Depth-Optimized Quantum Implementation of ARIA

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

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

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

Abstract. The advancement of large-scale quantum computers poses a threat to the security of current encryption systems. In particular, symmetric-key cryptography significantly is impacted by general attacks using the Grover's search algorithm. In recent years, studies have been presented to estimate the complexity of Grover's key search for symmetric-key ciphers and assess post-quantum security. In this paper, we propose a depth-optimized quantum circuit implementation for ARIA, which is a symmetric key cipher included as a validation target the Korean Cryptographic Module Validation Program (KCMVP). Our quantum circuit implementation for ARIA improves the depth by more than 88.2% and Toffoli depth by more than 98.7% compared to the implementation presented in Chauhan et al.'s SPACE'20 paper. Finally, we present the cost of Grover's key search for our circuit and evaluate the post-quantum security strength of ARIA according to relevant evaluation criteria provided NIST.

Keywords: Quantum Computer · Grover's Search Algorithm · ARIA · Post-quantum Security

1 Introduction

Quantum computers, built upon principles of quantum mechanics like quantum superposition and entanglement, have the capability to solve specific problems at a faster rate compared to classical computers. As a result, many companies and research institutions are concentrating on quantum computer development. However, it is known that the advancement of large-scale quantum computers has the potential to pose a threat to the security of current cryptographic systems. In particular, symmetric-key cryptography can be significantly compromised by general attacks using the Grover's search algorithm, which can reduce the data search complexity. As a result, in recent years, studies have been presented to estimate the complexity of recovering secret keys in existing symmetric-key ciphers using the Grover's search algorithm and evaluate post-quantum security based on these findings [6,13,12,8,23,9,20].

ARIA is a symmetric-key cryptography algorithm optimized for ultra-light environments and hardware implementation, and is included as a validation target in the Korean Cryptographic Module Validation Program (KCMVP). This means that ARIA is widely used in verified cryptographic modules, so it is very important to measure ARIA's quantum security strength for future preparedness against emerging threats. Fortunately, there is already a study that measured the quantum security strength of ARIA in 2020 [2]. However, since [2] primarily focuses on qubit optimization, there is also a need for research that addresses the recent emphasis on optimizing depth.

In a document guiding evaluation criteria for post-quantum cryptography standardization, NIST provided a criteria for estimating quantum attack complexity and proposed a parameter called MAXDEPTH, which refers to the maximum circuit depth that a quantum computer can execute. In order to evaluate the strength of quantum security, not only the quantum attack complexity but also the MAXDEPTH related to execution must be considered. Further elaboration on this topic can be found in Sections 2.4, and 4.

The paper is structured as follows. Section 2 offers the background for this paper. Section 2.1 provides an introduction to ARIA. In Section 2.2, the quantum gates utilized to implement quantum circuits are covered. In Section 2.3 Grover's key search is examined because it relates to measuring quantum resources, and in Section 2.4, NIST post-quantum security and MAXDEPTH are covered because they are crucial for estimating security strength. Following this, in Sections **??**, the design of quantum circuits for ARIA is suggested, drawing upon the information presented in Section 2. Section 4.2 presents the cost of Grover's key search for our circuit and evaluates ARIA's post-quantum security strength based on the estimates. Lastly, Section 5 delves into the summarizing conclusions and outlines potential directions for future research.

1.1 Our Contribution

This paper makes the following contributions:

1. **Low depth quantum implementation of ARIA**. In our implementation of the ARIA quantum circuit, our main focus is minimizing the Toffoli depth and ensuring full depth. We achieve a reduction in Toffoli depth and full depth through various techniques for optimization.
2. **Various techniques for optimization**. We utilize various techniques for optimization to reduce the depth. For optimizing binary field operations, we choose a multiplication optimizer that implements the Karatsuba algorithm in parallel and a squaring method using PLU factorization. Furthermore, we enhance implementing parallel processing for applicable components.
3. **Evaluation of post-quantum security**. We estimate the resources required for implementing quantum circuits for ARIA. The resource estimation for the ARIA quantum circuit also includes the comparison with previous research. Furthermore, we evaluate the quantum security of ARIA by estimating the cost of Grover's key search based on the implemented quantum circuit and comparing them with the security levels provided by NIST.

2 Background

2.1 ARIA Block Cipher

ARIA [15], which stands for Academy, Research Institute, and Agency, is a Korean symmetric key block cipher jointly developed by the three organizations mentioned above. Since the adoption of ARIA as a national standard encryption algorithm in 2004, it has been widely used for secure communication and data protection. Especially, ARIA holds significance as symmetric key ciphers included in the validation subjects of the KCMVP. ARIA has an interface similar to AES, a symmetric key block cipher standard, because its designers considered the design principles of AES during its development. It has an Involutional Substitution-Permutation Network (ISPN) structure optimized for lightweight hardware implementation. The input/output size of ARIA is fixed at 128-bit, and only the key size is different as 128, 192, and 256-bit.

Round Function The round function is made of three main operations: *AddRoundKey*, *Substitution layer*, and *Diffusion layer*.

In the *AddRoundKey*, the round key suitable for each round is XORed to intermediate state.

In the *Substitution layer*, the input 128-bit state is divided into 8-bit units, and substitutions are performed using the S-boxes. ARIA employs a total of four S-boxes $(S_1, S_1^{-1}, S_2, S_2^{-1})$, which include the inverse S-boxes. The S_1, S_1^{-1} are identical to the ones used in AES, and the S_2, S_2^{-1} are newly designed S-boxes specifically for ARIA. These S-boxes used in ARIA are generated by applying an affine transformation to the functions x^{-1} and x^{247} over $GF(2^8)$. The S-boxes $S_1(x), S_2(x)$ are obtained by performing multiplication between 8×8 non-singular matrix ($\mathbf{A}$ or $\mathbf{B}$) and the function(x^{-1} or x^{247}), followed by XOR with 8×1 vector. This can be expressed as follows:

$$
\begin{aligned}
S_1(x) &= \mathbf{A} \cdot x^{-1} \oplus [1,1,0,0,0,1,1,0]^{\mathrm{T}}, \\
S_2(x) &= \mathbf{B} \cdot x^{247} \oplus [0,1,0,0,0,1,1,1]^{\mathrm{T}}
\end{aligned}
$$

$$
\text{where} \quad \mathbf{A} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0&1&0&1&1&1&1&0 \\ 0&0&1&1&1&1&0&1 \\ 1&1&0&1&0&1&1&1 \\ 1&0&0&1&1&1&0&1 \\ 0&0&1&0&1&1&0&0 \\ 1&0&0&0&0&0&0&1 \\ 0&1&0&1&1&1&0&1 \\ 1&1&0&1&0&0&1&1 \end{pmatrix} \tag{1}
$$

ARIA features two types of S-box layers consisting of four S-boxes. *Type 1* comprises four 32-bit sets consisting of S_1, S_2, S_1^{-1}, and S_2^{-1} in this order. Since the two types are the inverse relationship to each other,

Type 2 is the inverse of *Type 1* (i.e., Type1^{-1} = Type2). *Type 1* is used for odd rounds and *Type 2* for even rounds in the round function. The two types of S-box layers in ARIA are shown in Figure 1.

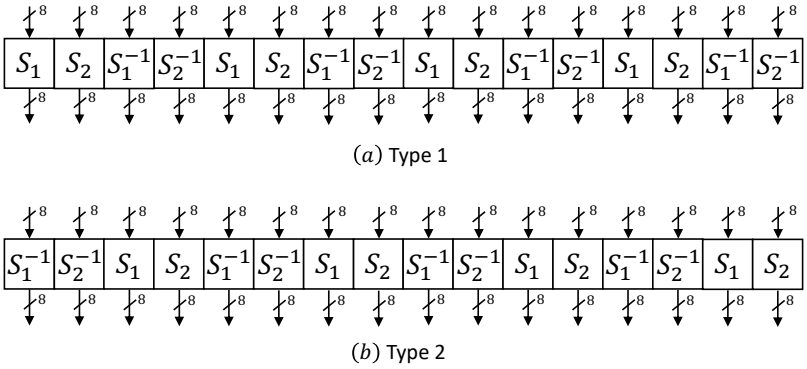

Fig. 1: Two types of S-box layers in ARIA

The Diffusion layer performs byte-wise matrix multiplication by multiplying the given 16 × 16 involution binary matrix with the output of the substitution layer. The involution binary matrix does not require a separate implementation of the inverse matrix during the decryption process, as its inverse matrix is the same as itself.

The detailed composition of the round function differs depending on whether the round is odd, even, or final. The main difference between odd and even rounds lies in the type of the S-box layer used: odd rounds use a *Type 1*, whereas even rounds use a *Type 2*. In the final round, the diffusion step is omitted and the AddRoundKey is performed once more. A brief outline of the round function of ARIA is shown in Figure 2.

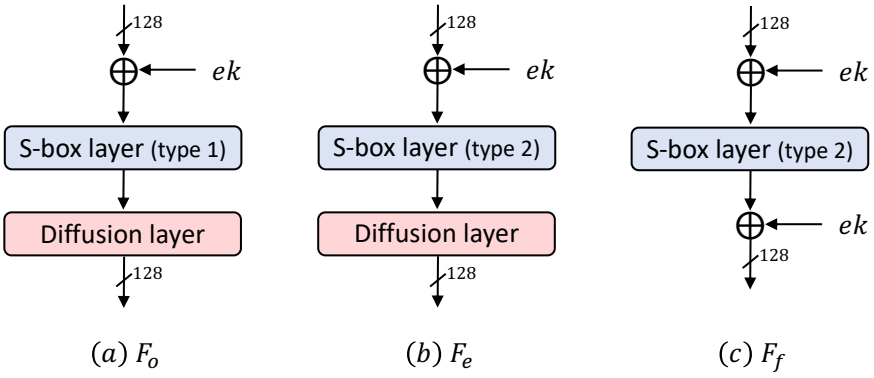

Fig. 2: Brief outline of round function of ARIA.

Key Schedule In the *key initialization step* (Figure 3), 128-bit initial constants W_0, W_1, W_2, and W_3 are generated as essential components for generating a round key . During this step, the round functions F_o and F_e are utilized.

$$KL||KR = MK||0\cdots0. \tag{2}$$

Equation 2 represents the formula used to generate the input values KL and KR in the key initialization step. This equation is derived from the master key MK. Since the concatenated result of KL and KR, which are each 128-bit, is fixed to 256-bit (i.e., $KL||KR$), if MK is smaller than 256, padding is performed to match the size by filling the insufficient bits with 0s. The 128-bit initial round constant keys $CK_{1,2,3}$ are the 128-bit constant values of the rational part of π^{-1}. The order of using the 128-bit initial round constant keys $CK_{1,2,3}$ depends on the length of MK. Figure 3 shows the key initialization step.

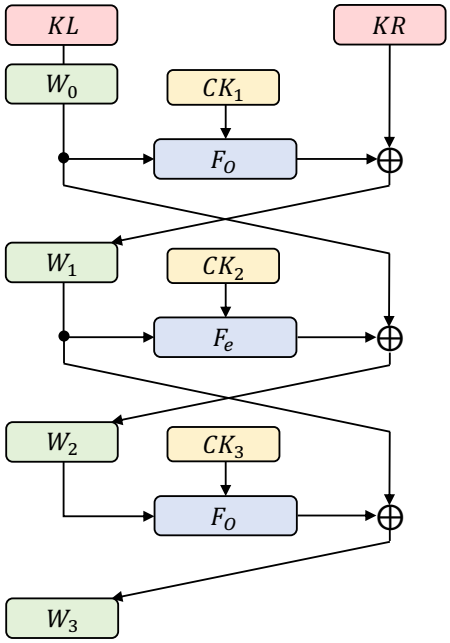

Fig. 3: Key Initialization of ARIA

In the *key generation phase*, a round key is generated and used as the key for each round. The round keys $ek_{1\sim17}$ are obtained by applying rotations ($\lll, \ggg$) and XOR operations to the initial constants $W_{0\sim3}$ generated during the key initialization step.

The round key in all ARIA instances has a size of 128 bits. The number of rounds for ARIA-128, 192, and 256 are 12, 14, and 16, respectively. Additionally, an extra round key is used in the AddRoundKey operation for the final round, resulting in a total of 13, 15, and 17 round keys for ARIA-128, 192, and 256, respectively. The round keys ek_i are generated as follows:

$$\begin{array}{ll}
ek_1 = (W_0) \oplus (W_1 \ggg 19), & ek_2 = (W_1) \oplus (W_2 \ggg 19) \\
ek_3 = (W_2) \oplus (W_3 \ggg 19), & ek_4 = (W_0 \ggg 19) \oplus (W_3) \\
ek_5 = (W_0) \oplus (W_1 \ggg 31), & ek_6 = (W_1) \oplus (W_2 \ggg 31) \\
ek_7 = (W_2) \oplus (W_3 \ggg 31), & ek_8 = (W_0 \ggg 31) \oplus (W_3) \\
ek_9 = (W_0) \oplus (W_1 \lll 61), & ek_{10} = (W_1) \oplus (W_2 \lll 61) \\
ek_{11} = (W_2) \oplus (W_3 \lll 61), & ek_{12} = (W_0 \lll 61) \oplus (W_3) \\
ek_{13} = (W_0) \oplus (W_1 \lll 31), & ek_{14} = (W_1) \oplus (W_2 \lll 31) \\
ek_{15} = (W_2) \oplus (W_3 \lll 31), & ek_{16} = (W_0 \lll 31) \oplus (W_3) \\
ek_{17} = (W_0) \oplus (W_1 \lll 19) &
\end{array} \tag{3}$$

2.2 Quantum Gates

Since in the quantum computer environment they do not provide logic gates such as AND, OR, and XOR, quantum gates are used as replacements for logic gates. This section describes commonly used quantum gates (Figure 4) 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. The Toffoli gate consists of 8 Clifford gates and 7 T gates. The T-count of the standard Toffoli gate [16] is 7 and the T-depth is 6. Many studies are reporting the implementation of Toffoli gate circuits with minimized depth and T-depth [1,5,19,21,14].

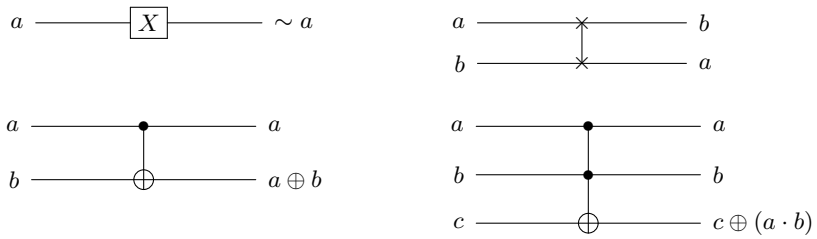

Fig. 4: Quantum gates: X (left top), Swap (right top), CNOT (left bottom) and Toffoli (right bottom) gates.

2.3 Grover's Key Search

Grover's search algorithm is a quantum algorithm that enables rapid searching for specific data within an unstructured database set N, reducing the search complexity from $O(N)$ to $O(\sqrt{N})$. When applied to an n-bit secret key search in symmetric key encryption, it reduces the search complexity from $O(2^n)$ resulting from a brute-force attack to $O(2^{n/2})$, halving the security level in theory. Grover's key search algorithm operates in three sequential steps as follows:

1. *Initialization*: Input the n-qubit key into the Hadamard gate to create a superposition of states $|\psi\rangle$ in which all 2^n computational basis states have equal amplitudes.

$$H|0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \tag{4}$$

$$|\psi\rangle = (H|0\rangle)^{\otimes n} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{5}$$

2. *Oracle Operator*: The quantum circuit for the target cipher encrypts the known plaintext using keys (prepared keys) generated through a superposition of states in the Oracle and produces ciphertext for all key values. Within the Oracle operator (U_f), the function $f(x)$ in Equation 6 compares the ciphertext generated by the circuit to the known ciphertext. The function $f(x)$ returns 0 if the generated ciphertext and the known ciphertext do not match and 1 if they do. When a match is identified, the state of the corresponding key in Equation 7, i.e., its amplitude, becomes negative because $f(x)$ is equal to 1. If no match is found, $(-1)^0$ equals 1, so the amplitude remains positive.

$$f(x) = \begin{cases} 1 \text{ if } Enc_{key}(p) = c \\ 0 \text{ if } Enc_{key}(p) \neq c \end{cases} \tag{6}$$

$$U_f(|\psi\rangle|-\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle |-\rangle \tag{7}$$

3. *Diffusion Operator*: The diffusion operator (D) serves the purpose of transforming a key state (target key state) with a negative amplitude into a symmetric state. This transformation involves computing the average value of all key states ($|s\rangle$) and then subtracting this average value from each key state element (I). During the second step, if the amplitude of the key state is initially negative, subtracting a negative number results in a positive value, thereby amplifying only the amplitude of that value.

$$D = 2|s\rangle\langle s| - I \tag{8}$$

In order to increase the probability of measuring the solution key, steps 2 and 3 must be repeated sufficiently. In general, when the number of repetitions is $\frac{\pi}{4}\sqrt{2^k}$, it has the highest measurement probability.

2.4 NIST Post-quantum Security and MAXDEPTH

In order to analyze the algorithms submitted during the post-quantum cryptography standardization, NIST provided security standards based on the security strength range specified in the existing NIST standard for symmetric cryptography in a related document [17,18]. This post-quantum security baseline is based on the complexity of quantum attacks against AES and SHA-2/3 variants. The following is a summary of the criteria for estimating the complexity of quantum attacks provided in NIST's document [18] :

- **Level 1**: Any attempt to compromise the applicable security definition should demand computational resources that are equal to or exceed the resources needed to conduct a key search on a 128-bit key block cipher, such as AES-128.
- **Level 3**: Any attempt to compromise the applicable security definition should demand computational resources that are equal to or exceed the resources needed to conduct a key search on a 192-bit key block cipher, such as AES-192.
- **Level 5**: Any attempt to compromise the applicable security definition should demand computational resources that are equal to or exceed the resources needed to conduct a key search on a 256-bit key block cipher, such as AES-256.

Grover's search algorithm is recognized as one of the most efficient quantum attacks for targeting symmetric key ciphers, and NIST also acknowledges this fact. The difficulty presented by attacks at Levels 1, 3, and 5 is assessed according to the cost needed for Grover's key search on AES-128, 192, and 256, respectively. This cost is determined by multiplying the total gate count by the depth of Grover's key search circuit. Through studies published over the past few years that optimized AES quantum circuits to reduce Grover's key search costs, NIST has defined the costs for Levels 1, 3, and 5 as $\mathbf{2^{157}}$, $\mathbf{2^{221}}$, and $\mathbf{2^{285}}$, respectively in their recent document [18] by citing the costs of depth-optimized quantum circuit implementations for AES [13] presented by Jaques et al. at Eurocrypt'20.

It should be mentioned that the quantum circuit implementation by Jaques et al. [13] has a few programming-related problems. Nevertheless, Jang et al. addressed and examined these issues in their research [9], showing that the cost values mentioned in [13] can be roughly achieved using their optimized AES quantum circuits. As far as our current understanding goes, the most notable outcomes are detailed in [9] (Level 1, 3, and 5 cost $\mathbf{2^{157}}, \mathbf{2^{192}}, \mathbf{2^{274}}$).

Additionally, we must also consider an approach called MAXDEPTH. NIST introduced a parameter called MAXDEPTH, which signifies the maximum circuit depth the quantum computer is able to execute, as an excessively large depth can lead to execution challenges in terms of time. The depth limits (i.e. MAXDEPTH) for quantum attacks provided by NIST range as follows: ($\mathbf{2^{40}} < \mathbf{2^{64}} < \mathbf{2^{96}}$).

3 Proposed Quantum Implementation of ARIA

This section describes our optimized quantum circuit implementation of ARIA. We compare the results of the previous work [2], which implemented ARIA as a quantum circuit, and examine the optimized components.

3.1 Implementation of S-box

In classical computers, the S-box of most block ciphers, including AES, employs a predefined look-up table. However, in a quantum computing environment, it is more efficient to implement the S-box using multiplicative inversion and affine transformation, primarily because of the limited number of qubits [4]. Therefore, it is necessary to obtain the multiplicative inverse and perform the affine transformation to implement the quantum circuit of ARIA's S-box.

In order to find ARIA's S-box S_1 and S_2, The inverse of x (i.e., x^{-1}) and the exponentiation value x^{247} of Equation 1 in Section 2.1 must be obtained. x^{247} in S_2 can be expressed as follows using the primitive polynomial $m(x)$ in the environment of $GF(2^8)$:

$$\begin{aligned} x^{247} &\equiv (x^{-1})^8 \equiv (((x^{-1})^2)^2)^2 \mod m(x) \\ m(x) &= x^8 + x^4 + x^3 + x + 1 \end{aligned} \tag{9}$$

Likewise, the multiplicative inverse of x in the environment of $GF(2^8)$ is equal to x^{254} The multiplicative inverse can be efficiently obtained using the Itoh-Tsujii algorithm [7]. Therefore, by applying the Itoh-Tsuji algorithm, it can be expressed as an expression consisting of square and multiplication as follows:

$$x^{-1} = x^{254} = ((x \cdot x^2) \cdot (x \cdot x^2)^4 \cdot (x \cdot x^2)^{16} \cdot x^{64})^2 \tag{10}$$

In order to increase the operation speed, the squaring operation is generally performed by converting the irreducible polynomial having linearity through modular reduction into a matrix form. Since this corresponds to a linear operation, it can be implemented as an in-place structure using only the XOR operation by using the PLU factorization.

The squaring operation in ARIA is implemented using CNOT gates and SWAP gates through modular reduction and PLU factorization [22]. This implementation utilizes 12 CNOT gates and has a circuit depth of 7. Figure 5 depicts the quantum circuit for the squaring operation in ARIA.

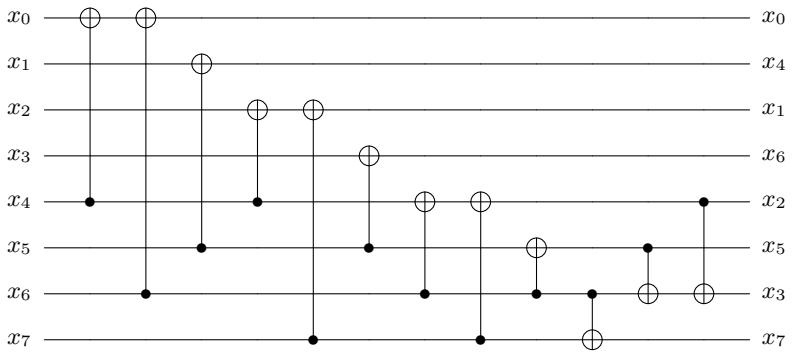

Fig. 5: Squaring in $\mathbb{F}_{2^8}/(x^8 + x^4 + x^3 + x + 1)$

For multiplication, Jang et al.'s Toffoli-depth optimized Karatsuba quantum multiplication [11], first announced at WISA'22, is used. By employing the Karatsuba algorithm, which is known for reducing the number of multiplications, the number of Toffoli gates required for a multiplication can be reduced. Jang et al.'s multiplication applies the Karatsuba algorithm recursively to perform all multiplications, i.e., AND operations, independently. In order to achieve a Toffoli depth of 1, more ancilla qubits are allocated to execute Toffoli gates in parallel. This method is only used for multiplications between quantum-quantum values.

Compared to the previous quantum implementation of ARIA [2], squaring uses the same method, so the resources used are the same, but the multiplication operation differs. In [2], the authors employed the schoolbook multiplication method [3]. In contrast, in our work, by adopting the Toffoli-depth optimized Karatsuba multiplication [11], we achieve a significant reduction in quantum resources. Table 1 compares the quantum resources required for multiplication by adopting different methods [3,11]. In Table 1, we can see that overall quantum resources have been reduced, and, in particular, Toffoli-depth have been optimized.

Table 1: Quantum resources required for multiplication.

Source	#Clifford	#T	Toffoli depth	Full depth
CMMP [2]	435	448	28	195
J++ [11]	390	189	1	28

※: The multiplication size n is 8.

After obtaining the exponentiation values, matrix-vector multiplication between the exponentiation and the matrix is computed by applying PLU factorization because it involves the product of classical and quantum values. Multiplying vector would have originally required applying 8 CNOT gates, but by taking advantage of the fact that the given vector is a constant, resources are saved by applying X gates only to the positions where inversion is necessary.

3.2 Implementation of Diffusion Layer

ARIA's diffusion function $A : GF(2^8)^{16} \rightarrow GF(2^8)^{16}$ is expressed as a 16×16 binary matrix product. For implementing matrix-vector multiplication in quantum, we utilize PLU factorization just as we implemented matrix-vector multiplication of SEED.

Since one element of the binary matrix is a byte, in order to multiply with the input bit, the byte must be converted to a bit unit and the calculation proceeded. To do so, the calculation proceeds assuming that the element 0 in the matrix represents an 8×8 zero matrix, and the element 1 in the matrix represents an 8×8 identity matrix. In the diffusion layer, only CNOT gates are utilized. Since 96 CNOT operations are required per byte, a total of 768 (96×8) CNOT gates are used. The implementation of the diffusion layer is the same as in [2].

3.3 Implementation of Key Schedule

In the key initialization phase, the 128-qubit W_1, W_2, and W_3 are generated using round functions. Since K_L is used only for the generation of W_0, instead of allocating new qubits for W_0, K_L is utilized as a substitute, resulting in a reduction of 128 qubits. In addition, when performing the XOR operation of KR and $W_{1\sim3}$, since KR is a constant, the X gate are applied to W1 only when the bit of KR is 1. By replacing the CNOT gates with cheaper X gates, the number of gates and gate cost are reduced. Chauhan et al. [2] utilized 379 X gates and 159,560 CNOT gates ($30{,}536 + 6 \times 21{,}504$). In contrast, our implementation in the key initialization stage employs 192 X gates and 87,544 CNOT gates, leading to a reduction of approximately 49% in X gates and about 45% in CNOT gates compared to [2].

In the key generation stage, a round key ek used as an encryption key for each round is generated using $W_{0\sim3}$. If W_0 is used in the generation of ek, we reduce the gate cost by applying the X gates instead of the CNOT gates as in the generation of W_0.

Since the value of ek is different for each round, new qubits must be allocated and stored each time. However, instead of allocating new qubits for ek every round, we initialize and reuse the qubits by performing a reverse operation on the round key generation at the end of every round. Since the reverse operation on key generation, which is related to CNOT gates and X gates, has little effect on the depth, it is more efficient to perform the reverse operation than to allocate 128 ancilla qubits every round.

Algorithm 1: Quantum circuit implementation of key schedule for ARIA.

Input: master key MK, key length l, vector a, b, ancilla qubit anc, round number r
Output: round key ek
$//MK[:128]$, $MK[l-128:l]$ is KL, KR, respectively.

$W_1 \leftarrow F_o(MK[:128], a, b, anc)$ ▷ Key Initialization
Constant_XOR($W_1[l-128:128]$, $MK[l-128:l]$)

$W_2 \leftarrow F_e(W_1, a, b, anc)$
$W_2 \leftarrow$ CNOT128($MK[:128], W_2$)

$W_3 \leftarrow F_o(W_2, a, b, anc)$
$W_3 \leftarrow$ CNOT128(W_1, W_3)

$num = [19, 31, 67, 97, 109]$ ▷ Key Generation
for $0 \leq i \leq$ r **do**
    if i % 4 == 0 **then**
        Constant_XOR(ek, $MK[:128]$)
    else
        $ek \leftarrow$CNOT128($W_{(i\%4)}, ek$)
    end if
    $ek \leftarrow$ CNOT128($W_{(i+1)\%4} \ggg num[i\%4], ek$)
end for
return ek

4 Evaluation

In this section, we estimate and analyze the quantum circuit resources for ARIA. The proposed quantum circuits cannot yet be implemented in large-scale quantum computers. Therefore, we use ProjectQ, a quantum programming tool, on a classical computer instead of real quantum computer to implement and simulate quantum circuits. A large number of qubits can be simulated using ProjectQ's own library, `ClassicalSimulator`, which is restricted to simple quantum gates (such as X, SWAP, CNOT, and Toffoli). With the aid of this functionality, the `ClassicalSimulator` is able to test the implementation of a quantum circuit by classically computing the output for a particular input. For the estimation of quantum resources, another internal library called `ResourceCounter` is needed. `ResourceCounter` solely counts quantum gates and circuit depth, doesn't run quantum circuits, in contrast to `ClassicalSimulator`.

4.1 Performance of the Proposed Quantum Circuit

Table 2 and 3 represent the quantum resources required to implement our proposed quantum circuits for ARIA. These tables compare the quantum resources between the quantum circuit proposed by Chauhan et al. [2] and our proposed quantum circuit. Table 2 shows quantum resources for ARIA at the NCT (NOT, CNOT, Toffoli) gate level, while Table 3 presents quantum resources for ARIA at the Clifford+T level, achieved by decomposing the Toffoli gate. In [2], the decomposed quantum resources were not explicitly provided, so the quantum resources in Table 3 are extrapolated based on the information provided in the paper [2]. Furthermore, our implementation places a primary emphasis on circuit depth optimization while carefully considering the balance with qubit utilization. We conduct evaluations, including metrics such as Toffoli depth(TD) multiplied by qubit count(M), to assess these trade-offs.

Table 2: Required quantum resources for ARIA quantum circuit implementation

Cipher	Source	#X	#CNOT	#Toffoli	Toffoli depth	#Qubit	Depth	TD-M cost
ARIA-128	CS[2]	1,595	231,124	157,696	4,312	1,560	9,260	6,726,720
	This work	1,408	285,784	25,920	60	29,216	3,500	1,752,960
ARIA-192	CS[2]	1,851	273,264	183,368	5,096	1,560	10,948	7,949,760
	This work	1,624	324,136	29,376	68	32,928	3,978	2,239,104
ARIA-256	CS[2]	2,171	325,352	222,208	6,076	1,688	13,054	10,256,288
	This work	1,856	362,488	32,832	76	36,640	4,455	2,784,640

Table 3: Required decomposed quantum resources for ARIA quantum circuit implementation

Variant		#Cliford	#T	T-depth	#Qubit	Full depth
ARIA-128	CS[2]$^{\diamondsuit}$	1,494,287	1,103,872	17,248	1,560	37,882
	This work	494,552	181,440	240	29,216	4,650
ARIA-192	CS[2]$^{\diamondsuit}$	1,742,059	1,283,576	20,376	1,560	44,774
	This work	560,768	205,632	272	32,928	5,285
ARIA-256	CS[2]$^{\diamondsuit}$	2,105,187	1,555,456	24,304	1,688	51,666
	This work	627,000	229,824	304	36,640	5,919

$\diamondsuit$ Extrapolated result

4.2 Evaluation of Grover's Search Complexity

In this section, we evaluate the quantum security of ARIA by estimating the cost of Grover's key search for this algorithm. As described in Section 2.3, the overhead of the diffusion operator can be considered insignificant

compared to the overhead of the oracle, so it is disregarded when estimating the cost of the Grover's key search. Therefore, the optimal number of iterations for Grover's key search for a cipher using a k-bit key is approximately $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$.

According to [13], finding a unique key requires r plaintext–ciphertext pairs, where r needs to be at least $\lceil$key size/block size$\rceil$. To calculate the quantum resources required for Grover's key search in the block cipher, the decomposed quantum resources need to be multiplied by 2, r, and $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$.

In the case of ARIA with the key size of 192 or 256 bits, the value of r is 2, indicating that the multiplication by r cannot be omitted. Therefore, the Grover's key search cost for ARIA is approximately Table 3 $\times r \times 2 \times \lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$(see Table 4).

Table 4: Cost of the Grover's key search for ARIA

Cipher	Source	Total gates	Full depth	Cost (complexity)	#Qubit	FD-M cost
ARIA-128	CS[2]	$1.998 \cdot 2^{85}$	$1.816 \cdot 2^{79}$	$1.814 \cdot 2^{165}$	1,561	$1.26 \cdot 2^{86}$
	This work	$1.117 \cdot 2^{84}$	$1.783 \cdot 2^{76}$	$1.991 \cdot 2^{160}$	29,217	$1.313 \cdot 2^{84}$
ARIA-192	CS[2]	$1.146 \cdot 2^{119}$	$1.073 \cdot 2^{112}$	$1.23 \cdot 2^{231}$	3,121	$1.489 \cdot 2^{118}$
	This work	$1.2 \cdot 2^{117}$	$1.013 \cdot 2^{109}$	$1.216 \cdot 2^{226}$	65,857	$1.677 \cdot 2^{116}$
ARIA-256	CS[2]	$1.384 \cdot 2^{151}$	$1.238 \cdot 2^{144}$	$1.714 \cdot 2^{295}$	3,377	$1.921 \cdot 2^{150}$
	This work	$1.336 \cdot 2^{149}$	$1.135 \cdot 2^{141}$	$1.516 \cdot 2^{290}$	72,081	$1.043 \cdot 2^{149}$

Cost is an indicator that can be compared with the security criteria provided by NIST. After comparing with the quantum attack cost ($\mathbf{2^{157}}$, $\mathbf{2^{221}}$, and $\mathbf{2^{285}}$) described in Section 2.4, it can be confirmed that all instances of ARIA attain the suitable level of security for their respective key sizes.

To take NIST's MAXDEPTH (mentioned in Section 2.4) into account, one cannot disregard parallelization. When comparing Full depth(FD) and NIST MAXDEPTH in Table 4, only ARIA-128 meets the MAXDEPTH requirement (ARIA-128 $< 2^{96}$). For ARIA-192 and ARIA-256, which do not meet the MAXDEPTH limit, Grover's key search must be performed in parallel. However, according to [13,10], parallelization of Grover's key search is highly inefficient; therefore, one should minimize the costs of relevant metrics (e.g., $FD^2 - M$, $TD^2 - M$).

5 Conclusion

In this paper, we propose optimized quantum circuit implementation for ARIA, focusing on circuit depth optimization. We utilize various techniques such as optimized multiplication and squaring methods in binary fields, along with parallelization, to reduce both Toffoli and full depths while ensuring a reasonable number of qubits. As a result, our quantum circuit implementation for ARIA achieves the depth improvement of over 88.2% and Toffoli depth by more than 98.7% compared to the implementation proposed in Chauhan et al.'s SPACE'20 paper [2]. Based on our quantum circuits, we estimate the quantum resources and the cost of Grover's attacks for the proposed circuit. We then evaluate the security strength based on the criteria provided by NIST. We demonstrate that ARIA achieves post-quantum security levels 1, 3, and 5, respectively, for all key sizes: 128, 192, and 256 bits (according to the recent standards [18]). Additionally, we have shown that only ARIA-128 satisfies the MAXDEPTH limit.

Our future plan involves optimizing ARIA's quantum circuits further, with greater consideration for the MAXDEPTH limit.

References

1. Amy, M., Maslov, D., Mosca, M., Roetteler, M., Roetteler, M.: 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 **32**(6), 818–830 (Jun 2013). https://doi.org/10.1109/tcad.2013.2244643, http://dx.doi.org/10.1109/TCAD.2013.2244643 5

2. Chauhan, A.K., Sanadhya, S.K.: Quantum resource estimates of grover's key search on aria. In: Security, Privacy, and Applied Cryptography Engineering: 10th International Conference, SPACE 2020, Kolkata, India, December 17–21, 2020, Proceedings 10. pp. 238–258. Springer (2020) 1, 6, 7, 8, 9, 10
3. Cheung, D., Maslov, D., Mathew, J., Pradhan, D.K.: 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) 7
4. Chung, D., Lee, S., Choi, D., Lee, J.: Alternative tower field construction for quantum implementation of the aes s-box. IEEE Transactions on Computers **71**(10), 2553–2564 (2021) 6
5. Fedorov, A., Steffen, L., Baur, M., da Silva, M.P., Wallraff, A.: Implementation of a Toffoli gate with superconducting circuits. Nature **481**(7380), 170–172 (2012) 5
6. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates (2015) 1
7. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in gf (2m) using normal bases. Information and computation **78**(3), 171–177 (1988) 6
8. Jang, K., Baksi, A., Kim, H., Seo, H., Chattopadhyay, A.: Improved quantum analysis of speck and lowmc (full version). Cryptology ePrint Archive (2022) 1
9. Jang, K., Baksi, A., Song, G., Kim, H., Seo, H., Chattopadhyay, A.: Quantum analysis of aes. Cryptology ePrint Archive (2022) 1, 6
10. Jang, K., Kim, D., Oh, Y., Lim, S., Yang, Y., Kim, H., Seo, H.: Quantum implementation of aim: Aiming for low-depth. Cryptology ePrint Archive (2023) 10
11. Jang, K., Kim, W., Lim, S., Kang, Y., Yang, Y., Seo, H.: Optimized implementation of quantum binary field multiplication with toffoli depth one. In: International Conference on Information Security Applications. pp. 251–264. Springer (2022) 7
12. Jang, K., Song, G., Kim, H., Kwon, H., Kim, H., Seo, H.: Efficient implementation of present and gift on quantum computers. Applied Sciences **11**(11), 4776 (2021) 1
13. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: 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) 1, 6, 10
14. Jones, C.: Low-overhead constructions for the fault-tolerant toffoli gate. Physical Review A **87**(2), 022328 (2013) 5
15. Kwon, D., Kim, J., Park, S., Sung, S.H., Sohn, Y., Song, J.H., Yeom, Y., Yoon, E.J., Lee, S., Lee, J., et al.: New block cipher: Aria. In: Information Security and Cryptology-ICISC 2003: 6th International Conference, Seoul, Korea, November 27-28, 2003. Revised Papers 6. pp. 432–445. Springer (2004) 2
16. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2002) 5
17. 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 6
18. 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 6, 10
19. Ralph, T., Resch, K., Gilchrist, A.: Efficient Toffoli gates using qudits. Physical Review A **75**(2), 022313 (2007) 5
20. Roy, S., Baksi, A., Chattopadhyay, A.: Quantum implementation of ascon linear layer. Cryptology ePrint Archive (2023) 1
21. Selinger, P.: Quantum circuits of t-depth one. Physical Review A **87**(4), 042302 (2013) 5
22. Van Hoof, I.: Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic toffoli gate count. arXiv preprint arXiv:1910.02849 (2019) 7
23. Yang, Y., Jang, K., Kim, H., Song, G., Seo, H.: Grover on sparkle. In: International Conference on Information Security Applications. pp. 44–59. Springer (2022) 1