Optimized Quantum Implementation of SEED

Yujin Oh, Kyungbae Jang, Yujin Yang, and Hwajeong Seo

Division of IT Convergence Engineering, Hansung University, Seoul, South Korea

oyj0922@gmail.com, starj1023@gmail.com, yujin.yang34@gmail.com, hwajeong84@gmail.com

Abstract. With the advancement of quantum computers, it has been demonstrated that Shor's algorithm enables public key cryptographic attacks to be performed in polynomial time. In response, NIST conducted a Post-Quantum Cryptography Standardization competition. Additionally, due to the potential reduction in the complexity of symmetric key cryptographic attacks to square root with Grover's algorithm, it is increasingly challenging to consider symmetric key cryptography as secure. In order to establish secure post-quantum cryptographic systems, there is a need for quantum post-quantum security evaluations of cryptographic algorithms. Consequently, NIST is estimating the strength of post-quantum security, driving active research in quantum cryptographic analysis for the establishment of secure post-quantum cryptographic systems.
In this regard, this paper presents a depth-optimized quantum circuit implementation for SEED, a symmetric key encryption algorithm included in the Korean Cryptographic Module Validation Program (KCMVP). Building upon our implementation, we conduct a thorough assessment of the post-quantum security for SEED. Our implementation for SEED represents the first quantum circuit implementation for this cipher.

Keywords: Quantum Circuit · SEED · Korean Block Cipher · Grover Algorithm.

1 Introduction

Quantum computers, developed using techniques from quantum mechanics like quantum superposition and entanglement, have the ability to solve specific problems faster than classical computers. Quantum computers are being developed by many companies and research teams. The introduction of the Shor algorithm [1], which is known for its ability to solve the integer factorization problem and the discrete logarithm problem in polynomial time, poses significant risks to public-key cryptography designed based on these problems. Similarly, the Grover search algorithm [2], known for its ability to reduce the complexity of data search by a square root factor, can have a significant impact on the security of symmetric key cryptography.

NIST has proposed criteria for estimating the quantum attack complexity on the AES family and a parameter called MAXDEPTH, which represents the maximum circuit depth that a quantum computer can execute, in its evaluation

criteria document for post-quantum cryptography standardization [3, 4]. Both of these aspects need to be considered to evaluate the quantum security strength of a cipher. Detailed explanations on these topics will be provided in Section 2.4, 5.

Based on these NIST criteria, continuous efforts have been made to estimate the complexity of Grover's key search for symmetric-key ciphers and evaluate post-quantum security [5, 6, 7]. In addition to AES, research has also been conducted on estimating quantum resources for well-known lightweight block ciphers such as SPECK, GIFT, and PRESENT [8, 9, 10], as well as lightweight block ciphers selected as finalists in the Lightweight Cryptography (LWC) competition, including SPARKLE [11, 12] and ASCON [13, 14].

In this paper, we propose an optimized quantum circuits for SEED, which is a symmetric key encryption algorithms included as validation subjects in the Korean Cryptographic Module Validation Program (KCMVP). Since these cryptographic algorithms are widely used in cryptographic modules in Korea, it is of great importance to estimate quantum resources and measure the quantum security strength of these ciphers. Using the proposed quantum circuit as a basis, we assess the post-quantum security strength of SEED in accordance with NIST criteria.

1.1 Our Contribution

The contribution in this paper is manifold and can be summarized as follows:

1. **Quantum Circuit Implementation of SEED.** We demonstrate the first implementation of a quantum circuit for SEED, which is the one of Korean cipher.
2. **Low-Depth Implementation of SEED.** In our quantum circuit implementation of SEED, we focus to optimize a low Toffoli depth and full depth. We implement the Itoh-Tsujii algorithm for S-box optimization. For the implementation, we utilize the WISA'22 quantum multiplication, and a squaring based on PLU factorization. Further, we enhance the efficiency of depth optimization by using an optimal quantum adder(which is called CDKM adder) and implementing parallelization for applicable components.
3. **Post-quantum Security Assessment of SEED.** We estimate the cost of Grover's key search using an our implemented quantum circuit for SEED in order to assess the quantum security of SEED. During this assessment, we compare the estimated cost of Grover's key search for SEED with the security levels defined by NIST.

2 Preliminaries

2.1 SEED Block Cipher

SEED is a block cipher of Feistel structure operates on 128-bit block and 128-bit key. It consists of 16 rounds and each round has a round function F.

A 128-bit block is divided into 64-bit blocks, and the right 64-bit block (R_0) serves as the input to the F function with 64-bit round key. The output of F function is XORed to the left 64-bit block (L_0). The overall structure of SEED cipher is shown in Figure 1.

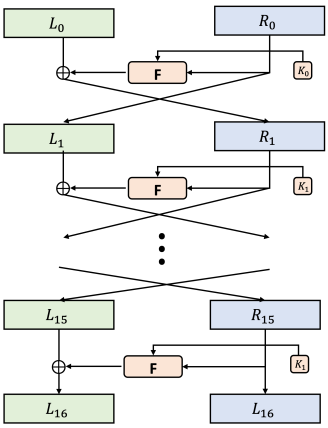

Fig. 1: Overall structure of SEED cipher.

***F* Function** The input of F function (Figure 2) is 64-bit block and 64-bit round key $RK_i = (K_{i,0}, K_{i,1})$. The 64-bit block is divided into two 32-bit blocks (C, D) and each block is XORed with the round key. The F function consists of XOR operations ($\oplus$), modular additions ($\boxplus$), and G functions.

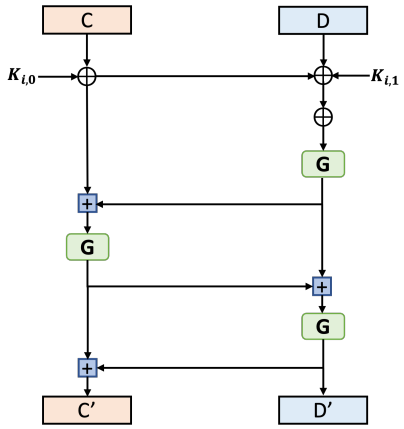

Fig. 2: The process of the F function.

***G* Function** The 32-bit input block of the G function (Figure 3) is divided into 8-bit blocks (X_{0-3}) and each block becomes input for the S-boxes.

To compute the output of an S-box, it involves exponentiation of $x \in \mathbb{F}_{2^8}/(x^8 + x^6 + x^5 + x + 1)$, matrix-vector multiplication, and XORing a single constant. Specifically, two distinct S-boxes (S_1 and S_2) are employed, each using its own corresponding set of matrices ($A^{(1)}$ or $A^{(2)}$), exponentiation values (x^{247} or x^{251}), and constant values (169 or 56), which are as follows:

$$S_1(x) = A^{(1)} \cdot x^{247} \oplus 169, \;\; S_2(x) = A^{(2)} \cdot x^{251} \oplus 56 \tag{1}$$

The output values of the S-boxes are ANDed (&) with the constants m_{0-3}, and the results of these AND operations are XORed with each other to compute the final output (i.e., Z_{0-3}).

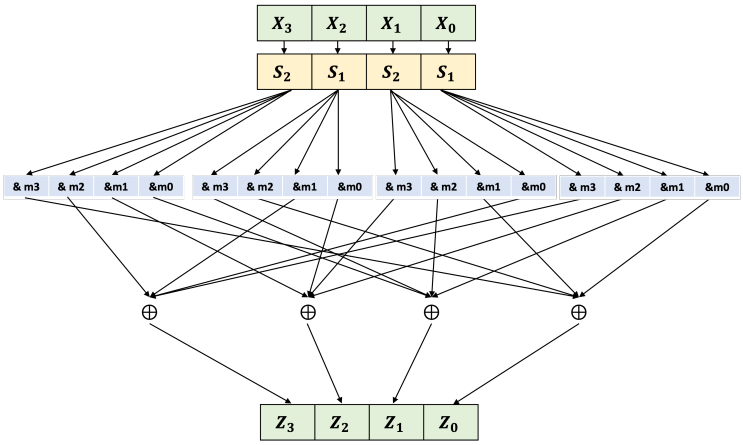

Fig. 3: The process of the G function.

Key Schedule In the key schedule (Figure 4), the 128-bit key is divided into four blocks ($A\|B\|C\|D$, where $\|$ denotes concatenation), and operations such as shift ($\gg$, $\ll$), modular addition, modular subtraction ($\boxminus$), and G function are applied.

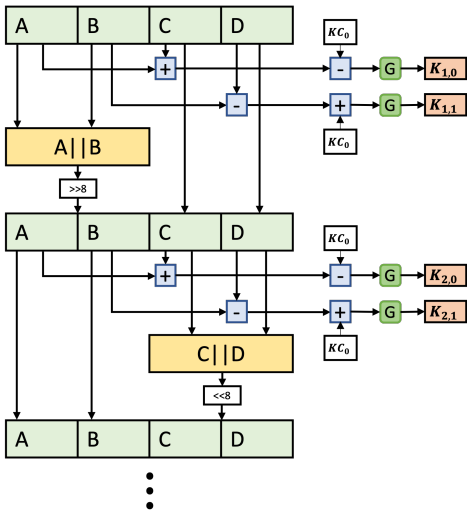

Fig. 4: The process of the key schedule

2.2 Quantum Gates

This section describes commonly used quantum gates (Figure 5) for implementing quantum circuits of block ciphers (note that this is not an exhaustive list of all possible gates that can be used).

The X gate acts like a NOT operation on a classical computer, reversing the state of the qubit that goes through it. The Swap gate exchanges the states of two qubits. The CNOT gate behaves like an XOR operation on a classical computer. In CNOT(a, b), the input qubit a is the control qubit, and b is the target qubit. When the control qubit a is in the state 1, the target qubit b is flipped. As a result, the value of $a \oplus b$ is stored in the qubit b (i.e., $b = a \oplus b$), while the state of qubit a remains unchanged. The Toffoli gate, represented as Toffoli(a, b, c), acts like an AND operation on a classical computer. It requires three input qubits, with the first two qubits (a and b) serving as control qubits. Only when both control qubits are in the state 1, the target qubit c is flipped. The result of the operation a & b is XORed with the qubit c (i.e., $c = c \oplus (a$ & $b)$), while the states of qubits a and b are preserved.

2.3 Grover's Key Search

Grover's algorithm searches for a specific data from an unsorted set of N with a search complexity of $O(\sqrt{N})$. In cryptography, for an encryption scheme that uses a k-bit key, a classical computer requires a search of $O(2^k)$ complexity for exhaustive key search. However, using Grover's algorithm, a quantum computer can perform this search with a complexity of only $O(\sqrt{2^k})$, which is reduced by a square root. In this section, we divide the progress of Grover's key search into

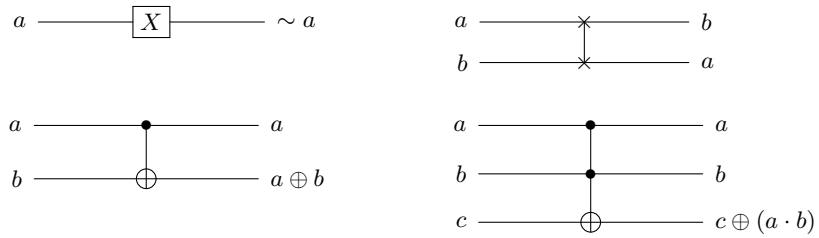

Fig. 5: Quantum gates: X (left top), Swap (right top), CNOT (left bottom) and Toffoli (right bottom) gates.

three stages: *Input Setting*, *Oracle*, and *Diffusion Operator*, and describe them as follows.

1. *Input Setting*: Prepare a k-qubit key in a superposition state using Hadamard gates. In this case, equal amplitudes are generated for all 2^k possible states.

$$H^{\otimes k} |0\rangle^{\otimes k} = |\psi\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{1}{2^{k/2}} \sum_{x=0}^{2^k-1} |x\rangle \tag{2}$$

2. In the *oracle*, the target encryption algorithm is implemented through a quantum circuit. This circuit encrypts a known plaintext in a superposition state using a pre-prepared key (as set in the input setting), producing ciphertexts for every possible key value. Subsequently, these generated ciphertexts are compared with the known ciphertexts. Upon discovering a match (i.e., when $f(x) = 1$ in Equation (3)), the sign of the desired key state to be recovered is negated according to when $f(x) = -1$ in Equation (4). Finally, the implemented quantum circuit reverses the generated ciphertexts back to the known plaintext for the next iteration.

$$f(x) = \begin{cases} 1 \text{ if } Enc_{key}(p) = c \\ 0 \text{ if } Enc_{key}(p) \neq c \end{cases} \tag{3}$$

$$U_f(|\psi\rangle |-\rangle) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle |-\rangle \tag{4}$$

3. The *Diffusion Operator* serves to amplify the amplitude of the target key state indicated by the oracle, identifying it by flipping the sign of said amplitude to negative. The quantum circuit for the diffusion operator is typically straightforward and does not require any special techniques to implement. Additionally, the overhead of the diffusion operator is usually negligible compared to the oracle, and therefore it is generally ignored when estimating the cost of Grover's algorithm [5,7,15]. Lastly, the Grover's algorithm provides a high probability of measuring the solution key by performing a sufficient number of iterations of the oracle and the diffusion operator to amplify the amplitude of the target key state.

2.4 NIST Security Criteria

NIST establishes security levels and estimates the required resources for block cipher and hash function attack costs for post-quantum security [3]. The estimates provided by NIST for the security levels defined and the number of classical and quantum gates for the attacks are as follows:

- **Level 1**: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a key search on a 128-bit key block cipher, such as AES128. ($\mathbf{2^{170} \rightarrow 2^{157}}$).
- **Level 2**:Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a collision search on a 256-bit hash function , such as SHA256/ SHA3-256.
- **Level 3**: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a key search on a 192-bit key block cipher, such as AES192. ($\mathbf{2^{233} \rightarrow 2^{221}}$).
- **Level 4**: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a collision search on a 384-bit hash function , such as SH3A846/ SHA3-384.
- **Level 5**: Any attempt to breach the defined security standards should necessitate computational capabilities equal to or greater than those required to perform a key search on a 256-bit key block cipher, such as AES256. ($\mathbf{2^{298} \rightarrow 2^{285}}$).

Level 1, 3, and 5 are based on the Grover's key search cost for AES, while Level 2 and 4 rely on the collision attack cost for SHA3. Additionally, for Levels 2 and 4, estimates are provided only for classical gates, not quantum attacks. In our implementation of SEED, which is a symmetric key cipher, we primarily focus on Levels 1, 3, and 5.

NIST sets the Grover's key search cost for AES-128, 192, and 256 based on the quantum circuits implemented by Grassl et al. [5], resulting in Levels 1, 3, and 5. During the execution of the Grover's key search, the number of gates and depth continue to increase, while the number of qubits remains constant. Therefore, the estimates provided by NIST are derived from the product of the total gates and total depth of the quantum circuit, excluding the number of qubits.

The estimates for Grover's key search on the quantum circuit from [5], which NIST used as a basis for setting security levels, are notably high. Subsequent efforts to optimize AES quantum circuits have led to a reduction in the cost of quantum attacks. In 2019, Jaques et al. presented optimized quantum circuits for AES at Eurocrypt '20 [16]. Based on this, NIST redefines the quantum attack costs for AES-128, 192, and 256 as $\mathbf{2^{157}}$, $\mathbf{2^{221}}$, $\mathbf{2^{285}}$, respectively [4].

Moreover, NIST proposes a restriction on circuit depth known as MAXDEPTH. This restriction stems from the challenge of executing highly prolonged sequential computations. In other words, it arises from the challenge of prolonged calculations due to sequential repetitions of quantum circuits in Grover's key search (especially in the Grover oracle). The MAXDEPTH specified by NIST is as follows. ($2^{40} < 2^{64} < 2^{96}$)

3 Proposed Quantum Implementation of SEED

In this section, we present our optimized quantum circuit implementation of SEED. Our optimization goal in implementation is to minimize the depth while allowing a reasonable number of qubits.

3.1 Implementation of S-box

In quantum computers,the utilization of look-up table-based methods for implementing S-boxes is not appropriate. Thus, we employ quantum gates to implement the S-boxes based on *Boolean* expression of Equation 1. We use x^{247} or x^{251} in the S-box implementation, and these values can be expressed using primitive polynomials in $\mathbb{F}_{2^8}/(x^8+x^6+x^5+x+1)$ as follows: The S-boxes are in GF(2^8), so they can be modified with inversion as follows:

$$\begin{aligned} (x^{-1})^8 &\equiv x^{247} \mod p(x) \\ (x^{-1})^4 &\equiv x^{251} \mod p(x) \\ p(x) &= x^8+x^6+x^5+x+1 \end{aligned} \tag{5}$$

We can obtain the value by multiplying the inverse by the square. And then, following the Itoh Tsujii inversion algorithm [17], the x^{-1} can be computed:

$$x^{-1} = x^{254} = ((x \cdot x^2) \cdot (x \cdot x^2)^4 \cdot (x \cdot x^2)^{16} \cdot x^{64})^2 \tag{6}$$

To compute the inversion of x, squaring and multiplication are used (as shown in Equation 6). In squaring, by using PLU factorization, it can be implemented without allocating additional ancilla qubits (i.e., in-place), using only the CNOT gates. By applying the PLU factorization, we obtain the following:

$$\begin{pmatrix} 0&0&0&0&1&1&1&0 \\ 0&0&0&0&1&1&0&0 \\ 0&1&0&0&0&1&0&1 \\ 0&0&0&0&0&1&1&0 \\ 0&0&1&0&0&0&1&0 \\ 0&0&0&0&1&1&0&1 \\ 0&0&0&1&1&1&0&1 \\ 0&0&0&0&0&1&0&0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0 \end{pmatrix} \cdot \begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&0&0&1&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} 1&0&0&0&1&1&1&0 \\ 0&1&0&0&0&1&0&1 \\ 0&0&1&0&0&0&1&0 \\ 0&0&0&1&1&1&0&1 \\ 0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{pmatrix} \tag{7}$$

These three matrices consist of a permutation matrix, a lower triangular matrix, and an upper triangular matrix, respectively. Figure 6 demonstrates the implementation of quantum circuit of squaring using only CNOT gates, utilizing these three matrices.

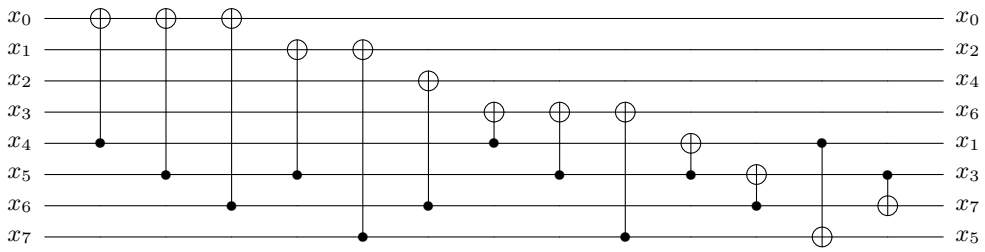

Fig. 6: Squaring in $\mathbb{F}_{2^8}/(x^8 + x^6 + x^5 + x + 1)$

For implementing multiplication in quantum, we adopt the method proposed in [18], which employs the Karatsuba multiplication instead of schoolbook multiplication [19]. The Karatsuba algorithm, when applied in the context of quantum computers, can lead to a reduction in the number of Toffoli gates (as it decrease the number of AND operations). This efficiency makes it a valuable technique for quantum computing.

In [18], a special Karatsuba algorithm is used, which enables the quantum multiplication with a Toffoli depth of one. By applying the Karatsuba algorithm recursively, all the AND operations for multiplication become independent. Additionally, by allocating more ancilla qubits, it becomes possible to operate all Toffoli gates in parallel, leading to a Toffoli depth of one.

Actually, allocating additional ancilla qubits is a known overhead in their method [18]. However, it is important to note that their method is more effective when used in conjunction with other operations rather than as a stand-alone multiplication. The authors of [18] mention that the ancilla qubits allocated for multiplication can be initialized (i.e., clean state) using reverse operations. This means that if it is not a stand-alone multiplication, the ancilla qubits can be reused in ongoing operations.

In Equation 6, multiple multiplications are performed to compute the inverse of the input x. Indeed, the method proposed in [18] is well-suited for implementing quantum circuits for inversion. Concretely, in our implementation, ancilla qubits are allocated only once for the initial multiplication ($x \cdot x^2$), and for subsequent multiplications, the initialized ancilla qubits are reused without incurring any additional cost.

As a result, we successfully optimize the number of qubits and the Toffoli-related metrics such as, the number of Toffoli gates, Toffoli depth, and full depth[1].

[1]The full depth is naturally reduced thanks to the reduction in the Toffoli depth.

Using these methods of squaring and multiplication, we can obtain the exponentiation values (x^{247} and x^{251}). And then, we compute the multiplication of the exponentiation values (x^{247} and x^{251}) and the matrices ($A^{(1)}$ and $A^{(2)}$). Since the matrices $A^{(1)}$ and $A^{(2)}$ are constant, the matrix-vector multiplication (classical-quantum) can be implemented in-place without requiring additional qubits. We apply the PLU factorization to the matrices $A^{(1)}$ and $A^{(2)}$, similar to how we implemented the quantum circuit for squaring (Equation 7).

3.2 Implementation of G function

In the G function, four S-boxes (two $S1$ and two $S2$) are used, and the implementation of these S-boxes follows the method described in Section 3.1. Each S-box requires 38 ancilla qubits, which can be initialized using reverse operations, enabling their reuse. Therefore, if the four S-boxes are implemented sequentially, the number of ancilla qubits can be saved by using only 38 of them. However, in this case, the depth of the circuit increases due to the sequential operations. Thus, considering the trade-off, we implement the four S-boxes in parallel to reduce the circuit depth. This is achieved by allocating a total of 152 (38×4) ancilla qubits at first. Additionally, these ancilla qubits are initialized (i.e., returning to 0), allowing the 4 sets of ancilla qubits to be reused in the G function of the next round.

3.3 Implementation of Key Schedule

Algorithm 1 describes the proposed quantum circuit implementation of the key schedule. In the key schedule, two 32-qubit subkeys ($K_{i,0}$ and $K_{i,1}$) are generated. To reduce the circuit depth, the implementation is parallelized by operating two processes simultaneously. For this, we allocate two sets of 152 ancilla qubits to implement two G functions in parallel. Also, for parallel processing, the operations with KeyConstant values of quantum state also need to be implemented in parallel. To enable parallel processing, two pairs of qubits (32×2) are allocated to store the KeyConstant values (using on $K_{i,0}, K_{i,1}$ respectively) in our implementation.

Due to the different KeyConstant values used in each round, it is necessary to allocate and store new qubits every time. Instead of allocating new qubits in each round, we utilize reverse operations to initialize and reuse the qubits. The reverse operation for the KeyConstant of quantum state involves only X gates, which have a trivial overhead on the circuit depth. Thanks to this approach, we can effectively parallelize the quantum circuit for the key schedule, resulting in a reduced circuit depth while using a reasonable number of qubits.

For implementing addition in quantum, we utilize the CDKM adder [20], an enhanced version of the quantum ripple-carry adder, which is implemented using X, CNOT, and Toffoli gates. The CDKM adder proves to be effective for n-qubit addition when $n \geq 4$, making it a suitable choice for SEED, where $n = 8$. This adder requires only one ancilla qubit and optimizes the circuit depth. Specifically, it utilizes one ancilla qubit, $(2n-3)$ Toffoli gates, $(5n-7)$ CNOT gates, $(2n-6)$ X gates, and achieves a circuit depth of $(2n+3)$.

In Shift operation, it can be implemented using swap gates, but in our approach, we utilize logical swaps that change the index of qubits, avoiding the use of quantum gates.

Algorithm 1: Quantum circuit implementation of SEED Key Schedule.

Input: A, B, C, D, c_0, c_1 $ancilla_0$, $ancilla_1$
Output: key_0, key_1, C, D
//Each operation in parallel.
for $0 \leq i \leq 16$ **do**
    $KC\_Q_0 \leftarrow$ Constant_XOR($KC[i]$, KC_Q_0)
    $KC\_Q_1 \leftarrow$ Constant_XOR($KC[i]$, KC_Q_1)

    $C_2 \leftarrow$ allocate new 32 qubits
    $D_2 \leftarrow$ allocate new 32 qubits

    $C_2 \leftarrow$ Copy32(C, C_2)
    $D_2 \leftarrow$ Copy32(D, D_2)

    $C_2 \leftarrow$ CDKM(A, C_2,c_0)
    $D_2 \leftarrow$ CDKM_minus(B, D_2, c_1)

    $C_2 \leftarrow$ CDKM_minus(KC_Q_0, C_2, c_0)
    $D_2 \leftarrow$ CDKM(KC_Q_1, D_2, c_1)

    $key_0 \leftarrow G$ function(C_2, $ancilla_0$)
    $key_1 \leftarrow G$ function(D_2, $ancilla_1$)

    //Initialize qubitsthrough reverse to reuse.
    $KC\_Q_0 \leftarrow$ Constant_XOR($KC[i]$, KC_Q_0)
    $KC\_Q_1 \leftarrow$ Constant_XOR($KC[i]$, KC_Q_1)

    if i % 2 == 0 **then**
        RightShift(A, B) ▷ logical Swap
    else
        LeftShift(C, D) ▷ logical Swap
return key_0, key_1, C, D

4 Performance of the Proposed Quantum Circuits

In this part, we present the performance of our SEED quantum circuit implementation. Our proposed quantum circuits of cryptographys are implemented using the ProjectQ tool provided by IBM. ProjectQ provides ClassicalSimulator, which can simulate simple quantum gates mentioned in Section 2.2, and ResourceCounter, which can measure circuit resources, as internal libraries. ClassicalSimulator has the advantage of providing enough quantum resources to run

our proposed quantum circuit. Real quantum computers still provide limited quantum resources that are not sufficient to run cryptography. Therefore, the circuits are run through the simulator provided by ProjectQ and the quantum resources are measured.

Table 1: Required quantum resources for SEED quantum circuit implementation

Cipher	#X	#CNOT	#Toffoli	Toffoli depth	#Qubit	Depth	TD-M cost
SEED	8116	409,520	41,392	321	41,496	11,837	13,320,216

Table 2: Required decomposed quantum resources for SEED quantum circuit implementation

Cipher	#Clifford	$\#T$	T-depth	#Qubit	Full depth	FD-M cost
SEED	748,740	289,680	1,284	41,496	34,566	1,434,350,736

Table 1 and 2 show the quantum resources required to implement our SEED quantum circuits. Table 1 provides a comprehensive analysis of quantum resources at the NCT (NOT, CNOT, Toffoli) level. The Toffoli gate can be decomposed into 8 Clifford gates and 7 T gates and Table 2 presents the decomposed quantum resource costs for the quantum circuit implementation of SEED. Additionally, our implementation focuses on optimizing the circuit depth while considering the trade-off for using qubit, and we also perform metrics to evaluate these trade-offs such as Toffoli depth $\times$ qubit count ($TD \times M$) and full depth $\times$ qubit count ($FD \times M$).

5 Evaluation of Grover's Search Complexity

We adopt the methodology detailed in Section 2.3 to estimate the cost of Grover's key search for SEED. Grover's search can be estimated based on our implemented SEED quantum circuit. Since the overhead of the diffusion operator can be considered insignificant compared to the oracle when most of the quantum resources are used for implementing the target cipher in the quantum circuit, it can be disregarded.

Additionally, the Grover oracle is comprised of two consecutive executions of the SEED quantum circuit. The first one constitutes the encryption circuit, while the second one is the reverse operation of encryption circuit to return back to the state prior to encryption. Therefore, the oracle requires twice the cost of implementing a quantum circuit, not including of qubits. The number of iterations of Grover key search for k-bit key length is about $\sqrt{2^k}$. In [21], Grover's key search algorithm was analyzed in detail and the optimal number of

iterations was suggested to be $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$. In conclusion, including Grover iterations, the Grover's key search cost for SEED is approximately Table 2 $\times 2 \times \lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$, as shown in Table 3.

Table 3: Cost of the Grover's key search for SEED

Cipher	Total gates	Total depth	Cost (complexity)	#Qubit	TD-M cost	FD-M cost
SEED	$1.559 \cdot 2^{84}$	$1.657 \cdot 2^{79}$	$1.291 \cdot 2^{164}$	41,497	$1.246 \cdot 2^{88}$	$1.049 \cdot 2^{95}$

6 Conclusion

We can assess the post-quantum security of SEED based on the cost of Grover's key search obtained earlier (in Section 5). In 2016, NIST defined post-quantum security levels by considering the estimated costs of Grover's key search attacks on AES-128, 192, and 256. Nevertheless, with the declining costs of AES attacks, NIST revised the cost assessments to align with the respective security levels in 2019.

According to Table 3, the Grover's key search attack cost for SEED is calculated to be $1.291 \cdot 2^{164}$. This leads to the assessment that SEED attains post-quantum security Level 1.

In summary, this paper presents the first implementation of a quantum circuit for SEED. We focus on optimizing Toffoli and full depths utilizing parallelization and optimized multiplication, squaring and an adder. By analyzing the cost of Grover's key search attack, we confirm that SEED achieves post-quantum security Level 1. Furthermore, we provide $TD \times M$ and $FD \times M$ costs to consider the trade-off between depth and qubits.

References

1. P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994. 1
2. L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996* (G. L. Miller, ed.), pp. 212–219, ACM, 1996. 1
3. NIST., "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process," 2016. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. 2, 7

4. NIST., "Call for additional digital signature schemes for the post-quantum cryptography standardization process," 2022. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf. 2, 7
5. M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: quantum resource estimates," 2015. 2, 6, 7
6. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc." Cryptology ePrint Archive, Report 2019/1146, 2019. https://eprint.iacr.org/2019/1146. 2
7. K. Jang, A. Baksi, G. Song, H. Kim, H. Seo, and A. Chattopadhyay, "Quantum analysis of aes," *Cryptology ePrint Archive*, 2022. 2, 6
8. K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, and H. Seo, "Efficient implementation of present and gift on quantum computers," *Applied Sciences*, vol. 11, no. 11, p. 4776, 2021. 2
9. K. Jang, A. Baksi, H. Kim, H. Seo, and A. Chattopadhyay, "Improved quantum analysis of speck and lowmc (full version)," *Cryptology ePrint Archive*, 2022. 2
10. R. Anand, A. Maitra, and S. Mukhopadhyay, "Evaluation of quantum cryptanalysis on speck," in *Progress in Cryptology – INDOCRYPT 2020* (K. Bhargavan, E. Oswald, and M. Prabhakaran, eds.), (Cham), pp. 395–413, Springer International Publishing, 2020. 2
11. Y. Yang, K. Jang, H. Kim, G. Song, and H. Seo, "Grover on sparkle," in *International Conference on Information Security Applications*, pp. 44–59, Springer, 2022. 2
12. A. Jagielski and K. Kanciak, "Quantum resource estimation for a nist lwc call finalist," *Quantum Information and Computation*, vol. 22, no. 13&14, pp. 1132–1143, 2022. 2
13. S. Roy, A. Baksi, and A. Chattopadhyay, "Quantum implementation of ascon linear layer," *Cryptology ePrint Archive*, 2023. 2
14. Y. Oh, K. Jang, A. Baksi, and H. Seo, "Depth-optimized implementation of ascon quantum circuit," *Cryptology ePrint Archive*, 2023. 2
15. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc." Cryptology ePrint Archive, Report 2019/1146, 2019. https://eprint.iacr.org/2019/1146. 6
16. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing grover oracles for quantum key search on aes and lowmc," in *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*, pp. 280–310, Springer, 2020. 7
17. T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in gf (2m) using normal bases," *Information and computation*, vol. 78, no. 3, pp. 171–177, 1988. 8
18. K. Jang, W. Kim, S. Lim, Y. Kang, Y. Yang, and H. Seo, "Optimized implementation of quantum binary field multiplication with toffoli depth one," in *International Conference on Information Security Applications*, pp. 251–264, Springer, 2022. 9
19. D. Cheung, D. Maslov, J. Mathew, and D. K. Pradhan, "On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography," in *Workshop on Quantum Computation, Communication, and Cryptography*, pp. 96–104, Springer, 2008. 9
20. S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A new quantum ripple-carry addition circuit," *arXiv preprint quant-ph/0410184*, 2004. 10
21. M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte der Physik*, vol. 46, p. 493–505, Jun 1998. 12