1 \\ &= q_0 \oplus q_1 = (a \oplus m_0) \oplus (b \oplus m_1) = (a \oplus b) \oplus m_2 \\ q_0 &= a_0 \wedge b_0 \oplus m_2 \oplus a_0 \wedge b_1 \oplus a_1 \wedge b_0 \oplus a_1 \wedge b_1,\ q1 = m2 \end{aligned}$$

The mask value after the operation is changed to a new mask value. Considering the case where mask values are the same, two input values have different masking states, and the mask value after the operation is an XOR value of the input mask values. There are no additional mask values.

Biryukov et al. realized a masked AND gate that does not require new randomness. Based on this, Gross et al. created a new AND mask structure as follows [4].

$$q_0 = (a_0 \wedge b_0 \oplus (a_0 \wedge b_1 \oplus b_1)) \oplus (a_1 \wedge b_0 \oplus [m_0 \vee m_1]),\ q_1 = a_1$$

[m0 ∨ m1] is a fixed value and can be used equally in one operation. After the operation, it is masked with a single mask ($m0$). It is easy to manage the mask state. The operation process is also more efficient than [4]. However, we applied the technique to PIPO, but our experimental CPA attack revealed the real key. Therefore, in the proposed technique, the existing PIPO AND masking technique was used instead of the corresponding masking technique.

3.3 Masking of OR operation

The OR operation masking is based on the AND masking technique of Gross et al. [4], and a new OR masking technique is used as shown in the following equation.

$$\begin{aligned} &q_0 = (a_0 \wedge b_0) \oplus (a_0 \vee b_1) \oplus ((a_1 \vee b_0) \oplus ((a_1 \wedge b_1) \oplus a_1 \oplus b_1)) = (a_0 \wedge b_0) \oplus (a_0 \vee b_1) \oplus ((a_1 \vee b_0) \oplus ([m_0 \vee m_1] \oplus b_1)) \\ &q_1 = a_1 = m_1 \end{aligned}$$

The AND masking technique of Gross et al. does not use a new random mask and operates using 1 AND, 2 ORs, and 4 XORs. It uses fewer operations than PIPO's existing technique using 4 ANDs and 6 XORs. $[m0 \vee m1]$ is a fixed value and can be used equally in one operation. For the masking technique to be safe, values of the intermediate operations must be statistically identical

Table 1. Security of the proposed masked OR; $t1 = (a_0 \wedge b_0), t2 = (a_0 \vee b_1), t3 = (a_1 \vee b_0, t4 = [m_0 \vee m_1] \oplus b_1, t5 = t3 \oplus t4, t6 = t1 \oplus t2$.

				Secrets			TT(Truth table)											Secrets			TT(Truth table)						
a_0	b_0	a_1	b_1	a	b	ab	$t1$	$t2$	$t3$	$t4$	$t5$	$t6$	$q0$	a_0	b_0	a_1	b_1	a	b	ab	$t1$	$t2$	$t3$	$t4$	$t5$	$t6$	$q0$
0	0	0	0				0	0	0	0	0	0	0	0	1	0	0				0	1	1	0	1	1	0
0	0	1	1	0	0	0	0	1	1	0	1	1	0	0	1	1	1	1	0	1	0	0	1	1	0	0	0
1	1	0	0				0	1	1	1	0	1	1	1	0	0	0				0	1	0	0	0	1	1
1	1	1	1				1	1	1	0	1	0	1	1	0	1	1				1	1	1	0	1	0	1
TT Hamming Weight							1	3	3	1	2	2	2	TT Hamming Weight							1	3	3	1	2	2	2
0	0	0	1				0	1	0	0	0	1	1	0	1	0	1				0	1	1	0	1	1	0
0	0	1	0	0	1	1	0	0	1	0	1	0	1	0	1	1	0	1	1	1	0	0	1	1	0	0	0
1	1	0	1				0	1	1	0	1	1	0	1	0	0	1				0	1	0	0	0	1	1
1	1	1	0				1	1	1	1	0	0	0	1	0	1	0				1	1	1	0	1	0	1
TT Hamming Weight							1	3	3	1	2	2	2	TT Hamming Weight							1	3	3	1	2	2	2

regardless of the values of the intermediate operations. As shown in Table 1, in the proposed method, the Hamming weight of the intermediate operation is the same regardless of the secret value to be hidden. Therefore, the proposed OR operation masking is statistically identical, and actual operation values are not revealed.

3.4 Masking Scheme

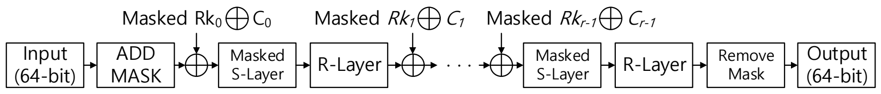

Fig. 1. Proposed masked encryption process.

In order to minimize the cost for generating a random mask value used in the masking technique, the proposed technique proposes PIPO masking using only a 2-byte random mask. For the efficient operation, the masking state of the input and output of the function operating in the round is kept constant. The state of the mask used in the implementation of this paper is shown in Table 2.

The proposed method can be divided into a mask initialization step and an encryption step. In the initialization step, as in Algorithm 1, 8-bit random mask values, $m0$ and $m1$ are generated, and 8 mask values are generated by combining them. The generated mask $m2 = m0 \oplus m1$, and the remaining mask values are generated as in line 4-10 of Algorithm 1. This mask is a pre-work to keep the mask state of the value input to the round function the same as the previous round and the next round, and these values are masked by the round key. The first 64 bits of the round key are masked with $m2, m2, m2, m1, m2, m2, m2$, and $m2$, respectively. For remaining round keys, $m1, m3, m4, m5, m6, m7, m8$, and $m9$ are masked in order.

Table 2. Masking state proposed PIPO masking.

	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]
AddMask	m_1	m_0	m_0	m_0	m_1	m_0	m_0	m_1
AddRK out	m_0	m_1	m_1	m_2	m_0	m_1	m_1	m_0
S-Layer out	m_2	m_1	m_2	m_0	m_1	m_1	m_2	m_1
R-Layer out	m_2	$m_1 \ggg 7$	$m_2 \ggg 4$	$m_0 \ggg 3$	$m_1 \ggg 6$	$m_1 \ggg 5$	$m_2 \ggg 1$	$m_1 \ggg 2$
Remove Mask	m_0	m_1	m_1	m_2	m_0	m_1	m_1	m_0

Table 3. Performance comparison.

Method	C(cycles)	ASM(cycles)
[3]	2,321	1,576
Proposed Method	6,169	3,460
Overhead factor	2.7	2.2

In the encryption process, the input plaintext is first masked with the value of the Masking row of the table. Next, the same process as in the existing PIPO is performed except that a masked S-Layer is used. In the final step, the encryption is completed by removing the masking state.

4 Implementation

In this Section, we describes the implementation of the proposed technique. The application of the masking technique causes a lot of overhead in the speed of the existing algorithm. We implemented the proposed technique in C language and optimized it to minimize overhead and implemented it in AVR assembly. In Atmel Studio 7.0 version, the clock cycle of the encryption process was calculated by operating it in the simulation environment of the ATmega128 board. In the C language implementation 6,169 cycles and 3,460 cycles were shown in the assembly implementation. At this time, the random value generation used in the mask value was not considered because there are various methods.

4.1 8-bit AVR Microcontrollers

With the growth of the Internet of Things (IoT), many low-cost platforms are being used for sensors, actuators and smart devices. In the IoT environment, it is necessary to have a certain level of security for a safe communication environment. A commonly used 8-bit microcontroller AVR has a RISC architecture with 32 general-purpose registers. Typically, one arithmetic instruction requires one clock cycle, while memory access and 8-bit multiply instructions require two clock cycles. Instructions used to implement the optimized PIPO block cipher are summarized in Table 4.

4.2 Mask initialization

As seen in Algorithm 1, 8 new mask values are generated with 2 random values. It is added to the rounded key to make the rotated mask state to the same state as

Table 4. Summarized instruction set of AVR microcontrollers for optimized PIPO block cipher; `Rd`: Destination register, `Rr`: Source register.

Instruction	Operands	Description	Operation	#Clock
AND	Rd, Rr	Logical AND	Rd ← Rd & Rr	1
OR	Rd, Rr	Logical OR	Rd ← Rd | Rr	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	1
MOV	Rd, Rr	Copy Register	Rd ← Rr	1

Algorithm 1 Initmask.

```
Input: 8-bit random values (r1, r2)
Output: m1, m2, m3, m4, m5, m6, m7, m8, m9
1: m0 = r1
2: m1 = r2
3: m2 = m0 ⊕ m1
4: m3 = (m1 >>> 1) ⊕ m1
5: m4 = (m2 >>> 4) in ⊕ m1
6: m5 = (m0 >>> 5) ⊕ m2
7: m6 = (m1 >>> 2) ⊕ m0
8: m7 = (m1 >>> 3) ⊕ m1
9: m8 = (m2 >>> 7) ⊕ m1
10: m9 = (m1 >>> 6) ⊕ m0
```

the AddRK out row in Table 2. With one operation of this pre-step, add-key and L-layer in the proposed method work the same as the existing operation without overhead. The clock cycles generated here are 1,074 cycles in the C language implementation and 604 cycles in the assembly implementation, accounting for 17% of the total.

4.3 Encryption Round

In the encryption process of the proposed method, Add key and L-layer operate without additional operation as a pre-operation of Mask initialization described above. The Add Mask step to mask the plaintext input to the existing algorithm and Remove Mask to remove the mask value after encryption operation are added. And Masked S-Layer with a masking technique applied to the S-layer is used. In the Add Mask step, the received plaintext is masked by XORing the values. In the Remove Mask step, the value is masked by XORing it. 8 XOR operations are added in the Add Mask stage and 8 operations are added in the Remove Mask stage. C implementation of S-layer works like Algorithm 4. S-layer's nonlinear operation and operation and or operation are changed to secand and secor, and additional xor operation is performed. The mask state is not removed in line 33 and line 34. As shown in Algorithm 2, SecAND operates in 12 clock cycles using 12 instructions. As shown in Algorithm 3, SecOR operates in 11 clock cycles using 11 instructions. As a result, the Masked S-layer assembly takes 208 cycles.

5 Experiment

5.1 Experiment Environment

10,000 waveforms from round 1 to round 3 were collected using ChipWhisperer-Lite in ATxmega128 MCU, which is an 8-bit processor, for PIPO-64/128 implemented by bit slicing.

Algorithm 2 SecAND in AVR assembly.

```
Input: temp registers: r4, r5, data registers: $a_0$, $b_0$, mask registers: $a_1$, $b_1$, m($a_1 \oplus b_1$)
Output: r4
1: MOV r4, $b_0$
2: AND r4, $a_0$
3: EOR r4, m
4: MOV r5, $a_0$
5: AND r5, $b_1$
6: EOR r4, r5
7: MOV r5, $a_1$
8: AND r5, $b_0$
9: EOR r4, r5
10: MOV r5, $a_1$
11: AND r5, $b_1$
12: EOR r4, r5
```

Algorithm 3 SecOR in AVR assembly.

```
Input: temp registers: r4, r5, r6, data registers: $a_0$, $b_0$, mask registers: $a_1$, $b_1$, m($a_1 \vee b_1$)
Output: r4
1: MOV r4, m
2: EOR r4, $b_1$
3: MOV r5, $b_0$
4: OR r5, $a_1$
5: EOR r5, r4
6: MOV r4, $b_0$
7: AND r4, $a_0$
8: MOV r6, $a_0$
9: OR r6, $b_1$
10: EOR r4, r6
11: EOR r4, r5
```

5.2 Power analysis of unprotected PIPO

In PIPO, the S-box is implemented and operated by the block-unit bit slicing technique. Considering this, after the S-Box operation, the CPA attack was performed by targeting the upper first bit of the median value [5]. Figure 1 shows the result of acquiring the first 8 bits of the key of the first 64-bit transposed round. It can be seen that the attack was successful by showing the highest correlation coefficient at `0x05`, which is the actual key value among the guessed key values. By repeating the same process eight times, the key of the first 64-bit transposed round can be found. If you attack the middle value of the S-Box operation of the second round based on the information found, you can find out the key value of the second transposed round. The original key value can be found by transposing the obtained two round kit values to their original state and performing XOR with the round constant.

8-bit intermediate value attack after S-box operation We first performed a CPA attack using the first 8-bit value after the S-box operation as an intermediate value. Figure 2 is the result of attacking the 1-byte partial key. If the attack is performed properly, the correct value of the guess key (`85`), should come out as the highest correlation coefficient value. However, as a result of the attack, the guess key with the highest correlation coefficient was `77`. In addition, high correlation coefficients were shown at `72`, `80`, and `85`. Unlike the original prediction, `85` did not have the highest correlation coefficient, so an accurate attack could not be performed with the attack method using the 8-bit value after the S-Layer operation as the median value.

1-bit attack of 8-bit median value after S-box operation We judged that the cause of the inaccurate attack result in 8-bit S-box attack is that the S-box of PIPO is implemented by bit slicing in block units, and taking this into account, we performed the attack on each bit of the value after the S-box

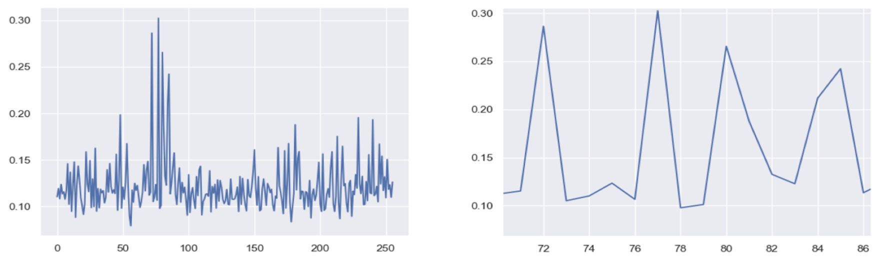

Fig. 2. Attack result (left) and attack analysis result (right) using 8-bit value after S-Box operation as an intermediate value.

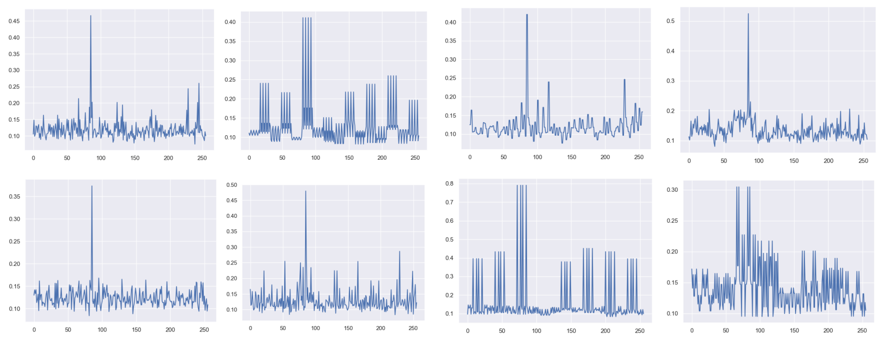

Fig. 3. Result of performing 1 bit CPA attack after S-Layer operation.

operation. As a result, it was confirmed as shown in Figure 3 that characteristic attack results were obtained for each bit. As can be seen in Figure 3, in the most significant bit, the fourth, fifth, and sixth bits, the highest correlation coefficient value is one, and the round key value to be obtained can be obtained at the point where the high correlation coefficient value is obtained. On the other hand, it was confirmed that the second upper bit, the third, the seventh, and the least significant bit had two or more identical correlation coefficient values.The attack on the 4-th bit showing the highest correlation coefficient was possible with at least 70 waveforms.

Leak Point Analysis A CPA attack was performed to determine where the power waveform leak occurred. Figure 4 is the result of performing CPA for each point using an 8-bit correct guess key and outputting it. Blue is the power consumption waveform of the 1st and second rounds of PIPO, and orange is the result of displaying the high correlation coefficient by performing CPA for each point. A leak occurred in S-box operation and P-box operation in round 1, keyAdd operation in round 2, and S-box operation.

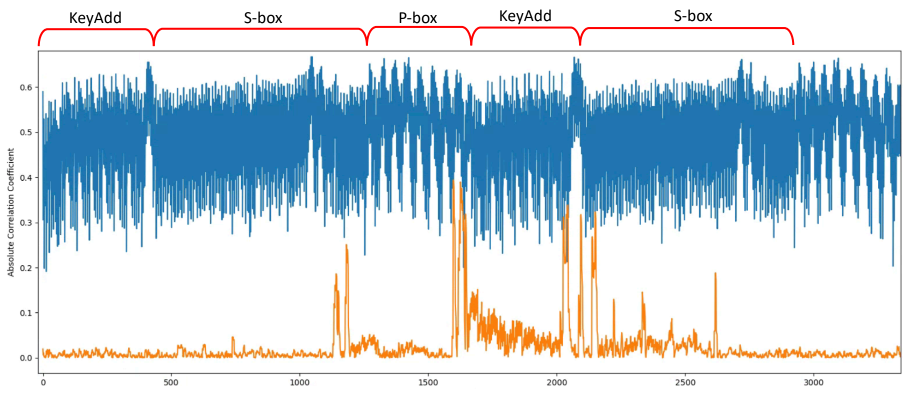

Fig. 4. The result of performing CPA for each point using an 8-bit correct guess key.

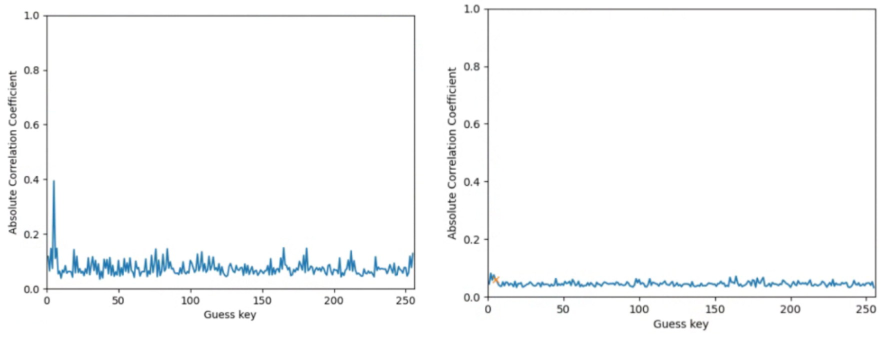

Fig. 5. CPA attack results on PIPO before protection (left) and CPA attack results on PIPO with masking (right).

5.3 Power analysis of the proposed masked PIPO

In order to check whether the proposed masking technique is safe against the primary side-channel analysis attack, the attack was performed in the same way as the power analysis for the existing PIPO. Figure 2 shows the results of the CPA attack against the masked PIPO. The correlation coefficients of all guess keys were low, and the value showing the highest correlation coefficient is irrelevant to the actual value. Therefore, the proposed method is safe for primary side-channel analysis.

Algorithm 4 S-layer in C code.

```
Input: $u8\ X[8],\ m0,\ m1,\ m2.$
Output: $u8\ X[8].$
1: $u8\ T[3] = \{0,\};$
2: $u8\ temp = 0;$
3: $u8\ c = m0|m1;$

4: $secand(temp, X[7], X[6], m0, m1, m2);$
5: $X[5] = X[5] \oplus temp;\ //(m1 \oplus m2) = m0$

6: $secand(temp, X[3], X[5], m2, m0, m1);$
7: $X[4] = X[4] \oplus temp;\ //m0 \oplus m1 = m2$
8: $X[7] = X[7] \oplus X[4];\ //m0 \oplus m2 = m1$
9: $X[6] = X[6] \oplus X[3];\ //m1 \oplus m2 = m0$

10: $secor(temp, X[5], X[4], m0, m2, c);$
11: $X[3] = X[3] \oplus temp;\ //m2 \oplus m0 = m1$
12: $X[5] = X[5] \oplus X[7];\ //m0 \oplus m1 = m2$

13: $secand(temp, X[5], X[6], m2, m0, m1);$
14: $X[4] = X[4] \oplus temp;\ //m2 \oplus m1 = m0$

15: $secand(temp, X[1], X[0], m1, m0, m2);$
16: $X[2] = X[2] \oplus temp;\ //m1 \oplus m2 = m0$

17: $secor(temp, X[1], X[2], m1, m0, c);$
18: $X[0] = X[0] \oplus temp;\ //m0 \oplus m1 = m2$

19: $secor(temp, X[2], X[0], m0, m2, c);$
20: $X[1] = X[1] \oplus temp;\ //m1 \oplus m0 = m2$
21: $X[2] =\ \ X[2];\ //m0$

22: $X[7] = X[7] \oplus X[1];\ //(m1 \oplus m2) = m0$
23: $X[3] = X[3] \oplus X[2];\ //m1 \oplus m0 = m2$
24: $X[4] = X[4] \oplus X[0];\ //m0 \oplus m2 = m1$
25: $T[0] = X[7];\ //m0$
26: $T[1] = X[3];\ //m2$
27: $T[2] = X[4];\ //m1$

28: $secand(temp, X[5], T[0], m2, m0, m1);$
29: $X[6] = X[6] \oplus temp;\ //m0 \oplus m1 = m2$

30: $T[0] = T[0] \oplus X[6];\ //m0 \oplus m2 = m1$

31: $secor(temp, T[2], T[1], m1, m2, c);$
32: $X[6] = X[6] \oplus temp;\ //m2 \oplus m1 = m0$

33: $T[1] = X[5] \oplus m0;\ //m2 \oplus m1 = m0$

34: $secor(temp, X[6], T[2], m0, m1, c);$
35: $X[5] = X[5] \oplus temp;\ //m2^{m}0 = m1$

36: $secand(temp, T[1], T[0], m0, m1, m2);$
37: $T[2] = T[2] \oplus temp;\ //m1 \oplus m2 = m0$

38: $X[2] = X[2] \oplus T[0];\ //m0 \oplus m1 = m2$
39: $T[0] = X[1] \oplus T[2];\ //m2 \oplus m0 = m1$
40: $X[1] = X[0] \oplus T[1];\ //m2 \oplus m0 = m1$
41: $X[0] = X[7] \oplus m1;\ //m2$
42: $X[7] = T[0];\ //m1$
43: $T[1] = X[3];\ //m2$
44: $X[3] = X[6];\ //m0$
45: $X[6] = T[1];\ //m2$
46: $T[2] = X[4];\ //m1$
47: $X[4] = X[5];\ //m1$
48: $X[5] = T[2];\ //m1$
```

6 Conclusions and Future Work

We proposed a side-channel analysis of the lightweight block cipher algorithm PIPO and an efficient first-order masking technique. As a result of the CPA attack targeting the provided and collected waveforms, a successful CPA attack was performed with a bit-by-bit attack on the 8-bit median value of the S-box operation. And we propose an efficient first-order masking technique for PIPO. For masking, a 2-byte random mask is used to minimize the cost of generating a mask, and for this, a new OR operation masking is proposed. Among the functions of PIPO, Masked S-layer with 23 AND operations, 5 OR operations, and 46 XOR operations added is used, and the rest performs the same operation and operates efficiently. The safety of the proposed side-channel countermeasures against CPA attacks was confirmed through experiments. And as a result of implementing the assembly code in the AVR environment, the new masking technique showed 50% and 120% overhead respectively compared to the reference c code and the optimized assembly code provided by [1]. The proposed method aimed at efficient primary masking for PIPO for IoT environment with few resources. PIPO is also efficient for high-order masking with few non-linear operations. In the future, we plan to apply efficient high-order masking for PIPO to protect side-channel attacks in various environments.

References

1. H. Kim, Y. Jeon, G. Kim, J. Kim, B.-Y. Sim, D.-G. Han, H. Seo, S. Kim, S. Hong, J. Sung, *et al.*, “PIPO: A lightweight block cipher with efficient higher-order masking software implementations,” in *International Conference on Information Security and Cryptology*, pp. 99–122, Springer, 2020.
2. Q. Tian, M. O’neill, and N. Hanley, “Can leakage models be more efficient? non-linear models in side channel attacks,” *2014 IEEE International Workshop on Information Forensics and Security, WIFS 2014*, pp. 215–220, 04 2015.
3. Y. Ishai, A. Sahai, and D. Wagner, “Private circuits: Securing hardware against probing attacks,” 07 2003.
4. H. Gross, K. Stoffelen, L. Meyer, M. Krenn, and S. Mangard, “First-order masking with only two random bits,” pp. 10–23, 11 2019.
5. E. Prouff, “DPA attacks and S-boxes,” vol. 3557, pp. 1–8, 07 2005.