electronics

MDPI

Article

Grover on PIPO

Kyungbae Jang [1], Gyeongju Song [1], Hyeokdong Kwon [1], Siwoo Uhm [1], Hyunji Kim [1] and Wai-Kong Lee [2] and Hwajeong Seo [1,*]

[1] Division of IT Convergence Engineering, Hansung University, Seoul 02876, Korea; starj1234@hansung.ac.kr (K.J.); thdrudwn98@hansung.ac.kr (G.S.); hyeok@hansung.ac.kr (H.K.); smile267@hansung.ac.kr (S.U.); 1594012@hansung.ac.kr (H.K.)

[2] Department of Computer Engineering, Gachon University, Incheon 13120, Korea; waikonglee@gachon.ac.kr

* Correspondence: hwajeong@hansung.ac.kr; Tel.: +82-760-8033

Abstract: The emergence of quantum computers is threatening the security of cryptography through various quantum algorithms. Among them, the Grover search algorithm is known to be efficient in accelerating brute force attacks on block cipher algorithms. To utilize the Grover's algorithm for brute force attacks, block ciphers must be implemented in quantum circuits. In this paper, we present optimized quantum circuits of the SPN (Substitution Permutation Network) structured lightweight block cipher, namely the PIPO block cipher. In particular, the compact design of quantum circuits for the 8-bit Sbox is investigated. These optimization techniques are used to implement other cryptographic operations as quantum circuits. Finally, we evaluate quantum resources of Grover search algorithm for the PIPO block cipher in ProejctQ, a quantum simulator provided by IBM.

Keywords: quantum computers; cryptography; grover search algorithm; PIPO; quantum resources

Citation: Jang, K.; Song, G.; Kwon, H.; Uhm, S.; Kim, H.; Lee, W.-K.; Seo, H. Grover on PIPO. *Electronics* **2021**, *10*, 1194. https://doi.org/10.3390/electronics10101194

Academic Editors: Hussein Abulkasim and Bingxian Mu

Received: 31 March 2021
Accepted: 13 May 2021
Published: 17 May 2021

1. Introduction

International IT (Information Technology) companies, such as Google and IBM, are investing in the development of quantum computers. Quantum computers use qubits with a superposition property that allows states of 0 and 1 at the same time with the certain probability. Through this property, n-qubit in superposition state can express 2^n cases and has the advantage of parallel processing. Quantum computers are expected to have strengths in fields of deep learning, chemistry, and simulation due to their parallel computing abilities. However, the computational power of quantum computers poses a huge threat to cryptography. Representative quantum algorithms threatening block cipher and public key cryptography are Grover's algorithm [1] and Shor's algorithm [2], respectively. Large-scale quantum computers using Shor's algorithm can solve factorization and discrete algebra problems within a polynomial time. If quantum computers with large-scale qubits arrive, ECC (Elliptic Curve Cryptography) and RSA (Rivest–Shamir–Adleman), which are widely used in asymmetric key cryptography based on these mathematical problems, are no longer available. Currently, there are no large quantum computers that can break RSA and ECC, but companies are launching quantum computers that use more qubits than ever. NIST (National Institute of Standards and Technology) is working to standardize asymmetric key cryptography https://csrc.nist.gov/projects/post-quantum-cryptography (accessed on 13 May 2021) that can replace RSA and ECC in the preparation for attacks by large quantum computers.

The Grover search algorithm accelerates brute force attacks against symmetric key cryptography. The n-bit security level of symmetric key cryptography is reduced to the $n/2$-bit security level on quantum computers using the Grover's algorithm. For example, the key-recovery attack on full AES are based on biclique attack [3]. The attack is faster than the brute-force by a factor of about four. To recover an AES-128 key, this attack requires $2^{126.2}$ operations. In Reference [4], the result has been further improved to $2^{126.0}$ for AES-128. For the case of Grover search algorithm, the complexity is only 2^{64} for AES-128. To operate

the Grover's algorithm on quantum computers, target symmetric key cryptography must be implemented in a quantum circuit. For this reason, studies have been conducted to optimize and implement symmetric key ciphers into quantum circuits. Starting with the implementation of the most widely used symmetric key cipher, such as AES [5–7], this research area has recently expanded to lightweight block ciphers [8–12].

In quantum circuit implementations, qubits, quantum gates, and circuit depth are key factors of the optimization. There are trade-offs among three optimization factors, such as reducing the use of quantum gates instead of increasing the use of qubits. If quantum computers that can use a number of qubits, it will be beneficial to increase qubits and reduce the circuit depth or quantum gates. However, using a large number of qubits is currently impossible at the quantum computer development level. The most ideal case is to optimize both qubits and quantum gates.

In this paper, we implement the SPN structured lightweight PIPO block cipher [13] on quantum computers for the application of the Grover search algorithm. When implementing SPN structured block ciphers as quantum circuits, optimizing the substitution layer is the main issue [6,7,14,15]. The PIPO block cipher uses 8-bit Sbox with the unbalanced-bridge structure. If the Sbox operation of PIPO block cipher is implemented in a quantum circuit straightforwardly, many qubits should be needed. However, we achieve the optimal number of qubits by remodeling the Sbox of PIPO block cipher. Furthermore, other PIPO operations are also optimized and optimal PIPO quantum circuits are obtained. Finally, based on the proposed PIPO quantum circuit, the cost of the attack for key search using the Grover's algorithm is estimated through the quantum simulator of the ProjectQ.

Contribution

- The first optimized implementation of the PIPO block cipher in quantum circuits: This paper is the first work that implements and optimizes the PIPO block cipher in quantum gates. We efficiently implement the PIPO block cipher as a quantum circuit, which was optimized in terms of qubits, quantum gates, and circuit depth.
- Compact design of substitution layer for quantum computers: When implementing SPN-structured block ciphers as quantum circuits, it is important to optimize the substitution layer. In this paper, we implemented the 8-bit wise substitution layer of the PIPO block cipher to the optimal number of qubits and quantum gates.
- Quantum resource estimation of Grover search algorithm on PIPO block cipher: The quantum programming tool IBM ProjectQ [16] is utilized to implement the PIPO block cipher in quantum circuits. Based on this, we evaluate quantum resources for using the Grover search algorithm to the PIPO block cipher.

2. Related Work

2.1. PIPO Block Cipher

In ICISC'20, a lightweight block cipher based on the innately bitslicing Sbox was firstly proposed [13]. The new Sbox has an unbalanced-bridge structure, which uses an 8-bit Sbox by combining 3-bit Sbox and 5-bit Sbox. They used these smaller Sboxes to design an efficient and secure Sbox. The approach also supports the bitslicing method in nature. The PIPO block cipher supports 64-bit plaintext and 128-bit (i.e., 13 rounds) or 256-bit keys (i.e., 17 rounds). The notations used in this paper are explained in Table 1.

Table 1. Descriptions of notations in this paper.

Notation	Explanation
$\oplus$	XOR operation
&	AND operation
$\vee$	OR operation
$\neg$	NOT operation
$\lll i$	Rotation left operation by i-bit

2.1.1. Encryption of PIPO Block Cipher

Each round consists of AddRoundkey, Substitution layer, and Permutation layer. Before starting rounds, the first 64-bit of the master key is used as the round key (i.e., whitening key) and are XORed to the plaintext. The encryption structure of the PIPO block cipher is shown in Figure 1.

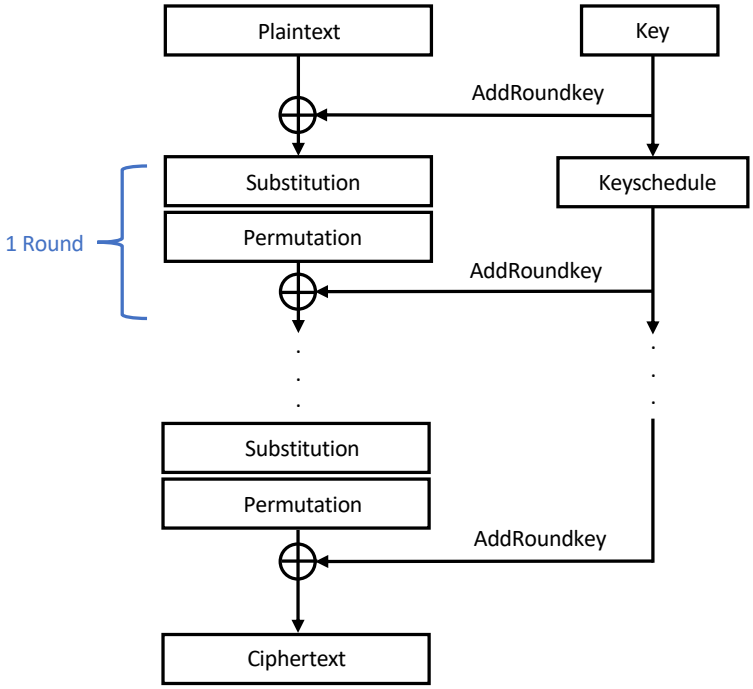

Figure 1. Encryption structure of PIPO.

2.1.2. AddRoundkey and Keyschedule of PIPO Block Cipher

In the AddRoundkey operation, the 64-bit round key (RK) generated from the master key (K) is XORed to the 64-bit block (B). The PIPO block cipher uses simple Keyschedule operation to generate round keys. In the 128-bit key version, round keys are generated as $RK_i = K_{i\,mod\,2} \oplus i (i = 0, 1, 2, ..., 13)$, and RK_0 is the whitening key. In the 256-bit key version, round keys are generated as $RK_i = K_{i\,mod\,4} \oplus i (i = 0, 1, 2, ..., 17)$, and RK_0 is the whitening key.

2.1.3. Substitution Layer of PIPO Block Cipher

The PIPO block cipher is designed to efficiently utilize the bitslicing method. The 64-bit block B is divided into an 8×8 array and the 8-bit Sbox is applied to each column, which is shown in Figure 2. In the bitsliced implementation, the bit of the Sbox (e.g., $b_0, b_8, b_{16}, b_{24}, b_{32}, b_{40}, b_{48}, b_{56}$) can be replaced with bytes (i.e., $B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7$). Therefore, the substitution layer for 64-bit block B can be performed at once.

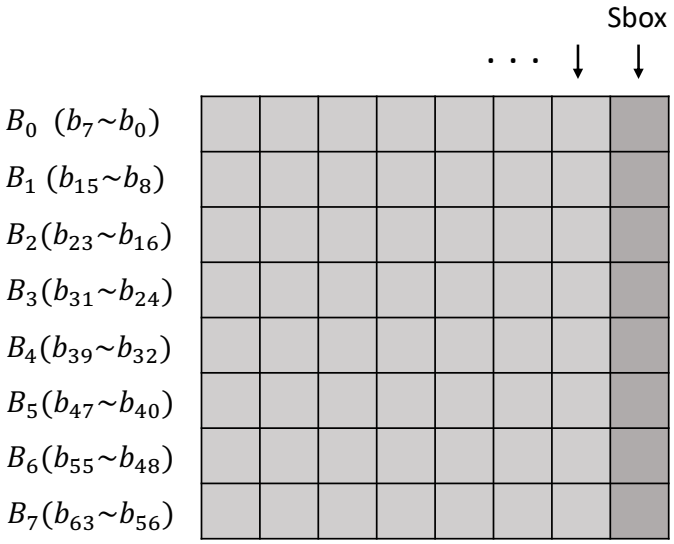

Figure 2. Substitution layer structure of PIPO.

The PIPO block cipher uses the 8-bit Sbox with an unbalanced bridge structure consisting of a combination of 3-bit and 5-bit Sboxes, as shown in Figure 3. It is designed to implement the efficient bitslicing and consists of 11 or fewer linear operations. The differential uniformity of the Sbox is 16 or less and the non-linearity is 96 or more. The bitslicing implementation of the PIPO Sbox is shown in Algorithm 1.

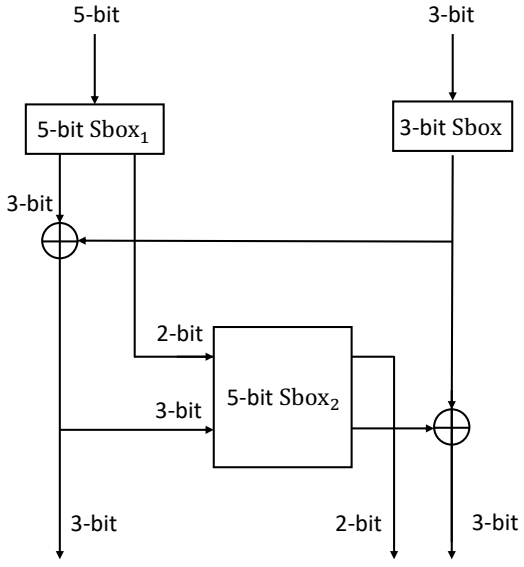

Figure 3. Unbalanced-bridge structure of PIPO Sbox.

Algorithm 1 Bitslicing implementation of PIPO Sbox.

Input: 8-bit $X(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$

Output: 8-bit $X(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$

1: 5-bit Sbox_1:

2: $x_5 \leftarrow x_5 \oplus (x_7 \ \& \ x_6)$

3: $x_4 \leftarrow x_4 \oplus (x_3 \ \& \ x_5)$

4: $x_7 \leftarrow x_7 \oplus x_4$

5: $x_6 \leftarrow x_6 \oplus x_3$

6: $x_3 \leftarrow x_3 \oplus (x_4 \vee x_5)$

7: $x_5 \leftarrow x_5 \oplus x_7$

8: $x_4 \leftarrow x_4 \oplus (x_5 \ \& \ x_6)$

9: 3-bit Sbox :

10: $x_2 \leftarrow x_2 \oplus (x_1 \ \& \ x_0)$

11: $x_0 \leftarrow x_0 \oplus (x_2 \vee x_1)$

12: $x_1 \leftarrow x_1 \oplus (x_2 \vee x_0)$

13: $x_2 \leftarrow \neg\, x_2$

14: Extend XOR :

15: $x_7 \leftarrow x_7 \oplus x_1$

16: $x_3 \leftarrow x_3 \oplus x_2$

17: $x_4 \leftarrow x_4 \oplus x_0$

18: 5-bit Sbox_2:

19: $t_0 \leftarrow x_7, \ \ t_1 \leftarrow x_3, \ \ t_2 \leftarrow x_4$

20: $x_6 \leftarrow x_6 \oplus (t_0 \ \& \ x_5)$

21: $t_0 \leftarrow t_0 \oplus x_6$

22: $x_6 \leftarrow x_6 \oplus (t_2 \vee t_1)$

23: $t_1 \leftarrow t_1 \oplus x_5$

24: $x_5 \leftarrow x_5 \oplus (x_6 \vee t_2)$

25: $t_2 \leftarrow t_2 \oplus (t_1 \ \& \ t_0)$

26: Truncate XOR and Swap :

27: $x_2 \leftarrow x_2 \oplus t_0, \ \ t_0 \leftarrow x_1 \oplus t_2, \ \ x_1 \leftarrow x_0 \oplus t_1, \ \ x_0 \leftarrow x_7, \ \ x_7 \leftarrow t_0$

28: $t_1 \leftarrow x_3, \ \ x_3 \leftarrow x_6, \ \ x_6 \leftarrow t_1$

29: $t_2 \leftarrow x_4, \ \ x_4 \leftarrow x_5, \ \ x_5 \leftarrow t_2$

30: **return** $X(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$

In Algorithm 1, 3 bits that are inputs of Sbox_2 are used as the final result. For this reason, temp bits (t_0, t_1, t_2) are required. This feature is inefficient when it is implemented in quantum circuits. However, we remodeled this Sbox design and optimized it without additional qubits. This is explained in Section 3.

2.1.4. Permutation Layer of PIPO Block Cipher

The PIPO block cipher performs bit rotations within a byte for the implementation efficiency. To achieve the full diffusion within the minimum number of rounds, the bit rotation is performed for seven rows (i.e., B_{1_7}) as shown in Equation (1). Through this, the PIPO block cipher can achieve the full diffusion within 2 rounds.

$$\begin{aligned} &B_1 \leftarrow B_1 \lll 7, \quad B_2 \leftarrow B_2 \lll 4, \quad B_3 \leftarrow B_3 \lll 3, \\ &B_4 \leftarrow B_4 \lll 6, \quad B_5 \leftarrow B_5 \lll 5, \quad B_6 \leftarrow B_6 \lll 1, \\ &B_7 \leftarrow B_7 \lll 2 \end{aligned} \tag{1}$$

2.2. Quantum Computer and Programming

Unlike classical logic gates, quantum gates used in quantum computers using qubits must be reversible. Quantum computers can perform the classical computing using several reversible quantum gates. Some of these examples are X gate, CNOT gate and Toffoli gate shown in Figure 4. The X gate is the same as the NOT operation in classical computers, which outputs the opposite value of the input qubit. The CNOT gate is the same as the XOR operation. In CNOT (X, Y), the input Y qubit becomes the result value $X \oplus Y$, and the input X qubit maintains the X state as it is. The Toffoli gate is the same as the AND operation. In Toffoli (X, Y, Z), the the result value $(X \,\&\, Y)$ is stored in Z, and the input qubits X and Y remain unchanged. Unlike classical computers, quantum gates used in quantum computers are reversible.

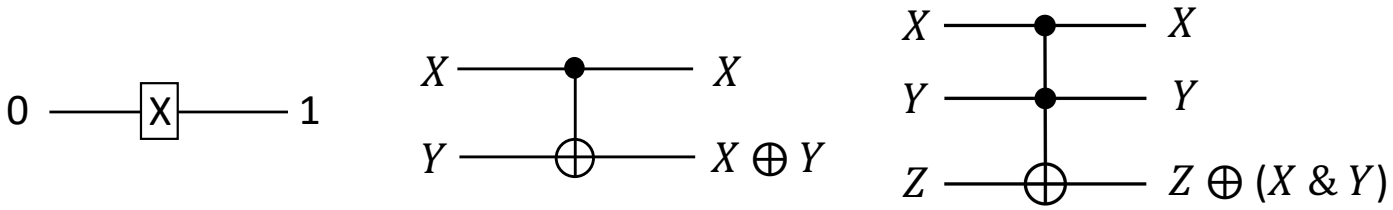

Figure 4. Descriptions: of X gate (**left**); CNOT gate (**middle**);and Toffoli gate (**right**).

In the Sbox of PIPO block cipher, the OR operation is used. The OR operation can be replaced by a combination of X gates and one Toffoli gate. This is shown in Figure 5. To calculate the OR result of X and Y, X gates should be executed before performing the Toffoli gate. Then, the Toffoli gate is performed and the OR result is stored in Z. If input values (X and Y) are needed, the reverse operation should be performed. In quantum circuits, reverse operations are used to perform certain operations again and return them to their original states.

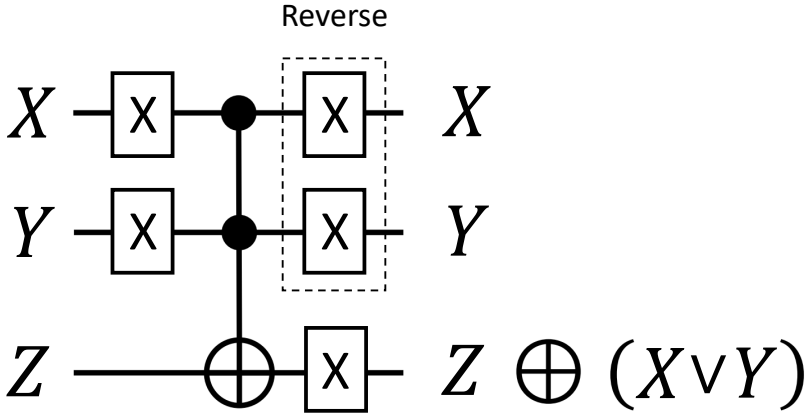

Figure 5. OR operation in quantum computers.

It is easy to control bits in the classical programming, but controlling qubits in the quantum programming is more challenging. When implementing quantum circuits, it is important to minimize the use of qubits. Although quantum computers are in the process of increasing the number of qubits, supported qubits in existing quantum computers are limited. Hence, reducing the use of qubits required for quantum circuits is a key factor in the optimization. If qubits used during the operations are no longer needed, they can be reused. To do this, qubits to be reused should be initialized to zero. Unlike classical computers, it is non-trivial to initialize qubits to zero.

For example, in quantum computers, to reuse the qubit A (i.e., initialize to zero), the qubit B in the same state as A exists somewhere. The A qubit can be reused by XORing the qubit B to A with the CNOT gate (i.e., CNOT $(B, A) \rightarrow A = A \oplus B = 0$). Otherwise, the qubit A will just become the garbage qubit. Similarly, it is hard to change the state of the existing qubit A to the same state as the qubit B when A and B are in different states. For this reason, the qubit A should be initialized to zero first, and then the qubit B should be XORed to the qubit A.

2.3. Grover's Algorithm

The Grover search algorithm accelerates the brute force attack. If n times were required for the brute force attack, it is reduced to $\sqrt{n}$ times by applying the Grover search algorithm. This quantum algorithm consists of an oracle that inverts the sign to return the answer, and a diffusion operator that increases the measurement probability of the returned answer. In the example where the answer of 2-qubit is 10, after the oracle and the diffusion operator are shown in Figure 6. Grover's search increases the probability of measuring the answer by repeating the oracle and diffusion operator. However, in the case of 2-qubit, the answer is found with 100% probability without repetition.

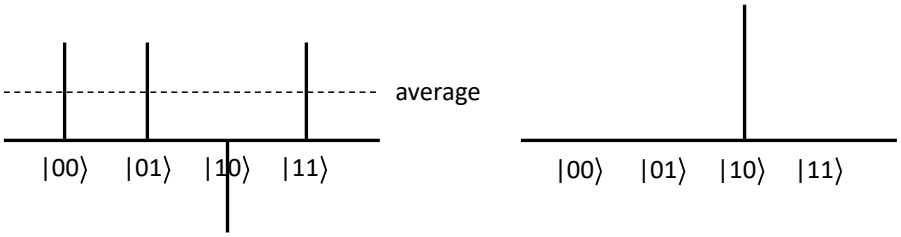

Figure 6. After performing oracle (**left**) and diffusion operator (**right**).

The Grover search algorithm can be applied to key search of block cipher, and the overall structure is shown in Figure 7.

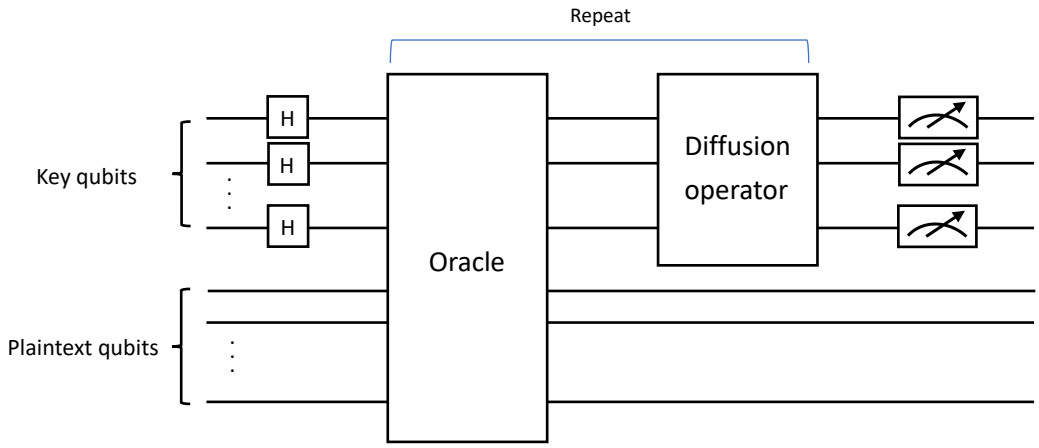

Figure 7. Grover search algorithm for block cipher key search.

First, the n-qubit key to which the Hadamard gate is applied is in a superposition state. For the n-qubit key, all key values (i.e., the number of 2^n cases) exist as a probability at the same time. This is the main advantage of quantum computers. In oracle, an encryption quantum circuit is implemented, and the plaintext qubits are encrypted with key qubits in the superposition state. Therefore, in plaintext qubits, all possible ciphertexts encrypted with all possible keys exist as a probability. We find all the ciphertexts that match the known ciphertext. This can be found with X gates and Controlled-Z gates.

Figure 8 is a simple example to help understand how the oracle finds the key when the 2-qubit plaintext $P(p_1, p_0)$ is encrypted with a 2-qubit key $K(k_1, k_0)$ and the known ciphertext $C(c_1, c_0)$ is 10. Among the generated ciphertexts, when the ciphertext C is 10, it becomes 11 due to the X gate and the Controlled-Z gate is activated. At this time, the key K qubits are entangled with the plaintext P qubits. Therefore, the sign of the key value (e.g., 10) used when generating ciphertext 10 is inverted. Finally, the reverse operation is performed on the previously performed operations. Because the Grover's search has to iterate oracle and diffusion operator. The ciphertext 11 must be returned to the generated ciphertext 10 by the reverse operation of the X gate, and the original plaintext must be returned through the reverse operation of encryption.

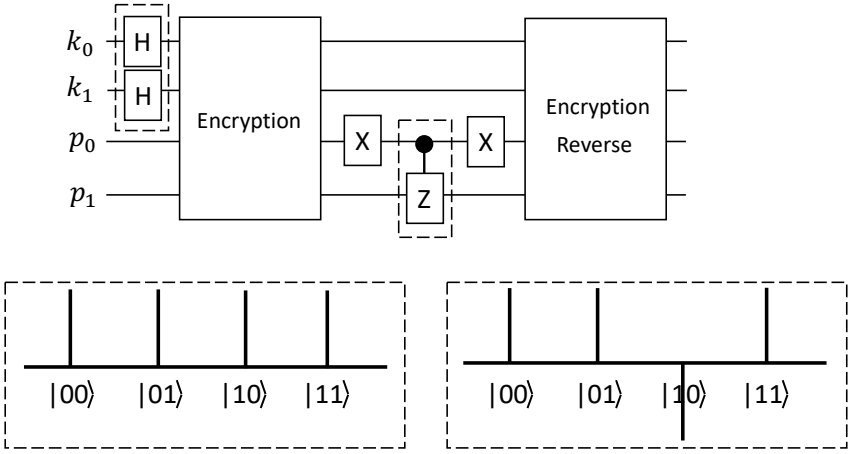

Figure 8. Oracle for Grover key search.

As mentioned above, the diffusion operator amplifies the amplitude of the solution and operates on key qubits. The Grover search algorithm properly iterates the oracle and diffusion operator to increase the probability of the correct key value, and then finally measures it. The diffusion operator does not change much because a formalized method is used. Therefore, in the Grover's search algorithm, how to implement the oracle is important, and, in the case of key search for block cipher, it is most important to optimize the encryption quantum circuit implemented in the oracle.

3. Proposed Method

3.1. Quantum Circuit Design for PIPO Block Cipher

Quantum resources required for key search using Grover's algorithm are determined by how optimized the target block cipher is in oracle. AddRoundkey, Keyschedule, Substitution, and Permutation used for the PIPO encryption are all implemented in quantum circuits. In our proposed PIPO quantum circuits, only qubits for plaintext and master key are allocated. In total, 192 qubits are used for the PIPO-64/128 encryption, and 320 qubits are used for the PIPO-64/256 encryption, respectively.

3.2. AddRoundKey of PIPO Block Cipher

In AddRoundkey, the 64-bit round key RK is XORed on the 64-bit block B. Since it is a simple structure using only XOR operation, AddRoundkey is designed only with

CNOT gates. The quantum circuit for AddRoundkey is explained in Algorithm 2. In the notation CNOT (a, b), the operation target is b. For example, CNOT (a, b) indicates $a = a$ and $b = a \oplus b$. In Algorithm 2, the result of the XOR operation is stored in the qubit of block B. Quantum resources used in the Grover's search algorithm are determined by how oracle is implemented.

Algorithm 2 AddRoundkey in quantum circuits.

Input: 64-qubit block $B(b_{63}, ..., b_0)$, 64-qubit round key $RK(rk_{63}, ..., rk_0)$

Output: 64-qubit block $B(b_{63}, ...b_0)$

1: **for** $i = 0$ to 63 **do**
2: $b_i \leftarrow$ CNOT (rk_i, b_i)
3: **end for**
4: **return** $B(b_{63}, ..., b_0)$

3.3. Keyschedule of PIPO Block Cipher

In Keyschedule, the master key K is divided into 64 bits ($K = K_1||K_0$ or $K = K_3||K_2||K_1||K_0$), and these are selected and XORed according to the round constant i to be used as round keys ($RK_i = K_{i\ mod\ 2} \oplus i$ or $RK_i = K_{i\ mod\ 4} \oplus i$). The operation is XOR-ing the round constant i. Unlike AddRoundkey, we only used X gates, which is simpler than CNOT gates. Since the value of i is known before the operation, qubits of K are flipped by the X gate to positions where the bit of i is 1. In the Round 1 $(i = 1)$, K_1 is used and i is 1. The X gate is performed on the least significant bit of K_1.

We minimized the use of these X gates. In the case of PIPO-64/128, we perform the XOR operation of round constants as shown in Figure 9. K_0 is used for Round 1 and 3, and K_1 is used for Round 2 and 4. Therefore, K_0 is XORed with round constants 1 and 3 in order, and K_1 is XORed with 2 and 4 in order, because K_1 is XORed with 2 first, the bit of $K_1[1]$ is flipped due to the X gate. In the next Round 4, since 4 is XORed, the X gate is performed only in $K_1[2]$. After being used as a key for Round 2, $K_1[1]$ should be returned to its original state by the reverse operation. However, there is a part where this reverse operation can be omitted. In the case of K_0, constant 1 is XORed in Round 1, and an X gate is performed at $K_0[0]$. In Round 3, since X gate is performed on both $K_0[1]$ and $K_0[0]$, the reverse operation is omitted after the Round 1, and X gate is additionally performed only on $K_0[1]$ in the Round 3. In PIPO-64/256, the maximum round constant is 17 and it works for the least significant 5-bit and the method is the same.

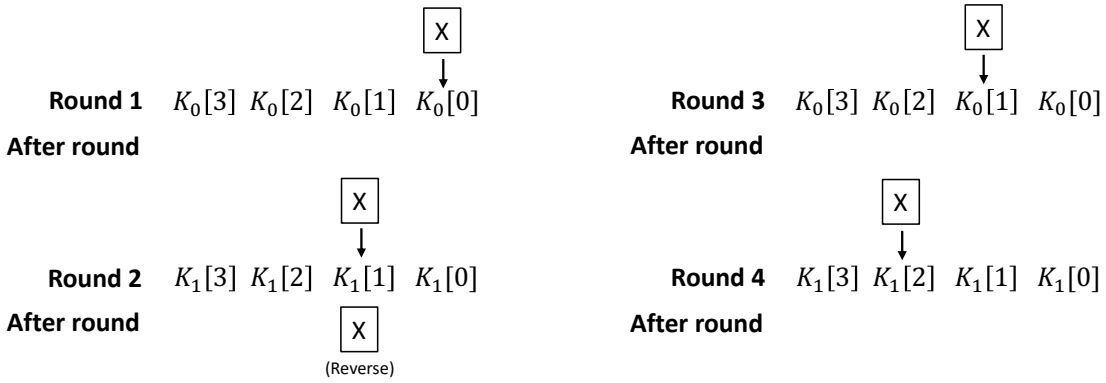

Figure 9. Keyshcedule of PIPO from Round 1 to Round 4.

3.4. Substitution of PIPO Block Cipher

3.4.1. Quantum Circuit Design for 3-Qubit Sbox and 5-Qubit $Sbox_1$

In Algorithm 1, 5-bit $Sbox_1$ and 3-bit Sbox can be implemented using CNOT gates, Toffoli gates, and X gates. It can also be implemented without any additional qubits. These are shown in Algorithms 3 and 4.

Algorithm 3 5-qubit $Sbox_1$ in quantum circuits.

Input: 5-qubit $X(x_7, x_6, x_5, x_4, x_3)$

Output: 5-qubit $X(x_7, x_6, x_5, x_4, x_3)$

1: $x_5 \leftarrow$ Toffoli (x_7, x_6, x_5)

2: $x_4 \leftarrow$ Toffoli (x_3, x_5, x_4)

3: $x_7 \leftarrow$ CNOT (x_4, x_7)

4: $x_6 \leftarrow$ CNOT (x_3, x_6)

5: $x_4 \leftarrow$ X (x_4)

6: $x_5 \leftarrow$ X (x_5)

7: $x_3 \leftarrow$ Toffoli (x_4, x_5, x_3)

8: $x_3 \leftarrow$ X (x_3)

9: $x_4 \leftarrow$ X (x_4) //reverse

10: $x_5 \leftarrow$ X (x_5) //reverse

11: $x_7 \leftarrow$ CNOT (x_5, x_7)

12: $x_4 \leftarrow$ Toffoli (x_5, x_6, x_4)

13: **return** $X(x_7, x_6, x_5, x_4, x_3)$

Additionally, we optimized the 3-qubit Sbox as follows. In the OR quantum gate in Figure 5, X gates are used to prepare input values (X, Y), perform the reverse operation (X, Y), and compute the result (Z). We minimized the use of X gates in the 3-qubit Sbox. In the 3-bit Sbox of Algorithm 1, the OR operation is used twice. In this case, the part using X gates for the output of the first OR (i.e., reverse (x_2, x_1) and result (x_0)) and the part using X gates of the second OR (i.e., input (x_2, x_0), result (x_1)) overlap each other. Therefore, the X gates in the overlapping part can be omitted. Lastly, since the NOT operation is performed on x_2 in the 3-bit Sbox of Algorithm 1, x_2 of the reverse(x_2, x_0) in the second OR can be omitted. X gates, which are omitted by overlapping each other, are marked with the same color.

In Algorithm 1, extend XOR is performed after 5-qubit and 3-qubit Sbox operations. The quantum circuit for extend XOR is simple and is shown in Equation (2).

Algorithm 4 The 3-qubit Sbox in quantum circuits.

Input: 3-qubit $X(x_2, x_1, x_0)$

Output: 3-qubit $X(x_2, x_1, x_0)$

1: $x_2 \leftarrow$ Toffoli (x_1, x_0, x_2)

2: $x_2 \leftarrow$ X (x_2)

3: $x_1 \leftarrow$ X (x_1)

4: $x_0 \leftarrow$ Toffoli (x_2, x_1, x_0)

5: $x_0 \leftarrow$ X (x_0)

6: $x_2 \leftarrow$ X (x_2) //reverse

7: $x_1 \leftarrow$ X (x_1) //reverse

8: $x_2 \leftarrow$ X (x_2)

9: $x_0 \leftarrow$ X (x_0)

10: $x_1 \leftarrow$ Toffoli (x_2, x_0, x_1)

11: $x_1 \leftarrow$ X (x_1)

12: $x_2 \leftarrow$ X (x_2) //reverse

13: $x_0 \leftarrow$ X (x_0) //reverse

14: $x_2 \leftarrow$ X (x_2)

15: **return** $X(x_2, x_1, x_0)$

$$\text{CNOT}(x_1, x_7), \quad \text{CNOT}(x_2, x_3), \quad \text{CNOT}(x_0, x_4) \tag{2}$$

3.4.2. Quantum Circuit Design for 5-Qubit Sbox_2

As mentioned above, the inefficient part when implementing PIPO Sbox as a quantum circuit is the 5-bit Sbox_2. In Figure 3, 3-bit of Sbox_2's output (5-bit) is XORed to the output of 3-bit Sbox, and the remaining 2-bit is the final result value. However, the input 3-bit of Sbox_2 is also the final result. Therefore, in the quantum circuit, before 3-qubit is entered into Sbox_2, 3 temp qubits to store input 3-qubit must be newly allocated. If only an additional 3 qubits are required to complete the PIPO encryption, this is working properly. However, every time the Sbox runs, it needs to allocate 3 qubits. For example, after allocating 3 additional temp qubits, we store the input 3-qubit in the temp qubits and use the temp qubits as the input for Sbox_2. Then, the output 2-qubit of Sbox_2 is used as the result value, but the remaining 3 qubits are XORed and are no longer needed. Unfortunately, as shown in Section 2.2, these 3 qubits cannot be initialized to zero. They become garbage qubits. We need to allocate 3 qubits for every Sbox.

However, we implemented the PIPO Sbox quantum circuit without additional qubits by using two new Sboxes (i.e., $\text{Sbox}_{\text{new1}}$, $\text{Sbox}_{\text{new2}}$) with the modified operation of Sbox_2 and the reverse operation. Equation (3) is the operation of Sbox_2 in Algorithm 1. $t_0(x_7)$, $t_1(x_3)$, and $t_2(x_4)$ are 3 bits that must be XORed to the 3-bit Sbox output. $\text{Sbox}_{\text{new1}}$ generates only 3-qubit to be XORed to the output of 3-qubit Sbox. Operations marked in red have no

effect on t_0, t_1, and t_2. Therefore, the proposed $\text{Sbox}_{\text{new1}}$ is optimized by excluding the red operations. This is shown in Algorithm 5.

$$\begin{aligned}
&t_0 \leftarrow x_7, \;\; t_1 \leftarrow x_3, \;\; t_2 \leftarrow x_4 \\
&x_6 \leftarrow x_6 \oplus (t_0 \;\&\; x_5) \\
&t_0 \leftarrow t_0 \oplus x_6 \\
&x_6 \leftarrow x_6 \oplus (t_2 \;\vee\; t_1) \\
&t_1 \leftarrow t_1 \oplus x_5 \\
&x_5 \leftarrow x_5 \oplus (x_6 \;\vee\; t_2) \\
&t_2 \leftarrow t_2 \oplus (t_1 \;\&\; t_0)
\end{aligned} \tag{3}$$

Algorithm 5 $\text{Sbox}_{\text{new1}}$ in quantum circuits.

Input: 5-qubit $X(x_7, x_6, x_5, x_4, x_3)$

Output: 3-qubit $X(x_7, x_4, x_3)$

1: $x_6 \leftarrow$ Toffoli (x_7, x_5, x_6)

2: $x_7 \leftarrow$ CNOT (x_6, x_7)

3: $x_3 \leftarrow$ CNOT (x_5, x_3)

4: $x_6 \leftarrow$ Toffoli (x_3, x_7, x_4)

5: **return** $X(x_7, x_4, x_3)$

After generating 3 qubits from $\text{Sbox}_{\text{new1}}$, XOR operation is performed to the output of 3-qubit Sbox (i.e., x_0, x_1, x_2), and then reverse operation of $\text{Sbox}_{\text{new1}}$ is performed. This is because x_7, x_4, and x_3 must be returned to values before $\text{Sbox}_{\text{new1}}$ to be the final values. After performing the reverse operation, $\text{Sbox}_{\text{new2}}$ is executed.

$\text{Sbox}_{\text{new2}}$ receives 5-qubit, and 2 qubits become final result values, but 3 qubits maintain their values as they are entered. Fortunately, this is possible because if we change operations of PIPO Sbox_2, we can generate the final result 2-bit, but it keeps the other 3-bit unchanged. Excluding the blue-marked operations of Equation (3), it is possible to generate 2-bit (x_5, x_6) while maintaining 3-bit (x_7, x_4, x_3). In the proposed $\text{Sbox}_{\text{new2}}$, the OR operation is used twice like a 3-qubit Sbox. The overlapping part (i.e., same color) is omitted to optimize. This is shown in Algorithm 6.

In Algorithm 1, bit swap operations are performed. This can be done with quantum swap gates. The quantum swap gate changes values of two target qubits to each other [17]. The implementation of quantum swap gates for swaps in Algorithm 1 is shown in Equation (4). The use of these four swap gates does not need to be measured in quantum resources by relabeling the qubits. This is described in detail in Section 3.5.

$$\text{Swap}(x_7, x_0), \quad \text{Swap}(x_7, x_1), \quad \text{Swap}(x_3, x_6), \quad \text{Swap}(x_4, x_5) \tag{4}$$

Algorithm 6 $\text{Sbox}_{\text{new2}}$ in quantum circuits.

Input: 5-qubit $X(x_7, x_6, x_5, x_4, x_3)$

Output: 5-qubit $X(x_7, x_6, x_5, x_4, x_3)$

1: $x_6 \leftarrow$ Toffoli (x_7, x_5, x_6)

2: $x_4 \leftarrow$ X (x_4)

3: $x_3 \leftarrow$ X (x_3)

4: $x_6 \leftarrow$ Toffoli (x_4, x_3, x_6)

5: $x_6 \leftarrow$ X (x_6)

6: $x_4 \leftarrow$ X (x_4) //reverse

7: $x_3 \leftarrow$ X (x_3) //reverse

8: $x_6 \leftarrow$ X (x_6)

9: $x_4 \leftarrow$ X (x_4)

10: $x_5 \leftarrow$ Toffoli (x_6, x_4, x_5)

11: $x_5 \leftarrow$ X (x_5)

12: $x_6 \leftarrow$ X (x_6) //reverse

13: $x_4 \leftarrow$ X (x_4) //reverse

14: **return** $X(x_7, x_6, x_5, x_4, x_3)$

3.4.3. Quantum Circuit Design for PIPO Sbox

The proposed PIPO Sbox quantum circuit consists of four steps, as shown in Figure 10. In Step 1, the input 8-qubit is divided into 5-qubit and 3-qubit, and each Sbox is executed. Then, XOR the output of the 3-qubit Sbox to the 5-qubit Sbox output. In Step 2, $\text{Sbox}_{\text{new1}}$ is executed and the output 3-qubit is XORed to the right 3-qubit line. As mentioned above, $\text{Sbox}_{\text{new1}}$ is optimized to generate only this 3-qubit. In Step 3, the reverse operation of $\text{Sbox}_{\text{new1}}$ is performed. Then, $\text{Sbox}_{\text{new1}}$ (reverse) output returns to the state before it is entered into $\text{Sbox}_{\text{new1}}$. In Step 4, $\text{Sbox}_{\text{new2}}$ is executed, which generates a final result value of 2-qubit and maintains the input value of 3-qubit.

We designed $\text{Sbox}_{\text{new1}}$ and $\text{Sbox}_{\text{new2}}$ by dividing the roles of 5-bit Sbox_2 and utilized reverse operation. As a result, we implemented an efficient quantum PIPO Sbox without qubits for the temp storage.

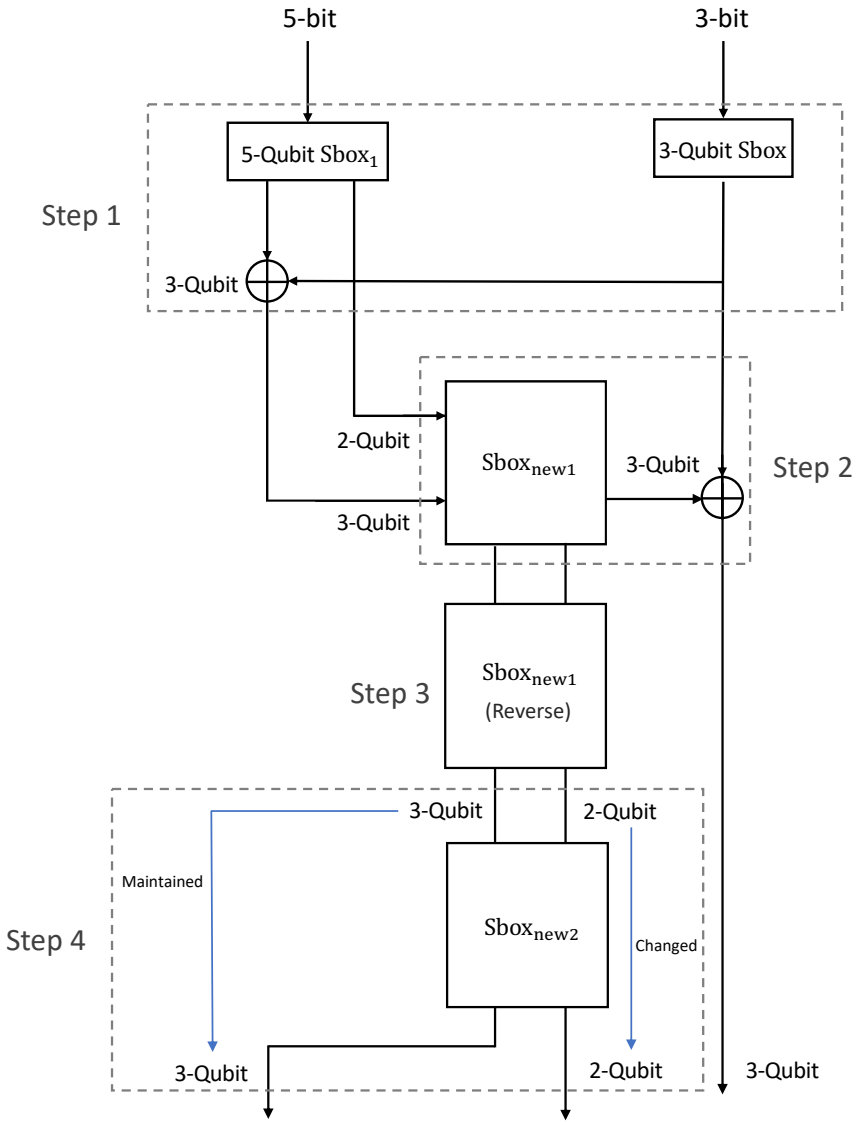

Figure 10. PIPO Sbox in quantum circuits.

3.5. Permutation of PIPO Block Cipher

Quantum resources are not used for the Permutation layer. In the Permutation layer, only rotation operations that change the position of qubits are used. This can be done with swap gates, but it can also be replaced by relabeling the qubits [18] without swap gates. Therefore, the use of swap gates does not have to be measured as quantum resources. This approach also applies to swap gates used in Sbox. Relabeling qubits for PIPO Permutation is detailed in Algorithm 7.

Algorithm 7 Relabeling qubits for block B.

Input: 64-qubit block $B(B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0)$

Output: 64-qubit block $B(B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0)$

1: $B_1(b_{15}, b_{14}, b_{13}, b_{12}, b_{11}, b_{10}, b_9, b_8) \leftarrow (b_8, b_{15}, b_{14}, b_{13}, b_{12}, b_{11}, b_{10}, b_9)$

2: $B_2(b_{23}, b_{22}, b_{21}, b_{20}, b_{19}, b_{18}, b_{17}, b_{16}) \leftarrow (b_{19}, b_{18}, b_{17}, b_{16}, b_{23}, b_{22}, b_{21}, b_{20})$

3: $B_3(b_{31}, b_{30}, b_{29}, b_{28}, b_{27}, b_{26}, b_{25}, b_{24}) \leftarrow (b_{28}, b_{27}, b_{26}, b_{25}, b_{24}, b_{31}, b_{30}, b_{29})$

4: $B_4(b_{39}, b_{38}, b_{37}, b_{36}, b_{35}, b_{34}, b_{33}, b_{32}) \leftarrow (b_{33}, b_{32}, b_{39}, b_{38}, b_{37}, b_{36}, b_{35}, b_{34})$

5: $B_5(b_{47}, b_{46}, b_{45}, b_{44}, b_{43}, b_{42}, b_{41}, b_{40}) \leftarrow (b_{42}, b_{41}, b_{40}, b_{47}, b_{46}, b_{45}, b_{44}, b_{43})$

6: $B_6(b_{55}, b_{54}, b_{53}, b_{52}, b_{51}, b_{50}, b_{49}, b_{48}) \leftarrow (b_{54}, b_{53}, b_{52}, b_{51}, b_{50}, b_{49}, b_{48}, b_{55})$

7: $B_7(b_{63}, b_{62}, b_{61}, b_{60}, b_{59}, b_{58}, b_{57}, b_{56}) \leftarrow (b_{61}, b_{60}, b_{59}, b_{58}, b_{57}, b_{56}, b_{63}, b_{62})$

8: **return** $B(B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0)$

4. Evaluation

We implemented the PIPO block cipher in quantum circuits using the IBM ProjectQ, a quantum programming tool provided by IBM. The ProjectQ provides a variety of compilers such as Classical–Simulator, Resource–Counter, and Circuit–Drawer. We used the Classical–Simulator to verify that the PIPO block cipher in the quantum circuit was implemented correctly and the Resource–Counter to analyze the quantum resources and circuit depth required for the PIPO quantum circuit. Based on this, quantum resources required for proposed PIPO quantum circuits and those required for other block cipher quantum circuits were compared, as shown in Table 2.

Table 2. Quantum resources required for PIPO quantum circuit implementation and comparison with other block ciphers.

Quantum Circuit	Qubits	Toffoli Gates	CNOT Gates	X Gates	Depth
PIPO-64/128	192	1248	2248	1477	248
PIPO-64/256	320	1632	2920	1930	324
SIMON-64/128 [8]	192	1408	7396	1216	2643
SPECK-64/128 [11]	193	3286	9238	57	-
CHAM-64/128 [10]	196	2400	12,285	240	-
HIGHT-64/128 [10]	201	6272	20,523	4	-
GIFT-64/128 [12]	192	1792	1792	3261	308

4.1. Analysis of Quantum Resources for PIPO and Other Block Ciphers

We focused on minimizing qubits and achieved the optimal number of qubits by allocating qubits for the plaintext and master key only. The circuit depth is related to execution time [19], and the circuit depth of PIPO block cipher is very low. This is possible because many of quantum gates of the PIPO quantum circuit are performed in parallel. In the proposed PIPO quantum circuit, most of the resources are used for Sbox. When implementing the block cipher of the SPN structure as a quantum circuit, it is important to optimize the Sbox operation, and this was achieved through the proposed Sbox implementation. In addition, all other operations of PIPO were optimized as much as possible.

Compared with block ciphers using the same plaintext size (i.e., 64-bit) and key size (i.e., 128-bit), the PIPO block cipher achieves an optimal number of qubits, and it has the

lowest circuit depth and quantum gate complexity. Table 2 does not show circuit depths of SPECK, CHAM, and HIGHT, but they are higher than SPECK.

By analyzing the quantum circuits and resources of Anand et al. [8], Jang et al. [11], the hardware-optimized block cipher (i.e., SIMON) was better optimized in quantum computers than in the software-optimized block cipher (i.e., SPECK). In Reference [12], the authors obtained optimal quantum resources by implementing a hardware-optimized operation of the GIFT Sbox operation as a quantum circuit. Through this, we confirmed that operations optimized for hardware are also optimized for quantum computers.

4.2. Resource Estimation for Using the Grover Search Algorithm to PIPO

For the block cipher of the n-bit security level, a maximum of 2^n queries are required to recover the key by the classical brute force attack. However, in a quantum brute force attack using the Grover's algorithm, the key can be recovered with only a maximum of $2^{\frac{n}{2}}$ queries.

The Grover search algorithm iterates over the oracle and diffusion operator. When applying the Grover search algorithm to the key search of the block cipher, the quantum circuit of the block cipher is implemented in the oracle and finds the key. Since the diffusion operator only increases the amplitude of the answer returned by the oracle, the required quantum resources are determined according to how the oracle is implemented.

In Reference [20], a block cipher key search using the Grover search algorithm requires r pairs of known plaintext and ciphertext ($r = \frac{keysize}{blocksize}$). That is, the PIPO-64/128 quantum circuit should be operated two times, and the PIPO-64/256 quantum circuit should be operated four times. In addition, the oracle of the Grover search algorithm needs the reverse operation. Therefore, it requires four times resources used in the PIPO-64/128 quantum circuit and eight times resources used in the PIPO-64/256 quantum circuit. We assume that the plaintext-ciphertext pairs was operated in parallel [20], which requires $2 \times$ key size $\times (r-1)$ additional CNOT gates including reverse operation [5]. Finally, quantum resources required for the oracle of the Grover search algorithm are as follows.

$$\begin{aligned} &\text{Qubits} \leftarrow \text{Qubits required for quantum circuit} \cdot r + 1 \\ &\text{Toffoli gates} \leftarrow \text{Toffoli gates required for quantum circuit} \cdot 2 \cdot r \\ &\text{CNOT gates} \leftarrow \text{CNOT gates required for quantum circuit} \cdot 2 \cdot r + 2 \cdot \text{key size} \cdot (r-1) \\ &\text{X gates} \leftarrow \text{X gates required for quantum circuit} \cdot 2 \cdot r \end{aligned} \tag{5}$$

Through this, quantum resources for applying PIPO block cipher and other ciphers to the oracle of the Grover search algorithm are shown in Table 3.

Table 3. Quantum resources for applying PIPO block cipher to the oracle of the Grover's search algorithm.

Block Cipher	r	Qubits	Toffoli Gates	CNOT Gates	X Gates
PIPO-64/128	2	385	4992	9248	5908
PIPO-64/256	4	1281	130,562	24,896	15,440
SPECK-64/128 (Extrapolation) [11]	2	387	13,144	37,208	228
GIFT-64/128 [12]	2	385	7168	7424	13,044

5. Conclusions

We implemented and optimized the SPN structured block cipher (i.e., PIPO) as a quantum circuit. Optimal qubits, quantum gates, and circuit depth are achieved, and the proposed PIPO quantum circuit is the most compact compared to research results of other block ciphers. When using the Grover search algorithm to the key search of the block cipher, the block cipher quantum circuit is implemented in the oracle. Therefore, a compact Grover search algorithm can be applied to the key search for PIPO block cipher through the proposed quantum circuit. Additionally, the proposed method, which optimizes various cryptographic operations as quantum circuits, can be an interesting

point in the upcoming quantum computer era. In the future, we plan to implement another variety of lightweight block ciphers in quantum circuits, including candidate from the NIST lightweight cryptography standardization https://csrc.nist.gov/projects/lightweight-cryptography (accessed on 13 May 2021). As in the case of PIPO Sbox, we expected that some operations that are not cost-critical in classical computers may be expensive when implemented in quantum computers. Moreover, these features found in implementing various block ciphers will be an interesting direction to pursue. By utilizing these features, if a new block cipher that requires a lot of quantum resources for implementation were developed, it could become a quantum resistant block cipher.

Author Contributions: Data curation, K.J.; Investigation, G.S., H.K. (Hyeokdong Kwon) and S.U.; Software, K.J. and W.-K.L.; Supervision, H.S.; Writing—original draft, K.J.; Writing—review and editing, K.J., H.K. (Hyunji Kim) and H.S. All authors have read and agreed to the published version of the manuscript.

Funding: 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) and 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). This research was financially supported by Hansung University for Hwajeong Seo.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Grover, L.K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*; ACM: New York, NY, USA, 1996; pp. 212–219.
2. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
3. Bogdanov, A.; Khovratovich, D.; Rechberger, C. Biclique cryptanalysis of the full AES. In *International Conference on the Theory and Application of Cryptology and Information Security*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 344–371.
4. Tao, B.; Wu, H. Improving the biclique cryptanalysis of AES. In *Australasian Conference on Information Security and Privacy*; Springer: Cham, Switzerland, 2015; pp. 39–56.
5. Grassl, M.; Langenberg, B.; Roetteler, M.; Steinwandt, R. Applying Grover's algorithm to AES: Quantum resource estimates. In *Post-Quantum Cryptography*; Springer: Cham, Switzerland, 2016; pp. 29–43.
6. Langenberg, B.; Pham, H.; Steinwandt, R. Reducing the Cost of Implementing AES as a Quantum Circuit. *IEEE Trans. Quantum Eng.* **2020**, *1*, 1–12. [CrossRef]
7. Jaques, S.; Naehrig, M.; Roetteler, M.; Virdia, F. Implementing Grover oracles for quantum key search on AES and LowMC. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Cham, Switzerland, 2020; pp. 280–310.
8. Anand, R.; Maitra, A.; Mukhopadhyay, S. Grover on SIMON. *Quantum Inf. Process.* **2020**, *19*, 1–17. [CrossRef]
9. Schlieper, L. In-place implementation of Quantum-Gimli. *arXiv* **2020**, arXiv:2007.06319.
10. Jang, K.; Choi, S.; Kwon, H.; Kim, H.; Park, J.; Seo, H. Grover on Korean Block Ciphers. *Appl. Sci.* **2020**, *10*, 6407. [CrossRef]
11. Jang, K.; Choi, S.; Kwon, H.; Seo, H. Grover on SPECK: Quantum Resource Estimates. Cryptology ePrint Archive, Report 2020/640, 2020. Available online: https://eprint.iacr.org/2020/640 (accessed on 13 May 2021).
12. Jang, K.; Kim, H.; Eum, S.; Seo, H. Grover on GIFT. Cryptology ePrint Archive, Report 2020/1405, 2020. Available online: https://eprint.iacr.org/2020/1405 (accessed on 13 May 2021).
13. Kim, H.; Jeon, Y.; Kim, G.; Kim, J.; Sim, B.Y.; Han, D.G.; Seo, H.; Kim, S.; Hong, S.; Sung, J.; et al. PIPO: A Lightweight Block Cipher with Efficient Higher-Order Masking Software Implementations. In *International Conference on Information Security and Cryptology*; Springer: Cham, Switzerland, 2020; pp. 99–122.
14. Dasu, V.A.; Baksi, A.; Sarkar, S.; Chattopadhyay, A. LIGHTER-R: Optimized Reversible Circuit Implementation For SBoxes. In Proceedings of the 2019 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 3–6 September 2019; pp. 260–265. [CrossRef]
15. Jean, J.; Peyrin, T.; Sim, S.M.; Tourteaux, J. Optimizing Implementations of Lightweight Building Blocks. *IACR Trans. Symmetric Cryptol.* **2017**, *2017*, 130–168. [CrossRef]
16. Steiger, D.S.; Häner, T.; Troyer, M. ProjectQ: An open source software framework for quantum computing. *Quantum* **2018**, *2*, 49. [CrossRef]
17. Garcia-Escartin, J.C.; Chamorro-Posada, P. A SWAP gate for qudits. *Quantum Inf. Process.* **2013**, *12*, 3625–3631. [CrossRef]

18. Znidaric, M.; Giraud, O.; Georgeot, B. How many CNOT gates does it take to generate a three-qubit state? *arXiv* **2007**, arXiv:0711.4021.
19. Bhattacharjee, D.; Chattopadhyay, A. Depth-Optimal Quantum Circuit Placement for Arbitrary Topologies. *arXiv* **2017**, arXiv:1703.08540.
20. Amento-Adelmann, B.; Grassl, M.; Langenberg, B.; Liu, Y.K.; Schoute, E.; Steinwandt, R. Quantum cryptanalysis of block ciphers: A case study. In Proceedings of the Poster at Quantum Information Processing QIP, Delft, The Netherlands, 15–19 January 2018; pp. 235–243.