Parallel Quantum Addition for Korean Block Cipher

Kyungbae Jang[1], Gyeongju Song[1], Hyunjun Kim[1], Hyeokdong Kwon[1], Hyunji Kim[1], and Hwajeong Seo[1][0000−0003−0069−9061]

[1]IT Department, Hansung University, Seoul (02876), South Korea,
{starj1023, thdrudwn98, khj930704, korlethean, khj1594012, hwajeong84}@gmail.com

Abstract. Adversaries using quantum computers can employ new attacks on cryptography that are not possible with classical computers. Grover's search algorithm, a well-known quantum algorithm, can reduce the search complexity of $O(2^n)$ to $\sqrt{2^n}$ for symmetric key cryptography using an n-bit key. To apply the Grover search algorithm, the target encryption process must be implemented as a quantum circuit. In this paper, we present optimized quantum circuits for Korean block ciphers based on ARX architectures. We adopt the optimal quantum adder and design in parallel way with only a few trade-offs between quantum resources. As a result, we provide a performance improvement of 78% in LEA, 85% in HIGHT, and 70% in CHAM in terms of circuit depth, respectively. Finally, we estimate the cost of the Grover key search for Korean block ciphers and evaluate the post-quantum security based on the criteria presented by NIST.

Keywords: Parallel Quantum Addition · Korean Block Cipher · ARX Architecture · Grover Algorithm.

1 Introduction

International companies, such as IBM, Google, Microsoft, and Amazon are advancing the development of large-scale quantum computers. If a large-scale quantum computer that can operate quantum algorithms is developed, the safety of cryptography, which is widely used today, will be lowered or broken by these upcoming attacks. Shor's algorithm has been proven to break the safety of RSA and Elliptic Curve Cryptography (ECC) [1]. The security of RSA and ECC depends on advances in quantum computers and optimization of the Shor algorithm [2–6]. The National Institute of Standards and Technology (NIST) is currently conducting a Post-Quantum Cryptography (PQC) competition in preparation for the security collapse of public key cryptography. Another quantum algorithm, Grover's search algorithm, is well known for reducing the security level of symmetric key cryptography [7]. The Grover search algorithm accelerates brute force attacks, reducing n-bit symmetric key ciphers with n-bit security to $\frac{n}{2}$-bit security. NIST presented the cost of Grover key search for symmetric key

cryptography as a post-quantum security strength. In other words, as the cost of Grover key search increases, the strength of post-quantum security increases. With this motivation, estimating the cost of Grover key search for symmetric key cryptography is an active research field in recent years [8–18].

In this paper, we focus on quantum cryptanalysis of Korean block ciphers. For this, we firstly optimize quantum circuits for the Korean block ciphers LEA, CHAM, and HIGHT, which are based on ARX architectures. In previous works, Jang et al. estimated the cost required to implement Korean block ciphers as quantum circuits [12]. We present improved performance with proposed implementation techniques and perform tight quantum cryptanalysis on Korean block ciphers. We adopt an optimal quantum adder [19] to optimize the ARX architecture. This quantum adder uses one ancilla qubit in proposed implementation but this can reduce Toffoli gates and circuit depth. In addition, we present a technique to implement the parallel addition by allocating only a few more qubits. As a result, we reduce the number of Toffoli gates and provide high performance improvements in terms of circuit depth. In [12], only costs of quantum circuits for Korean block ciphers are estimated. We tightly estimate the costs of Grover key search based on the proposed quantum circuits. Finally, based on the criteria presented by NIST [20], we evaluate the post-quantum security level of Korean block ciphers. We use ProjectQ [21], a quantum programming tool provided by IBM for validation of our implementation results. Our source code will be available as a public domain for the reproduction of proposed methods.

1.1 Research Contributions

- **Optimized quantum circuits for Korean block ciphers of ARX architecture using an optimal quantum adder** We use an improved quantum adder that can be used when the addition unit is more than 4-bit. Also, we adopt an optimized implementation technique that considers the modular addition without input carry and output carry.
- **Exploring parallel addition points in ARX architecture** LEA, HIGHT, and CHAM block ciphers have ranges that allow the parallel addition. We explore the round function and key schedule structure deeply to find the room for the parallel addition. Then, the parallel addition is implemented with some trade-offs depending on the allowed range.
- **Tight quantum cryptanalysis of Korean block ciphers** Costs of Grover key search are estimated based on optimized quantum circuits. Then, we evaluate the post-quantum security level for Korean block ciphers compared to costs presented by NIST as security strength.

1.2 Organization of the paper

The organization of this paper is as follows. Section 2 briefly describes Grover's search algorithm and presents the background of quantum computing. In Section 3, design techniques for the proposed quantum circuits performing parallel addition are presented. In Section 4, quantum cryptanalysis of Korean block ciphers is performed. Finally, Section 5 concludes the paper.

2 Related Work

2.1 Quantum brute force attack using Grover's search algorithm

A brute force attack on symmetric key cryptography involves finding a key that satisfies a specific plaintext-ciphertext pair. For symmetric key cryptography using an n-bit key, $O(2^n)$ searches are required for a brute force attack. The Grover search algorithm is a quantum algorithm that is optimal for brute force attacks on symmetric key cryptography [7]. Compared to a classic computer that requires $O(2^n)$ searches, the Grover search algorithm recovers the key with a high probability in just $\sqrt{2^n}$ searches. The procedure for Grover key search is as follows.

1. Prepare n-qubit key in superposition $|\psi\rangle$ by applying Hadamard gates. All states of qubits have the same amplitude.

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

2. The symmetric key cryptography is implemented as a quantum circuit and placed in oracle. In oracle $f(x)$, the plaintext is encrypted with the key k in the superposition state. As a result, ciphertexts for all key values are generated. The sign of the solution key is changed to a negative by comparing it with the known ciphertext c. Only when $f(x) = 1$ changes the sign to negative and applies to all states.

$$f(x) = \begin{cases} 1 \text{ if } \mathrm{Enc}(k) = c \\ 0 \text{ if } \mathrm{Enc}(k) \neq c \end{cases} \tag{2}$$

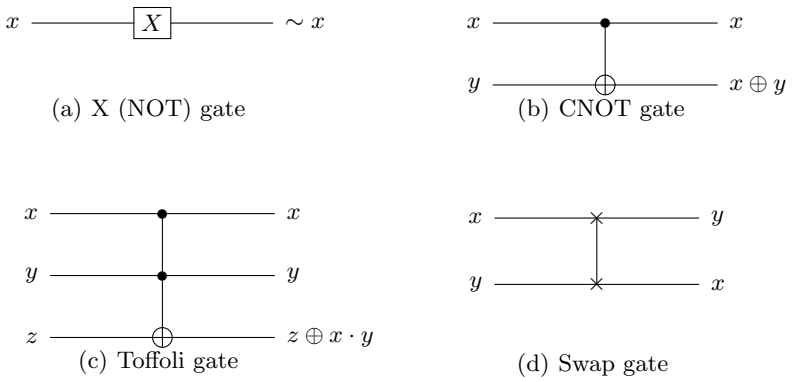
$$U_f(|\psi\rangle |-\rangle) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle |-\rangle \tag{3}$$

3. Lastly, the diffusion operator amplifies the amplitude of the negative sign state.

The Grover algorithm iterates phases 2 and 3 to sufficiently increase the amplitude of the solution and observes it at the end of stage. For an n-bit key, the optimal number of iterations of the Grover search algorithm is $\lfloor \frac{\pi}{4}\sqrt{2^n} \rfloor$, which is about $\sqrt{2^n}$. A brute force attack that requires 2^n searches in a classic computer is reduced to $\sqrt{2^n}$ searches in a quantum computer. The most important thing in this attack is the efficient implementation of symmetric key cryptography as a quantum circuit. Since the diffusion operator is a generic implementation, it does not require any special techniques to implement.

2.2 Quantum gates

Quantum gates used in quantum computers are reversible for all changes during computations. In other words, in quantum computing, it is possible to return to the initial state using only the output state. There are quantum gates with reversible properties that can replace classical gates used in cryptography. Figure 1 shows representative quantum gates frequently used in cryptography implementations.



(a) X (NOT) gate (b) CNOT gate

(c) Toffoli gate (d) Swap gate

Fig. 1: Functionalities and descriptions of quantum gates.

The X (NOT) gate (a) inverts the state of the input qubit. The CNOT gate (b) inverts the state of y only if x is 1. The Toffoli (CCNOT) gate (c) replaces the AND operation, inverting the state of z only if x and y is 1. The Swap gate (d) changes the state of two input qubits to each other.

2.3 Quantum adder

A quantum adder is implemented as a configuration of X, CNOT, and Toffoli gates. In previous implementations of Korean block ciphers [12], a quantum adder based on the ripple-carry approach is used [19]. The quantum adder in previous works utilizes one ancilla qubit, $(2n-2)$ Toffoli gates, $(4n-2)$ CNOT gates, and the circuit depth is $(5n-3)$. In another implementation of the ARX architecture, SPECK [10], Anand. R et al. adopted a different quantum adder [22]. The quantum adder in their works utilizes $(2n-2)$ Toffoli gates, $(5n-6)$ CNOT gates, and the circuit depth is $(5n-5)$, but does not use ancilla qubit. Compared to the quantum adder used in [12], it saves one qubit, but does not improve much in terms of quantum gates or circuit depth.

We use an improved quantum adder based on the ripple-carry approach [19]. This quantum adder uses one ancilla but this can reduce Toffoli gates and circuit depth. When $n \geq 4$ in n-bit addition, an improved quantum adder can be

implemented. Since 8-bit addition of HIGHT block cipher is the smallest unit in Koean block ciphers, it can be applied to all. Also, in the case of modular addition, one ancilla qubit can be saved (generic addition uses two ancilla), and the quantum gates and circuit depth can be reduced. Finally, the quantum adder we adopted uses one anilla qubit, $(2n-3)$ Toffoli gates, $(5n-7)$ CNOT gates, $(2n-6)$ X gates, and the circuit depth is $(2n+3)$. Details of the implementation can be found in [19].

3 Proposed Method

In this section, we present optimized quantum circuit implementation techniques for Korean block ciphers. We use an optimal quantum adder and design a parallel addition structure with only small trade-offs between quantum resources. Proposed implementations provide a 78% improvement in LEA, 85% in HIGHT and 70% in CHAM, compared to previous results in terms of depth, respectively. Furthermore, proposed implementations reduce the use of the Toffoli gate, which is the highest cost in NCT (NOT, CNOT, and Toffoli) gates.

3.1 LEA

LEA is working on the 128-bit plaintext organized in a 32×4 array and uses 128, 192, or 256-bit key. LEA requires constants for generating round keys as follows, and the key schedule of LEA-128/128 is as follows.

$$\begin{array}{ll} \delta_0 = \texttt{0xc3efe9db}, & \delta_1 = \texttt{0x44626b02}, \\ \delta_2 = \texttt{0x79e27c8a}, & \delta_3 = \texttt{0x78df30ec}, \\ \delta_4 = \texttt{0x715ea49e}, & \delta_5 = \texttt{0xc785da0a}, \\ \delta_6 = \texttt{0xe04ef22a}, & \delta_7 = \texttt{0xe5c40957} \end{array} \tag{4}$$

$$\begin{array}{l} K[0] = (K[0] \boxplus (\delta_{i \bmod 4} \lll i)) \lll 1 \\ K[1] = (K[1] \boxplus (\delta_{i \bmod 4} \lll (i+1))) \lll 3 \\ K[2] = (K[2] \boxplus (\delta_{i \bmod 4} \lll (i+2))) \lll 6 \\ K[3] = (K[3] \boxplus (\delta_{i \bmod 4} \lll (i+3))) \lll 11 \\ RK_i = (K[0], K[1], K[2], K[1], K[3], K[1]) \end{array} \tag{5}$$

In the previous implementation [12], $\delta_{0\sim3}$ are initially allocated for the key schedule (LEA-128), and $\delta_{i \bmod 4}$ is used to generate the i-th round key. δ_0 is used to generate the first round key. δ_0 is added to the initial key $K(K[0] \sim [3])$ and the result is used as a round key. First, the result of addition (i.e. $\delta_0 + K[0]$) is updated in $K[0]$ qubits, while δ_0 is kept in unchanged states. δ_0 is sequentially added to $K[1], K[2]$, and $K[3]$. On the other hand, we take a different approach by initially setting four δ_0s.

First, we set four δ_0 (not $\delta_0, \delta_1, \delta_2, \delta_3$). In the key schedule of LEA-128, $K[0], K[1]$, $K[2]$, and $K[3]$ are independent of each other. They can be operated in a parallel. For the parallel addition, four same values of $\delta_0[0] \sim [3]$ are

required, and $\delta_0[j]$ is added to $K[j]$ $(j = 0, 1, 2, 3)$. As in the previous implementation, if one δ_0 is used in an addition, it cannot be added in a parallel. δ_0 is returned when the addition is completed. However, in the process of addition with $K[0] \sim [3]$, the value of δ_0 is changed by each $K[0] \sim [3]$. It is a design feature that a ripple-carry quantum adder for the addition $(x + y)$, the result is stored in y, x is changed to x' during the process, and returned to x when the addition is complete [19].

Second, we use four carry qubits (c_0, c_1, c_2, c_3). In the ripple-carry modular adder, one carry qubit c_0 is used, which is initialized to 0 after the addition. Taking the advantage of saving qubits, the previous implementation reuses one carry qubit c_0 for all additions. However, reusing initialized c_0 makes the parallel addition impossible. For the parallel addition with K_0, K_1, K_2, and K_3, we additionally allocate 3 qubits (c_1, c_2, and c_3). As a result, we provide a high performance improvement in terms of depth.

Lastly, we change four δ_0s to four δ_1s for the next key schedule. Since `0xc3efe9db` (δ_0) and `0x44626b02` (δ_1) are known values in advance, only X gates are used to change values of qubits. The first round key generation in the proposed quantum circuit design for the key schedule of LEA-128 is described in Algorithm 1. The detailed process for changing δ is described in Algorithm 2.

Algorithm 1 Quantum circuit for LEA-128 key schedule (first round key generation).

Input: Initial key $K[0] \sim [3], \delta_0[0] \sim [3], c_{0\sim3}$
Output: Round key $RK_0, \delta_1[0] \sim [3]$
1: $\delta_0[0] \leftarrow \delta_0[0] \lll 0$
2: $K[0] \leftarrow ADD(\delta_0[0], K[0], c_0) \lll 1$
3: $\delta_0[1] \leftarrow \delta_0[1] \lll 1$
4: $K[1] \leftarrow ADD(\delta_0[1], K[1], c_1) \lll 3$
5: $\delta_0[2] \leftarrow \delta_0[2] \lll 2$
6: $K[2] \leftarrow ADD(\delta_0[2], K[2], c_2) \lll 6$
7: $\delta_0[3] \leftarrow \delta_0[3] \lll 3$
8: $K[3] \leftarrow ADD(\delta_0[3], K[3], c_3) \lll 11$

// Reverse
9: $\delta_0[0] \leftarrow \delta_0[0] \ggg 0$
10: $\delta_0[1] \leftarrow \delta_0[1] \ggg 1$
11: $\delta_0[2] \leftarrow \delta_0[2] \ggg 2$
12: $\delta_0[3] \leftarrow \delta_0[3] \ggg 3$
13: $\delta_1[0] \sim [3] \leftarrow$ Change $\delta(\delta_0[0] \sim [3],$ `0xc3efe9db`, `0x44626b02`$)$
14: **return** $RK_0(K_0, K_1, K_2, K_1, K_3, K_1), \delta_1[0] \sim [3]$

LEA-192 uses $\delta_{0\sim5}$ and LEA-256 uses $\delta_{0\sim7}$, respectively. The key schedule structure of LEA-192 and LEA-256 is the same as that of LEA-128, and LEA-256 can save qubits compared to the previous implementation [12]. The key schedule of LEA-256 uses $\delta_{0\sim7}$. However, five δ variables are used. We allocate only qubits

Algorithm 2 Change δ.

Input: $\delta[0] \sim [3]$, current δ, next δ
Output: $\delta[0] \sim [3]$
current $\delta \leftarrow$ current $\delta \oplus$ next δ
for $i = 0$ to 31 **do**
  if (current $\delta \gg i$) & 1 **then**
    $\delta[0][i] \leftarrow$X($\delta[0][i]$)
    $\delta[1][i] \leftarrow$X($\delta[1][i]$)
    $\delta[2][i] \leftarrow$X($\delta[2][i]$)
    $\delta[3][i] \leftarrow$X($\delta[3][i]$)
  end if
end for
return $\delta[0], \delta[1], \delta[2], \delta[3]$

for $\delta[0] \sim [5]$. In [12], qubits for $\delta_{0\sim7}$ are allocated. On the other hand, we save 64 qubits by allocating only qubits for $\delta[0] \sim [5]$. The first round key generation of the proposed quantum circuit design for LEA-192 and LEA-256 is same and is described in Algorithm 3.

Algorithm 3 Quantum circuit for key schedule of LEA-192 and LEA-256 (first round key generation).

Input: Initial key $K[0] \sim [5], \delta_0[0] \sim [5], c_{0\sim5}$
Output: Round key $RK_0, \delta_1[0] \sim [5]$
$\delta_0[0] \leftarrow \delta_0[0] \lll 0$
$K[0] \leftarrow ADD(\delta_0[0], K[0], c_0) \lll 1$
$\delta_0[1] \leftarrow \delta_0[1] \lll 1$
$K[1] \leftarrow ADD(\delta_0[1], K[1], c_1) \lll 3$
$\delta_0[2] \leftarrow \delta_0[2] \lll 2$
$K[2] \leftarrow ADD(\delta_0[2], K[2], c_2) \lll 6$
$\delta_0[3] \leftarrow \delta_0[3] \lll 3$
$K[3] \leftarrow ADD(\delta_0[3], K[3], c_3) \lll 11$
$\delta_0[4] \leftarrow \delta_0[4] \lll 4$
$K[4] \leftarrow ADD(\delta_0[4], K[4], c_4) \lll 13$
$\delta_0[5] \leftarrow \delta_0[5] \lll 5$
$K[5] \leftarrow ADD(\delta_0[5], K[5], c_5) \lll 17$

// Reverse
$\delta_0[0] \leftarrow \delta_0[0] \ggg 0$
$\delta_0[1] \leftarrow \delta_0[1] \ggg 1$
$\delta_0[2] \leftarrow \delta_0[2] \ggg 2$
$\delta_0[3] \leftarrow \delta_0[3] \ggg 3$
$\delta_0[4] \leftarrow \delta_0[4] \ggg 4$
$\delta_0[5] \leftarrow \delta_0[5] \ggg 5$
$\delta_1[0] \sim [5] \leftarrow$ Change $\delta(\delta_0[0] \sim [5], \texttt{0xc3efe9db}, \texttt{0x44626b02})$
return $RK_0(K_0, K_1, K_2, K_3, K_4, K_5), \delta_1[0] \sim [5]$

The round function of LEA is performed on 128-qubit plaintext ($X[0], X[1], X[2], X[3]$) and there is no point of parallelism because the X variables are not independent each other. Algorithm 4 describes the round function of LEA block cipher. Additions that update $X[3]$ and $X[2]$ cannot be done in a parallel way because we use $X[2]$ to update $X[3]$. CNOT32 means CNOT gate operation in 32-qubit units.

Algorithm 4 Quantum circuit for round function of LEA.

Input: $X[0] \sim [3], RK[0] \sim [5], c_0$
Output: $X(X[0] \sim [3])$

//Update $X[3]$
1: $X[3] \leftarrow \text{CNOT32}(RK[5], X[3])$
2: $X[2] \leftarrow \text{CNOT32}(RK[4], X[2])$
3: $X[3] \leftarrow ADD(X[2], X[3], c_0) \ggg 3$
4: $X[2] \leftarrow \text{CNOT32}(RK[4], X[2])$ //Reverse

//Update $X[2]$
5: $X[2] \leftarrow \text{CNOT32}(RK[3], X[2])$
6: $X[1] \leftarrow \text{CNOT32}(RK[2], X[1])$
7: $X[2] \leftarrow ADD(X[1], X[2], c_0) \ggg 5$
8: $X[1] \leftarrow \text{CNOT32}(RK[2], X[1])$ //Reverse

//Update $X[1]$
9: $X[2] \leftarrow \text{CNOT32}(RK[1], X[1])$
10: $X[1] \leftarrow \text{CNOT32}(RK[0], X[0])$
11: $X[2] \leftarrow ADD(X[0], X[1], c_0) \ggg 9$
12: $X[1] \leftarrow \text{CNOT32}(RK[0], X[0])$ //Reverse
13: **return** $X[1], X[2], X[3], X[0]$

3.2 HIGHT

HIGHT is working with 64-bit plaintext organized in a 8×8 array and uses 128-bit key in 8×16 array. In HIGHT block cipher, the key schedule to generate round keys using δ and the round function are as follows.

$$\begin{aligned} s_{i+6} &= s_{i+2} \oplus s_{i-1} \\ \delta_i &= (s_{i+6}, s_{i+5}, s_{i+4}, s_{i+3}, s_{i+2}, s_{i+1}, s_i) \end{aligned} \tag{6}$$

$$\begin{aligned} &\text{for } i = 0 \text{ to } 7: \\ &\quad \text{for } j = 0 \text{ to } 7: \\ &\qquad RK[16 \cdot i + j] = K[j - i \bmod 8] \boxplus \delta_{16 \cdot i + j} \\ &\quad \text{for } j = 0 \text{ to } 7: \\ &\qquad RK[16 \cdot i + j + 8] = K[(j - i \bmod 8) + 8] \boxplus \delta_{16 \cdot i + j + 8} \end{aligned} \tag{7}$$

$$
\begin{aligned}
X_i[j] &= X_{i-1}[j-1], \quad j = 1, 3, 5, 7 \\
X_i[0] &= X_{i-1}[7] \oplus (F_0(X_{i-1}[6]) \boxplus RK[4i-1]) \\
X_i[2] &= X_{i-1}[1] \boxplus (F_1(X_{i-1}[0]) \oplus RK[4i-4]) \\
X_i[4] &= X_{i-1}[3] \oplus (F_0(X_{i-1}[2]) \boxplus RK[4i-3]) \\
X_i[6] &= X_{i-1}[5] \boxplus (F_1(X_{i-1}[4]) \oplus RK[4i-2])
\end{aligned} \tag{8}
$$

In the previous implementation [12], an on-the-fly approach is used to reduce qubits. Initial key qubits are updated and used as round keys. To reduce the use of qubits, δ_0, which is required for the round key generation, is set and updated to the next $\delta_i (1 \leq i \leq 127)$. Since the four additions are independent of each other in the round function, the parallel addition is possible. However, if a round key is generated by updating the initially set δ_0, the following additions cannot be performed in a parallel way. Thus, we initially set $\delta_0, \delta_1, \delta_2, \delta_3$ for the parallel addition. The next round uses $\delta_0, \delta_1, \delta_2, \delta_3 \rightarrow \delta_4, \delta_5, \delta_6, \delta_7$ updated by + 4. Updating δ can be implemented simply with a few CNOT gates and logical swap gate. The proposed round function quantum circuit of HIGHT is described in Algorithm 5, including key schedule. Since the functions F_0 and F_1 are the same as the implementation of [12], F_0 and F_1 implementations are omitted in this paper. Reverse operations performed at the end of Algorithm 5 return values of $X[0], X[2], X[4], X[6]$ and also return values of $K[0] \sim [3]$ for the later key schedule. In HIGHT, the parallel addition can be performed in both round function and key schedule. This is the reason for the highest performance improvement (i.e. 85 %) in HIGHT.

3.3 CHAM

In the circuit depth, LEA and HIGHT provide performance improvements of 78 % and 85 %, and CHAM provides 70 % improvement. In this section, we describe parallel addition in CHAM. We discuss the performance differences between CHAM, LEA, and HIGHT in Section 4.

CHAM is working with $64(16 \times 4)$ or $128(32 \times 4)$-bit plaintext and uses 128 or 256-bit key. Figure 2 shows round functions of CHAM and proposed technique at the same time. We focus on four round functions $(i = 0, 1, 2, 3)$ and describe the parallel point. We perform the parallel addition for three rounds $(i = 0, 1, 2)$. In the blue box (i.e. $i = 3$) in Figure 2, we need the addition result $(X'[0])$ of the round function when $i = 0$ previously. This is why the parallel addition is only possible for three rounds. Now we prepare values for the parallel addition.

First, we allocate three additional qubits (c_0, c_1, and c_2) for he parallel addition. In the red box, RK is XORed to X. Since X is required in the next addition, we XOR X to RK using CNOT gates. The parallel addition is possible without changing the value of X. The rotation operation on X reverts back after updating RK. Lastly, we need three round keys. In the previous implementation [12], t additional qubits are allocated for round key generation ($t = 3$ for 64-bit plaintext, $t = 11$ for 128-bit plaintext). We allocate $(3 \times t)$ qubits to generate three round keys. In this way, the parallel addition is performed in

Algorithm 5 Quantum circuit for round function of HIGHT (first round).

Input: $X[0] \sim [7], K[0] \sim [3], \delta_{0\sim3}, c_{0\sim3}$
Output: $X[0] \sim [7], \delta_{4\sim7}$
Generate $RK[0]$ **and transform** $X[0]$**:**
    $RK[0] \leftarrow ADD(\delta_0, K[0], c_0)$
    $X[0] \leftarrow F_1(X[0])$
    $X[0] \leftarrow \text{CNOT8}(RK[0], X[0])$
$X[1] \leftarrow ADD(X[0], X[1], c_0)$

Generate $RK[1]$ **and transform** $X[2]$**:**
    $RK[1] \leftarrow ADD(\delta_1, K[1], c_1)$
    $X[2] \leftarrow F_0(X[2])$
    $X[2] \leftarrow ADD(RK[1], X[2], c1)$
$X[3] \leftarrow \text{CNOT8}(X[2], X[3])$

Generate $RK[2]$ **and transform** $X[4]$**:**
    $RK[2] \leftarrow ADD(\delta_2, K[2], c_2)$
    $X[4] \leftarrow F_1(X[4])$
    $X[4] \leftarrow \text{CNOT8}(RK[2], X[4])$
$X[5] \leftarrow ADD(X[4], X[5], c_2)$

Generate $RK[3]$ **and transform** $X[6]$**:**
    $RK[3] \leftarrow ADD(\delta_3, K[3], c_3)$
    $X[6] \leftarrow F_0(X[6])$
    $X[6] \leftarrow ADD(RK[3], X[6], c_3)$
$X[7] \leftarrow \text{CNOT8}(X[6], X[7])$

Reverse(**generate** $RK[0]$ **and transform** $X[0]$**)**
Reverse(**generate** $RK[1]$ **and transform** $X[2]$**)**
Reverse(**generate** $RK[2]$ **and transform** $X[4]$**)**
Reverse(**generate** $RK[3]$ **and transform** $X[6]$**)**
$\delta_{4\sim7} \leftarrow$ Update $\delta(\delta_{0\sim3})$
return $X[7], X[0], X[1], X[2], X[3], X[4], X[5], X[6], \delta_{4\sim7}$

three units. Algorithm 6 describes a quantum circuit for a round function when $i =$ odd, because only the number of rotations changes depending on whether i is odd or even. For the round key generation, we perform it in three units. Since the round key generation itself is the same as [12], it is omitted in this paper.

4 Evaluation

In this section, we discuss performance improvements of Korean block ciphers on quantum circuits. Then, we estimate the costs for Grover's key search based on the proposed quantum circuits for Korean block ciphers. Finally, we evaluate the post-quantum security strength. We apply post-quantum security strength estimation for symmetric key cryptography presented by NIST [20].

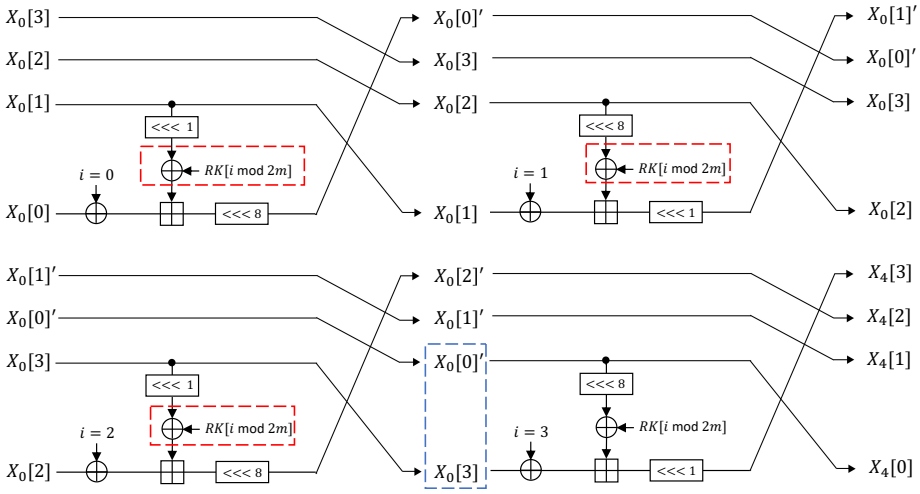


Fig. 2: Four round functions of CHAM ($i = 0, 1, 2, 3$).

4.1 Comparison of quantum circuit implementations for Korean block ciphers

Table 1 shows quantum resources required for the Korean block ciphers implemented in [12]. Since the circuit depth was not specified, we estimated it ourselves. Table 2 shows the quantum resources required for the implementations presented in this paper. The design of quantum circuits in [12] is overly focused on saving qubits and does not take into account the parallelism of additions. On the other hand, by adopting the optimal quantum adder and performing parallel addition, optimized implementation gives 78 % improvement in LEA, 85 % in HIGHT and 70 % in CHAM compared to previous implementations in terms of depth, respectively.

For LEA-128/128, four additions are performed in a parallel way in the key schedule, and for LEA-128/192 and LEA-128/256, six additions are performed in a parallel way. In CHAM, three additions are performed in a parallel way in the round function. Therefore, the performance improvement is higher in the case of LEA than that of CHAM. HIGHT provides the highest performance improvement, as the four additions are performed in parallel in both the round function and key schedule. Also, the number of Toffoli gates, which are high-cost quantum gates, is reduced due to improvements in the quantum adder itself.

4.2 Cost estimation for Grover oracle

We estimate the cost of the oracle of Grover's search algorithm. The proposed quantum circuit is located in the oracle, and the plaintext is encrypted with the key in the superposition state. The generated ciphertext is compared to the known ciphertext and reverse operations are performed. In [11], Grassl et

Algorithm 6 Quantum circuit for round function of CHAM-64/128 (i is odd).

Input: $X[0] \sim [3], RK, c$
Output: $X[0] \sim [3]$
//Add round constant(i)
for $j = 0$ to 7 **do**
  if $(i \gg j)$ & 1 **then**
    $X[0] \leftarrow$X($X[0][j]$)
  end if
end for

//Preparation for parallel addition
$X[1] \leftarrow X[1] \lll 8$
$RK \leftarrow$ CNOT16($X[1], RK$)
$X[1] \leftarrow X[1] \ggg 8$

//Parallel addition part
$X[0] \leftarrow ADD(RK, X[0], c)$

//Reverse
$X[1] \leftarrow X[1] \lll 8$
$RK \leftarrow$ CNOT16($X[1], RK$)
$X[1] \leftarrow X[1] \ggg 8$
return $X[1], X[2], X[3], X[0] \lll 1$

al. estimated the cost of the Grover key search for AES and suggested that r plaintext-ciphertext pairs are required to recover a unique key ($r = 3, 4, 5$ for AES-128, 192, 256). Later, Langenberg et al. suggested that $r = \lceil k/n \rceil$ (i.e. key size/block size) is sufficient to recover a unique key [13]. Based on the approach in [13], we assume that $r = \lceil k/n \rceil$ and estimate the cost of oracle. Encryptions(r) can be performed in parallel.

Finally, for oracle, ($2 \times r \times$Table 2) quantum gates and ($r \times$Table 2+1) qubits are used and the depth is ($2 \times$ Table $2 + 1$). For the analysis at the Clifford + T level, we decompose the Toffoli gate into seven T gates + eight Clifford gates following the decomposition in [23]. X gates and CNOT gates are counted as Clifford gates. For the n-bit ciphertext, an $n \cdot r$ multi-controlled NOT gate is used to check whether it matches the known ciphertext. It is decomposed into $(32 \cdot n \cdot r - 84)$ T gates [24]. Multi-controlled NOT gates also use one additional qubit, which is flipped if the ciphertext matches. Table 3 shows the quantum resources required for Grover oracle. In oracle, when $r \geq 2$, plaintexts(r) are encrypted with the same key. So there is room for optimization of qubits and quantum gates. However, for simplicity of estimation and following Grassl's approach [11], this optimization is not taken into account.

Table 1: Quantum resources required for previous implementations [12].

Cipher		Qubits	Toffoli gates	CNOT gates	X gates	Depth (Extrapolation)
LEA	128/128	385	10,416	28,080	68	26,328
	128/192	513	15,624	39,816	100	39,452
	128/256	641	17,856	45,504	130	45,057
HIGHT	64/128	201	6,272	20,523	4	16,447
CHAM	64/128	196	2,400	12,285	240	7,807
	128/128	268	4,960	26,885	240	19,880
	128/256	396	5,952	32,277	304	23,856

Table 2: Quantum resources required for proposed implementations.

Cipher		Qubits	Toffoli gates	CNOT gates	X gates	Depth
LEA (this work)	128/128	388	10,248	32,616	11,152	6,505
	128/192	518	15,372	46,620	17,004	7,589
	128/256	582	17,568	53,280	19,494	8,580
HIGHT (this work)	64/128	228	5,824	22,614	4,496	2,479
CHAM (this work)	64/128	204	2,320	13,200	2,320	2,615
	128/128	292	4,880	28,760	4,880	5,307
	128/256	420	5,856	34,944	5,872	6,594

Table 3: Quantum resources required for Grover oracle.

Cipher		r	Qubits	T gates	Clifford gates	Total gates	Depth
LEA	128/128	1	389	147,484	251,504	398,988	13,011
	128/192	2	1,037	438,524	746,400	1,184,924	15,179
	128/256	2	1,165	500,012	853,272	1,353,284	17,161
HIGHT	64/128	2	457	167,084	294,808	461,892	4,959
CHAM	64/128	2	409	68,972	136,320	205,292	5,231
	128/128	1	293	72,332	145,360	217,692	10,615
	128/256	2	841	172,076	350,656	522,732	13,189

4.3 Cost estimation for Grover key search

The Grover search algorithm is well known for reducing complexity $O(2^k)$ to $\sqrt{2^k}$. Later, M. Boyer et al. provided a tight analysis of Grover's search algorithm and suggested $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ instead of $\sqrt{2^k}$ as the optimal number of iterations [25]. Thus, we estimate Table 3 $\times \lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ excluding qubits as the cost of Grover key search, and is shown in Table 4. The cost of Grover's key search depends on the complexity of oracle. The diffusion operator is also included in the Grover iteration, but it is mostly excluded from the cost estimation [10, 11, 13].

Table 4: Cost estimation for Grover key search.

Cipher		Qubits	Total gates	Total depth	Cost	NIST security
LEA	128/128	389	$1.195 \cdot 2^{82}$	$1.247 \cdot 2^{77}$	$1.491 \cdot 2^{159}$	Not achieved
	128/192	1,037	$1.775 \cdot 2^{115}$	$1.455 \cdot 2^{109}$	$1.292 \cdot 2^{225}$	Level 1
	128/256	1,165	$1.014 \cdot 2^{148}$	$1.645 \cdot 2^{141}$	$1.668 \cdot 2^{289}$	Level 3
HIGHT	64/128	457	$1.384 \cdot 2^{82}$	$1.901 \cdot 2^{75}$	$1.316 \cdot 2^{158}$	Not achieved
CHAM	64/128	409	$1.23 \cdot 2^{81}$	$1.003 \cdot 2^{76}$	$1.234 \cdot 2^{157}$	Not achieved
	128/128	293	$1.304 \cdot 2^{81}$	$1.018 \cdot 2^{77}$	$1.328 \cdot 2^{158}$	Not achieved
	128/256	841	$1.566 \cdot 2^{146}$	$1.264 \cdot 2^{141}$	$1.98 \cdot 2^{287}$	Level 3

Level 1 : 2^{170}, Level 3 : 2^{233}, Level 5 : 2^{298}

NIST presents the following post-quantum security strengths [20] based on the cost of Grover key search for AES estimated by Grassl et al [11].

- **Level 1**: Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g. AES-128)
- **Level 3**: Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 192-bit key (e.g. AES-192)
- **Level 5**: Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 256-bit key (e.g. AES-256)

NIST estimates the cost for AES-128, 196 and 256 as 2^{170}, 2^{233} and 2^{298} which are (total gates $\times$ total depth) in Grassl et al's implementations [11]. Now, we compare the cost of Grover key search for Korean block ciphers with the post-quantum security strength presented by NIST. The cost of key search for LEA-128/128, HIGHT, CHAM-64/128, CAHM-128/128 using a 128-bit key is less than the cost of AES-128, which is Level 1 (2^{170}). Therefore, no security level is achieved. In the post-quantum era, increasing the key size for symmetric key cryptography is a well-known countermeasure. LEA-128/192 using a 192-bit key requires $\lfloor \frac{\pi}{4}\sqrt{2^{192}} \rfloor$ searches, which increases the cost of key search. However, compared to AES-192(2^{233}), which has the same key size, LEA-128/192 is exposed to attack at a lower cost and achieves only Level 1. In the case of LEA-128/256 and CHAM-128/256 using a 256-bit key, the security level also increases according to the key size, but only achieves Level 3 because it is exposed to attack at a lower cost than AES-256(2^{298}).

5 Conclusion

In this paper, we presented optimized quantum circuit implementations for Korean block ciphers. We improved the performance of the quantum adder itself and implemented the parallel addition by exploring the structure of Korean block ciphers which are ARX architectures. Finally, we estimated costs of Grover search and evaluated the security level of Korean block ciphers based on the post-quantum security strength presented by NIST. Future work is to apply Grover search algorithm to cryptanalysis rather than exhaustive key search. One such prominent candidate would be differential or linear cryptanalysis.

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.
2. T. Häner, M. Roetteler, and K. M. Svore, "Factoring using $2n + 2$ qubits with toffoli based modular multiplication," 2017.
3. C. Gidney, "Factoring with $n + 2$ clean qubits and n-1 dirty qubits," 2018.
4. M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," 2017.
5. T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, "Improved quantum circuits for elliptic curve discrete logarithms," 2020.
6. G. Banegas, D. J. Bernstein, I. van Hoof, and T. Lange, "Concrete quantum cryptanalysis of binary elliptic curves," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, p. 451–472, Dec. 2020.
7. L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.
8. R. Anand, A. Maitra, and S. Mukhopadhyay, "Grover on SIMON," *Quantum Information Processing*, vol. 19, p. 340, 09 2020.
9. K. Jang, S. Choi, H. Kwon, and H. Seo, "Grover on SPECK: Quantum resource estimates." Cryptology ePrint Archive, Report 2020/640, 2020. `https://ia.cr/2020/640`.
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.
11. M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: quantum resource estimates," 2015.
12. K. Jang, S. Choi, H. Kwon, H. Kim, J. Park, and H. Seo, "Grover on Korean block ciphers," *Applied Sciences*, vol. 10, no. 18, 2020.
13. B. Langenberg, H. Pham, and R. Steinwandt, "Reducing the cost of implementing the advanced encryption standard as a quantum circuit," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–12, 2020.
14. M. Almazrooie, A. Samsudin, R. Abdullah, and K. Mutter, "Quantum reversible circuit of AES-128," *Quantum Information Processing*, vol. 17, 03 2018.

15. 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, 2021.
16. A. Chauhan and S. Sanadhya, *Quantum Resource Estimates of Grover's Key Search on ARIA*, pp. 238–258. 12 2020.
17. S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing Grover oracles for quantum key search on AES and LowMC," 2019.
18. R. Anand, S. Maitra, A. Maitra, C. S. Mukherjee, and S. Mukhopadhyay, "Resource estimation of Grovers-kind quantum cryptanalysis against FSR based symmetric ciphers." Cryptology ePrint Archive, Report 2020/1438, 2020. `https://ia.cr/2020/1438`.
19. S. Cuccaro, T. Draper, S. Kutin, and D. Moulton, "A new quantum ripple-carry addition circuit," 11 2004.
20. 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`.
21. D. S. Steiger, T. Häner, and M. Troyer, "ProjectQ: an open source software framework for quantum computing," *Quantum*, vol. 2, p. 49, 2018.
22. Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out," 2009.
23. M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, p. 818–830, Jun 2013.
24. N. Wiebe and M. Roetteler, "Quantum arithmetic and numerical analysis using repeat-until-success circuits," 2014.
25. M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte der Physik*, vol. 46, p. 493–505, Jun 1998.