Improved Quantum Analysis of ARIA

Yujin Oh, Kyungbae Jang, and Hwajeong Seo

Hansung University, Seoul (02876), South Korea

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

Abstract. As advancements in quantum computing present potential threats to current cryptographic systems, it is necessary to reconsider and adapt existing cryptographic frameworks. Among these, Grover's algorithm reduces the attack complexity of symmetric-key encryption, making it crucial to evaluate the security strength of traditional symmetric-key systems.

In this paper, we implement an efficient quantum circuit for the ARIA symmetric-key encryption and estimate the required quantum resources. Our approach achieves a reduction of over 61% in full depth and over 65.5% in qubit usage compared to the most optimized previous research. Additionally, we estimate the cost of a Grover attack on ARIA and evaluate its post-quantum security strength.

Keywords: Quantum Circuit · ARIA · Boyar-Peralta · Grover's key search.

1 Introduction

Current cryptographic systems rely on the computational difficulty of specific mathematical problems, which require infeasible amounts of time and resources for classical computers to solve. However, quantum computers, leveraging principles such as superposition and entanglement, possess significantly enhanced computational capabilities. By employing quantum algorithms, they can efficiently solve certain mathematical challenges that form the foundation of current cryptography. Notably, Shor's algorithm [13] enables the efficient factorization of large composite numbers and solves the discrete logarithm problem, which are fundamental to the security of public-key cryptosystems such as RSA and ECC. As a response to the potential threats posed by quantum computing, NIST has introduced the Post-Quantum Cryptography (PQC) Standardization project to develop cryptographic systems resilient to quantum attacks.

In symmetric-key cryptography, Grover's algorithm [5] can reduce the attack complexity by a factor of the square root compared to classical computers. To address this, NIST has defined security levels to ensure symmetric encryption remains robust against quantum computing advancements.

This paper proposes an optimized quantum circuit for ARIA, a Korean symmetric key algorithm, which is one of the modules verified by the Korean Cryptographic Module Validation Program (KCMVP). We minimize the quantum

circuit depth of ARIA while keeping a qubit count reasonable. This approach aligns with the optimization of Grover's algorithm, which is explained in Section 2.3. We compare our implementation with previous studies and assess the quantum resources required. Additionally, based on the proposed quantum circuit, we estimate the Grover attack cost and evaluate the post-quantum security strength of ARIA according to the NIST security levels.

Our Contributions We focus on optimizing the depth of the ARIA circuit using various optimized methods. First, we apply Boyar-Peralta algorithm to the S-boxes. This is the first time that this algorithm has been used for all S-boxes in ARIA. Furthermore, we reduce the circuit depth by employing out-of-place operations and parallelization. Finally, we estimate the quantum resources and assess the security strength in comparison to previous works. To the best of our knowledge, our work represents the most optimized quantum implementation of ARIA so far.

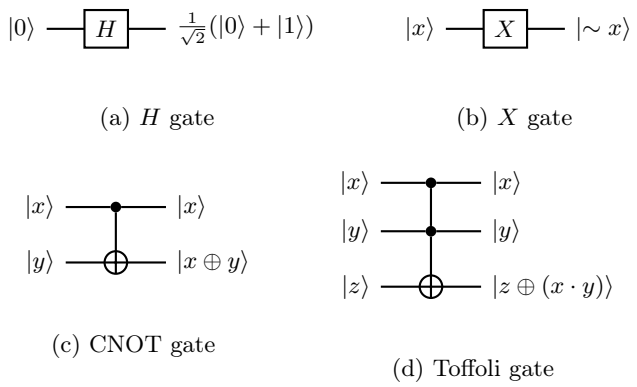
2 Background

2.1 Quantum Gates

Quantum gates are the basic elements of quantum circuits, similar to classical logic gates but operating on qubits. The Hadamard gate (H) creates a superposition by transforming a basis state ($|0\rangle$ or $|1\rangle$) into an equal superposition of both states. For example, when applied to $|0\rangle$, it results in the state $H|0\rangle = \frac{(|0\rangle+|1\rangle)}{\sqrt{2}}$. The X gate acts as the quantum equivalent of a classical NOT gate. It flips the state of a qubit, inverting it from $|0\rangle$ to $|1\rangle$ or from $|1\rangle$ to $|0\rangle$. The CNOT gate (Controlled-NOT) flips the state of a target qubit if the control qubit is in the $|1\rangle$ state. The control qubit remains unchanged, while the target qubit is flipped. This is a two-qubit gate that is essential for creating entanglement. The Toffoli gate (CCNOT) is a three-qubit gate that flips the state of the target qubit only if both control qubits are in the $|1\rangle$ state. It acts like a logical AND operation followed by a XOR operation on the target qubit. The Toffoli gate can be decomposed into a combination of gates such as H, CNOT and T gates, as shown in 2.

2.2 The Grover algorithm

The Grover's algorithm is a quantum algorithm that significantly reduces the computational complexity of brute-force key search attacks on symmetric-key cryptography. It provides a quadratic speedup compared to classical algorithms. For a cipher with a k-bit key, classical attacks require $O(2^k)$ operations. However, by leveraging Grover's algorithm, the complexity is reduced to $O(\sqrt{2^k})$, effectively halving the security level in bits. For example, in the case of AES-128, which has a 128-bit key, the security level under classical attacks would require



(a) H gate (b) X gate

(c) CNOT gate (d) Toffoli gate

Fig. 1: Quantum gates

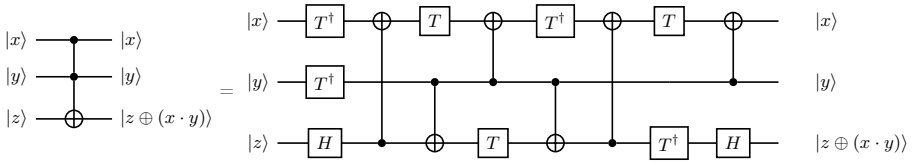


Fig. 2: Decomposed Toffoli gate

$O(2^{128})$ operations. With Grover's Algorithm, the effective complexity drops to $O(2^{64})$, reducing its effective security to 64 bits against quantum attacks. The Grover algorithm consists of three main steps as follows

1. *Input setting*:
   Prepare a k-qubit state by creating a superposition of all possible states. This is achieved by applying Hadamard gates to all qubits, resulting in an equal probability amplitude for every potential solution.

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

2. The oracle implements the target function and processes input states in superposition, providing a result that indicates the solution. This is achieved by encoding the function into a quantum circuit composed of quantum gates. When the circuit identifies a valid solution (i.e., if $f(x) = 1$), it flips the amplitude of the corresponding input in the superposition.

$$f(x) = \begin{cases} 1 \text{ if } ENC_{key}(x) = \text{target output} \\ 0 \text{ if } ENC_{key}(x) \neq \text{target output} \end{cases} \tag{2}$$

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

3. The diffusion operator amplifies the probability of measuring the solution that the oracle has returned. It can be achieved by increasing the amplitude of the correct answer in the quantum state.

2.3 NIST security level

NIST defines post-quantum security levels to address quantum attacks [10,11]. Levels 1, 3, and 5 correspond to the complexity of Grover's search on AES, as summarized in Table 1. Levels 2 and 4, on the other hand, relate to the complexity of collision searches for SHA-2/3. Since this study focuses on estimating the attack cost on ARIA symmetric keys, levels 2 and 4 are excluded from the discussion.

Table 1: NIST Security Levels

Level	Cipher	Quantum Cost (Gate count × Full depth)
Level 1	AES-128	$2^{170} \rightarrow 2^{157}$
Level 3	AES-192	$2^{233} \rightarrow 2^{221}$
Level 5	AES-256	$2^{298} \rightarrow 2^{285}$

The complexity of attacks for security Levels 1, 3, and 5 is determined by the cost of applying Grover's key search to AES-128, AES-192, and AES-256, respectively. This cost is calculated as the product of the total gate count and the circuit depth required for Grover's key search. Based on the AES quantum circuit design by [4], NIST initially estimated the computational costs for Levels 1, 3, and 5 to be 2^{170}, 2^{233} and 2^{298}. In recent years, however, substantial progress has been made in refining the quantum circuit designs for AES, aiming to enhance efficiency and reduce resource requirements. For this reason, NIST has revised the security level costs [11]. In particular, based on the costs reported in Jaques at al. [7], NIST updated the quantum attack costs for AES-128, AES-192, and AES-256 to 2^{157}, 2^{221}, and 2^{285}, respectively.

In addition, NIST has defined the maximum circuit depth, referred to as MAXDEPTH. NIST classifies the quantum attack depth limitations indicated by MAXDEPTH into specific ranges: 2^{157}, 2^{221} and 2^{285}. This classification reflects the fact that the substantial depth required in Grover's attack, due to numerous sequential iterations, makes the attack practically difficult beyond these thresholds.

Considering this, the quantum circuit depth for Grover's search should not exceed 2^{96}, the highest estimated limit of MAXDEPTH. If the depth exceeds

the limit, it may be necessary to consider parallelizing the Grover's search. The trade-off metrics for quantum circuits are adjusted by multiplying them with the circuit depth for parallelizing Grover's algorithm. In this paper, we also estimate the adjusted trade-off metrics for Grover's parallelization

2.4 Description of ARIA

ARIA [8] is a Korean symmetric key cipher recognized within the Korean Cryptographic Module Validation Program (KCMVP) as a standard encryption algorithm. It employs a Substitution-Permutation Network (SPN) structure, similar to AES (Advanced Encryption Standard), as its design was influenced by AES principles to ensure high performance and strong security. ARIA processes data in 128-bit blocks for both input and output and supports three key sizes: 128 bits, 192 bits, and 256 bits, which determine the number of encryption rounds as 12, 14, and 16, respectively. Its encryption mechanism consists of three core components: the substitution layer, the diffusion layer, and the key schedule, which work together to ensure secure and efficient data encryption. Compared to AES, ARIA introduces unique design features such as the use of two distinct S-boxes, alternating between them to enhance resistance against cryptanalytic attacks.

Substitution layer ARIA applies two different types of substitution layers, (LS, LS, LS, LS), $(LS^{-1}, LS^{-1}, LS^{-1}, LS^{-1})$, each applied in sequence. The LS layer contains two S-boxes, S_1, S_2 , along with their inverses, S_1^{-1}, S_2^{-1}. These S-boxes combine the principles of Rijndael S-boxes and Serpent S-boxes, providing robust security while maintaining efficiency. The S-boxes consist of the inversion of x and an affine transformation, which involves an 8×8 matrix and XOR with 8×1 vector. All operations are performed in $GF(2^8)$, using the same irreducible polynomial $x \in \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$ as in AES. The equations for each S-box are expressed as follows :

$$S_1(x) = L_1 \cdot x^{-1} \oplus a,$$

$$\text{where} \quad L_1 = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \quad \text{and} \quad a = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{4}$$

$$S_1^{-1}(x) = (L_1^{-1} \cdot (x \oplus a))^{-1},$$

$$\text{where} \quad L_1^{-1} = \begin{pmatrix} 0\,0\,1\,0\,0\,1\,0\,1 \\ 1\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,0\,0\,1 \\ 1\,0\,1\,0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,0\,1\,0\,1\,0\,0\,1 \\ 1\,0\,0\,1\,0\,1\,0\,0 \\ 0\,1\,0\,0\,1\,0\,1\,0 \end{pmatrix} \quad \text{and} \quad a = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{5}$$

$$S_2(x) = \mathbf{B} \cdot x^{247} \oplus b = \mathbf{B} \cdot \mathbf{C} \cdot (x^{-1})^8 \oplus b = \mathbf{L_2} \cdot x^{-1} \oplus b$$

$$\text{where} \quad \mathbf{L_2} = \begin{pmatrix} 0\,1\,0\,1\,0\,1\,1\,1 \\ 0\,0\,1\,1\,1\,1\,1\,1 \\ 1\,1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,0\,0\,0\,1\,1 \\ 0\,1\,0\,0\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1\,1\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,1\,1\,0 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{6}$$

$$S_2^{-1}(x) = (L_2^{-1} \cdot (x \oplus b))^{-1},$$

$$\text{where} \quad \mathbf{L_2^{-1}} = \begin{pmatrix} 0\,0\,0\,1\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,0 \\ 0\,0\,9\,0\,1\,0\,1\,0 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,0\,1\,1\,0\,0 \\ 0\,1\,1\,0\,1\,0\,1\,1 \\ 1\,0\,1\,1\,1\,1\,0\,1 \\ 1\,0\,0\,1\,0\,0\,1\,1 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{7}$$

Diffusion Layer The diffusion layer is described by an invertible transformation A : $\mathrm{GF}(2^8)^{16} \rightarrow \mathrm{GF}(2^8)^{16}$, which maps an input vector $(x_0, x_1, ..., x_{15})$ to an output vector $(y_0, y_1, ..., y_{15})$.

$$\begin{aligned}
y_0 &= x_3 \oplus x_4 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{13} \oplus x_{14}, & y_8 &= x_0 \oplus x_1 \oplus x_4 \oplus x_7 \oplus x_{10} \oplus x_{13} \oplus x_{15}, \\
y_1 &= x_2 \oplus x_5 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{12} \oplus x_{15}, & y_9 &= x_0 \oplus x_1 \oplus x_5 \oplus x_6 \oplus x_{11} \oplus x_{12} \oplus x_{14}, \\
y_2 &= x_1 \oplus x_4 \oplus x_6 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{15}, & y_{10} &= x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_8 \oplus x_{13} \oplus x_{15}, \\
y_3 &= x_0 \oplus x_5 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus x_{13} \oplus x_{14}, & y_{11} &= x_2 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_9 \oplus x_{12} \oplus x_{14}, \\
y_4 &= x_0 \oplus x_2 \oplus x_5 \oplus x_8 \oplus x_{11} \oplus x_{14} \oplus x_{15}, & y_{12} &= x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_9 \oplus x_{11} \oplus x_{12}, \\
y_5 &= x_1 \oplus x_3 \oplus x_4 \oplus x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}, & y_{13} &= x_0 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{10} \oplus x_{13}, \\
y_6 &= x_0 \oplus x_2 \oplus x_7 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{13}, & y_{14} &= x_0 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_9 \oplus x_{11} \oplus x_{14}, \\
y_7 &= x_1 \oplus x_3 \oplus x_6 \oplus x_8 \oplus x_{11} \oplus x_{12} \oplus x_{13}, & y_{15} &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_{10} \oplus x_{15},
\end{aligned} \tag{8}$$

Additionally, the process can be represented by multiplying a 16×16 binary matrix, as shown below.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \end{pmatrix} = \begin{pmatrix} 0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,0\,1 \\ 1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0 \\ 1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,1 \\ 1\,0\,1\,0\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,1\,0\,1\,0\,0\,1\,1\,1\,0\,0 \\ 1\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,1 \\ 1\,1\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0 \\ 0\,0\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,1\,1\,1\,0\,0\,1\,0\,1\,0\,0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,0\,0\,0 \\ 1\,0\,0\,1\,0\,0\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0 \\ 1\,0\,0\,1\,1\,1\,0\,0\,0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,1\,0\,0\,0\,0\,1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{pmatrix} \tag{9}$$

KeySchedule The key scheduling of ARIA comprises two parts: initialization and round key generation. In the initialization of the ARIA key schedule, a 3-round Feistel structure is used to generate four 128-bit values: W0, W1, W2, and W3, from the master key (MK). The size of MK can be 128, 192, or 256 bits. To create the key components, MK is divided into two parts: KL and KR. The first 128 bits of MK are used to form KL, and the remaining bits are assigned to KR. If KR has less than 128 bits, the remaining space is padded with zeros.

The round key generation in ARIA utilizes the four values W_0, W_1, W_2, and W_3 to generate the encryption round keys (ek_i), each consisting of 128 bits. The number of rounds in ARIA depends on the size of the master key: 12 rounds for a 128-bit key, 14 rounds for a 192-bit key, and 16 rounds for a 256-bit key. Additionally, an extra round key is required for the final round of encryption, leading to a total of 13, 15, or 17 round keys, respectively. The generation of round keys in ARIA is performed through a combination of rotation and XOR operations, as shown in Equation 10.

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

3 Quantum Circuit Implementation of ARIA

In this section, we describe our equantum circuit implementation of ARIA. Our primary focus is to strategically optimize the circuit depth, aiming to significantly enhance the efficiency of the Grover's key search.

3.1 Sboxes using Boyar-Perlata

To implement the S-box for ARIA, the inversion of x must be computed. Typically, this is done using the Itoh-Tsuji algorithm, which involves multiplication and squarings. Previous studies [3,16] have utilized the Itoh-Tsuji algorithm. In [12] , since S_1 is identical to the AES S-box, they applied the AES S-box optimization technique using Jang et al.'s implementation with the Boyar-Perlata method [1,2]. However, they still used the Itoh-Tsuji algorithm for S_2, as the affine matrices for S_1 and S_2 differ. In our approach, we also adopt Jang's approach, which utilizes Boyar-Perlata but apply this approach across all S-boxes unlike [12].

The Boyar-Peralta S-Box offers an alternative design to the AES S-Box with a reduced circuit depth. This S-Box employs a novel logic minimization approach that can optimize complex combinational logic problems. Their approach involves a two-step process: first, reducing nonlinear gates, followed by the minimization of linear gates. They derive a representation $S(x) = B \cdot F(U \cdot x) + [11000110]^T$. In this case, U is a linear transformation matrix that maps 8 bits to 22 bits, F is a nonlinear transformation that maps 22 bits to 18 bits, and B is a linear layer that maps 18 bits back to 8 bits. For S_1 in ARIA, which is identical to AES, this can also be expressed as: $S_1(x) = L_1 \cdot x^{-1} = B \cdot F(U \cdot x)$.

Using this algorithm, S_2 can also be derived, as shown in Equation 6. To begin, the inversion of x is computed using S_1 as follows:

$$
L_1 \cdot x^{-1} + a \rightarrow L_1^{-1} \cdot (L_1 \cdot x^{-1}) \rightarrow x^{-1} \tag{11}
$$

Next, $L_2 \cdot x^{-1} + b$ is calculated. The complete circuit for computing S_2 is illustrated in Figure 3. In the circuit, U_0 means to operate $B \cdot F(U \cdot x)$. Following this method, new equations for S_1^{-1} and S_2^{-1} are also derived, as shown in Equation 12. Their computation circuits are presented in Figures 4 and 5, respectively.

$$\begin{aligned} S_1^{-1}(x) &= (L_1^{-1} \cdot (x+a))^{-1} \rightarrow L_1^{-1} \cdot L_1 \cdot (L_1^{-1} \cdot (x+a))^{-1} \\ S_2^{-1}(x) &= (L_2^{-1} \cdot (x+b))^{-1} \rightarrow L_1^{-1} \cdot L_1 \cdot (L_2^{-1} \cdot (x+b))^{-1} \end{aligned} \tag{12}$$

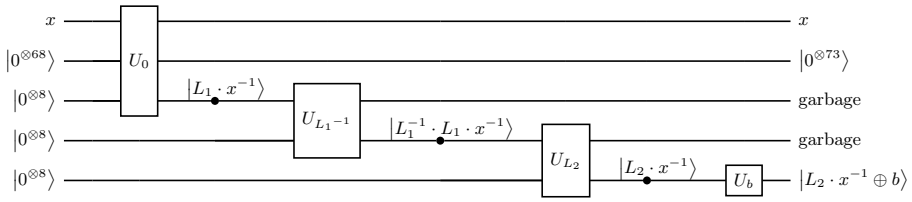


Fig. 3: The design of a circuit for implementing S_2 based on S_1

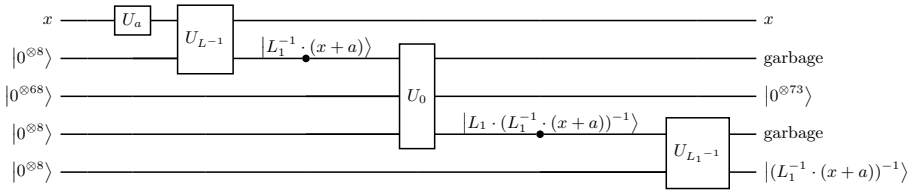


Fig. 4: The design of a circuit for implementing S_1^{-1} based on S_1

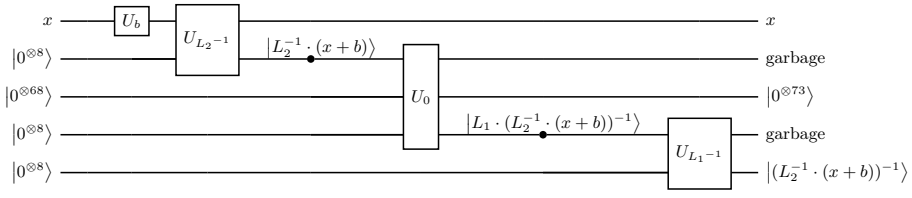


Fig. 5: The design of a circuit for implementing S_2^{-1} based on S_1

The Boyar-Peralta algorithm has inspired various efforts [6,7,9,17] to enhance the efficiency of AES quantum circuits. Building on this progress, we leverage the method proposed by Jang et al. [6], which is highly effective at minimizing depth with a practical number of qubits, and apply it to the ARIA S-box. This approach enables us to achieve notable reductions in depth, qubit count, and gate count compared to previous works. Specifically, the S-box requires 68 ancilla qubits, 8 output qubits and the depth of 33. To further optimize the depth, we implement the matrix multiplication with additional ancilla qubits for output, allocating 8 ancilla qubits per matrix multiplication. During this process, we maximize the parallel execution of CNOT operations (XOR) wherever feasible, ensuring improved performance.

As shown in Table 2, the quantum resource of circuit that we design achieves the lowest depth and gate count. Also, we reduce the qubit count compared [16,12].

Table 2: Quantum resources required for implementations of a S-box.

S-box	Method	Source	#CNOT	#X	#Toffoli	Toffoli depth	#Qubit	depth
S_1	Itoh-Tsujii	[3]	569	4	448	196	40	391
		[16]	1114	4	108	4	162	151
		[12]	1106	4	108	4	170	137
	Boyar-Peralta	[12]	**162**	4	**34**	4	**84**	**33**
		Ours						
S_1^{-1}	Itoh-Tsujii	[3]	561	4	448	4	40	-
		[16]	1106	4	108	4	162	146
		[12]	1,090	4	108	4	170	135
	Boyar-Peralta	[12]	190	4	34	4	84	55
		Ours	**162**	4	**34**	4	**84**	**40**
S_2	Itoh-Tsujii	[3]	569	4	448	196	40	-
		[16]	1123	4	108	4	162	157
		[12]	1105	4	108	4	170	139
	Boyar-Peralta	**Ours**	**225**	4	**34**	4	**100**	**44**
S_2^{-1}	Itoh-Tsujii	[3]	570	4	448	196	40	-
		[16]	1115	4	108	4	162	150
		[12]	1106	4	108	4	170	137
	Boyar-Peralta	**Ours**	**218**	4	**34**	4	**100**	**43**

3.2 Parallelization in Substitution Layer

ARIA uses two different substitution layers, (LS , LS , LS , LS) and (LS^{-1}, LS^{-1}, LS^{-1}, LS^{-1}) , with LS defined as $LS = (S_1, S_2, S_1^{-1}, S_2^{-1})$. Consequently, each substitution layer makes use of 16 S-boxes. As mentioned in Secotion 3.1,

one S-box uses 68 ancilla qubits and 8 output qubits. When a single set of ancilla qubits is allocated, all the S-boxes are processed in sequence, resulting in a considerable increase in the circuit depth.

Based on the importance of depth optimization mentioned in Section , we minimize the depth by executing all S-boxes in parallel within each substitution layer, as demonstrated in previous studies [16,12]. To allow for the parallel execution of S-boxes within the substitution layer, it is necessary to allocate additional ancilla qubits for each S-box. Since S_1 requires 68 ancilla qubits, a total of 1088 (68×16) ancilla qubits are needed.

The substitution layer operates in every round. Therefore, ancilla qubits must be allocated for each round, as the ancilla qubits are not clean. Although allocating ancilla qubits in each round can reduce both depth and qubit count compared to [16,12], it seems to overuse qubits. Qubit count is crucial in optimizing quantum circuits, not just depth, and it is important to carefully balance the trade-off. In light of this, we effectively reduce the additional qubit overhead by reusing the ancilla qubits through reverse operations, without increasing the circuit depth.

3.3 Reusing ancilla qubits through reverse operations

We can optimize the circuit depth by parallel operations of 16 S-boxes. After performing the S-box operations, the ancilla qubits are left in an unclean state. Thus, to process the S-boxes in parallel in the substitution layer used in the next round, we have two options. First, we allocate ancilla qubit sets in each round. In this case, ARIA-128 requires 16,320 ($1088 \times (12 + 3)$), ARIA-192 requires 18,496 ($1088 \times (14 + 3)$), and ARIA-256 requires 20,672 ($1088 \times (16 + 3)$) qubits (Each round: 12, 14, 16, key schedule: 3 rounds). This reduces the depth, but it results in an increase in qubit count, which may be considered an abuse.

Another thing is to allocate the ancilla set only once and reuse it throughout the process. To reuse the ancilla set, we perform reverse operations. Initialization with 16 S-boxes† operation (i.e., returning to 0) is performed in parallel for the next round. Thanks to this, we can reuse the initialized ancilla qubits in the next round of SubBytes. However, these reverse operations save qubits, but increase depth. Hence, we adopt a method that maintains the circuit depth by slightly increasing the qubit count.

Figure 6 shows our proposed architecture (in case of ARIA-128). When i-th round $SL^{\dagger}$ is being executed, $i+1$-th round SL is performed. To enable this, two sets of ancilla qubits (a total of $2176 = 1088 \times 2$) must be allocated, and the previous input must be preserved. As demonstrated in the circuits proposed in Section 3.1, the result qubits are allocated separately (i.e., out-of-place), allowing the original input values to be retained.

By allocating two sets of ancilla qubits, the i-th round $SL^{\dagger}$ and the $i+1$-th round SL can be processed in parallel without increasing the circuit depth. To summarize, we adopt an approach that reuses two sets of ancilla qubits alternately. This method minimizes the increase in the total number of qubits

while avoiding the need to allocate qubits for each round. Additionally, the circuit depth for reverse operations remains unaffected because of parallelization.

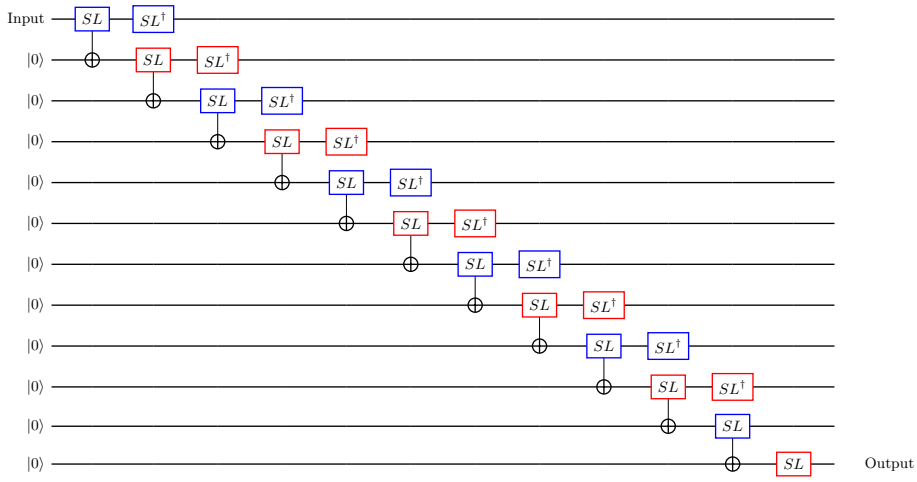


Fig. 6: Proposed parallel Compression function architecture

3.4 Implementation of Diffusion Layer

The diffusion layer can be represented as a 16×16 matrix, as shown in Equation 9. For linear operations such as matrix multiplication, various methods can be adopted to implement. In-place operations, like PLU [14] and XZLBZ [15], do not require additional qubits. While in-place operations reduce the number of qubits needed for the quantum circuit, they require sequential operations of CNOT gates, which increases circuit depth due to the limited computational space from fewer qubits.

In contrast, [12] applied an out-of-place approach, where result qubits are allocated separately, thereby reducing the depth. Their paper demonstrated a significant reduction in depth with this method. For depth optimization, we also assign 128 ancilla qubits to store the results in each round, similar to [12]. More specifically, we assign qubits for the y_i values as shown in Equation 9. To maximize parallel processing in this process, we minimize the depth by rearranging the order of CNOT gates, changing Equation 9 to Equation 13. In other words, the columns in Equation 13 are processed in parallel. Due to this parallel computation, the depth of our diffusion layer implementation is 7.

$$
\begin{aligned}
y_0 &= x_3 \oplus x_4 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{13} \oplus x_{14}, & y_8 &= x_{15} \oplus x_{13} \oplus x_{10} \oplus x_0 \oplus x_1 \oplus x_4 \oplus x_7,\\
y_1 &= x_2 \oplus x_5 \oplus x_7 \oplus x_9 \oplus x_8 \oplus x_{12} \oplus x_{15}, & y_9 &= x_{14} \oplus x_{12} \oplus x_{11} \oplus x_1 \oplus x_0 \oplus x_5 \oplus x_6,\\
y_2 &= x_1 \oplus x_6 \oplus x_4 \oplus x_{10} \oplus x_{11} \oplus x_{15} \oplus x_{12}, & y_{10} &= x_{13} \oplus x_{15} \oplus x_8 \oplus x_2 \oplus x_3 \oplus x_6 \oplus x_5,\\
y_3 &= x_0 \oplus x_7 \oplus x_5 \oplus x_{11} \oplus x_{10} \oplus x_{14} \oplus x_{13}, & y_{11} &= x_{12} \oplus x_{14} \oplus x_9 \oplus x_3 \oplus x_2 \oplus x_7 \oplus x_4,\\
y_4 &= x_5 \oplus x_0 \oplus x_2 \oplus x_{14} \oplus x_{15} \oplus x_8 \oplus x_{11}, & y_{12} &= x_{11} \oplus x_9 \oplus x_{12} \oplus x_6 \oplus x_7 \oplus x_1 \oplus x_2,\\
y_5 &= x_4 \oplus x_1 \oplus x_3 \oplus x_{15} \oplus x_{14} \oplus x_9 \oplus x_{10}, & y_{13} &= x_{10} \oplus x_8 \oplus x_{13} \oplus x_7 \oplus x_6 \oplus x_0 \oplus x_3,\\
y_6 &= x_7 \oplus x_2 \oplus x_0 \oplus x_{12} \oplus x_{13} \oplus x_{10} \oplus x_9, & y_{14} &= x_9 \oplus x_{11} \oplus x_{14} \oplus x_4 \oplus x_5 \oplus x_3 \oplus x_0,\\
y_7 &= x_6 \oplus x_3 \oplus x_1 \oplus x_{13} \oplus x_{12} \oplus x_{11} \oplus x_8, & y_{15} &= x_8 \oplus x_{10} \oplus x_{15} \oplus x_5 \oplus x_4 \oplus x_2 \oplus x_1,
\end{aligned}
\tag{13}
$$

4 Performance & Evaluation

We estimate the quantum resources and the costs of Grover's key search attack on our proposed ARIA quantum circuit, comparing it to previous works. Additionally, we report the trade-off metrics for depth and qubit count. The qubit count, full depth, Toffoli depth, and T-depth are represented as M, FD, TD, and Td, respectively. Furthermore, we estimate the modified trade-off metrics for Grover's parallelization, denoted as FD^2-M, TD^2-M, and Td^2-M, to assess the quantum circuits.

For the implementation, we chose ProjectQ as the quantum programming framework. We specifically verified the implementation using the ClassicalSimulator library and assessed the quantum resources with the ResourceCounter.

The estimated quantum resources are presented in Tables 3 and 4. Table 4 showcases the decomposed Toffoli gates, categorized by the Clifford + T level, as explained in Section 2.1. As indicated in these tables, our implementations require fewer qubits compared to those reported in [16] and [12]. Additionally, the proposed circuits offer the lowest depth and trade-off metrics when compared to all other designs.

Table 3: Required quantum resources for ARIA quantum circuit implementation

Cipher	Source	#X	#CNOT	#Toffoli	Toffoli depth (TD)	#Qubit (M)	Depth	TD-M	TD^2-M
ARIA-128	[3]	1,595	231,124	157,696	4,312	1,560	9,260	$1.60 \cdot 2^{22}$	$1.69 \cdot 2^{34}$
	[16]	1,408	285,784	25,920	60	29,216	3,500	$1.67 \cdot 2^{20}$	$1.96 \cdot 2^{29}$
	[12]	1,408	173,652	17,040	60	26,864	2,187	$1.54 \cdot 2^{20}$	$1.44 \cdot 2^{26}$
	Ours	1,408	97,748	14,816	60	**9,536**	**764**	$\mathbf{1.09 \cdot 2^{19}}$	$\mathbf{1.02 \cdot 2^{25}}$
ARIA-192	[3]	1,851	273,264	183,368	5,096	1,560	10,948	$1.90 \cdot 2^{22}$	$1.18 \cdot 2^{35}$
	[16]	1,624	324,136	29,376	68	32,928	3,978	$1.07 \cdot 2^{21}$	$1.13 \cdot 2^{27}$
	[12]	1,624	197,036	19,312	68	30,320	2,480	$1.97 \cdot 2^{20}$	$1.04 \cdot 2^{27}$
	Ours	1,624	111,596	16,928	68	**10,432**	**867**	$\mathbf{1.35 \cdot 2^{19}}$	$\mathbf{1.44 \cdot 2^{25}}$
ARIA-256	[3]	2,171	325,352	222,208	6,076	1,688	13,054	$1.22 \cdot 2^{23}$	$1.81 \cdot 2^{35}$
	[16]	1,856	362,488	32,832	76	36,640	4,455	$1.33 \cdot 2^{21}$	$1.58 \cdot 2^{27}$
	[12]	1,856	220,420	21,584	76	33,776	2,772	$1.22 \cdot 2^{21}$	$1.45 \cdot 2^{27}$
	Ours	1,856	125,444	19,040	76	**11,328**	**969**	$\mathbf{1.64 \cdot 2^{19}}$	$\mathbf{1.95 \cdot 2^{25}}$

Table 4: Required decomposed quantum resources for ARIA quantum circuit implementation

Cipher	Source	#Clifford	#T	T-depth (Td)	#Qubit (M)	Full depth (FD)	Td-M	FD-M	Td^2-M	FD^2-M
ARIA-128	[3]	1,494,287	1,103,872	17,248	1,560	37,882	$1.60 \cdot 2^{24}$	$1.76 \cdot 2^{25}$	$1.69 \cdot 2^{38}$	$1.02 \cdot 2^{41}$
	[16]	494,552	181,440	240	29,216	4,650	$1.67 \cdot 2^{22}$	$1.01 \cdot 2^{27}$	$1.18 \cdot 2^{26}$	$1.15 \cdot 2^{39}$
	[12]	311,380	119,280	240	26,864	2,952	$1.35 \cdot 2^{19}$	$1.76 \cdot 2^{25}$	$1.44 \cdot 2^{30}$	$1.70 \cdot 2^{37}$
	Ours	205,620	103,712	240	**9,536**	**1,151**	$\mathbf{1.09 \cdot 2^{21}}$	$\mathbf{1.31 \cdot 2^{23}}$	$\mathbf{1.02 \cdot 2^{29}}$	$\mathbf{1.47 \cdot 2^{33}}$
ARIA-192	[3]	1,742,059	1,283,576	20,376	1,560	44,774	$1.89 \cdot 2^{24}$	$1.04 \cdot 2^{26}$	$1.18 \cdot 2^{39}$	$1.42 \cdot 2^{41}$
	[16]	560,768	205,632	272	32,928	5,285	$1.07 \cdot 2^{23}$	$1.30 \cdot 2^{27}$	$1.13 \cdot 2^{31}$	$1.67 \cdot 2^{39}$
	[12]	353,156	135,184	272	30,320	3,347	$1.97 \cdot 2^{22}$	$1.51 \cdot 2^{26}$	$1.04 \cdot 2^{31}$	$1.24 \cdot 2^{38}$
	Ours	234,724	118,496	272	**10,432**	**1,306**	$\mathbf{1.35 \cdot 2^{21}}$	$\mathbf{1.62 \cdot 2^{23}}$	$\mathbf{1.44 \cdot 2^{29}}$	$\mathbf{1.04 \cdot 2^{34}}$
ARIA-256	[3]	2,105,187	1,555,456	24,304	1,688	51,666	$1.22 \cdot 2^{25}$	$1.30 \cdot 2^{26}$	$1.81 \cdot 2^{39}$	$1.02 \cdot 2^{42}$
	[16]	627,000	229,824	304	36,640	5,919	$1.33 \cdot 2^{23}$	$1.62 \cdot 2^{27}$	$1.58 \cdot 2^{31}$	$1.17 \cdot 2^{40}$
	[12]	394,948	151,088	304	33,776	3,741	$1.22 \cdot 2^{23}$	$1.88 \cdot 2^{26}$	$1.45 \cdot 2^{31}$	$1.72 \cdot 2^{38}$
	Ours	263,844	133,280	304	**11,328**	**1,460**	$\mathbf{1.64 \cdot 2^{21}}$	$\mathbf{1.97 \cdot 2^{23}}$	$\mathbf{1.95 \cdot 2^{29}}$	$\mathbf{1.41 \cdot 2^{34}}$

Using the estimated resources for the ARIA quantum circuit, we can assess the cost of Grover's key search attack. The full details of the Grover algorithm are provided in Section 2.2. Within the oracle, two quantum circuits are used: one for encryption and the other to reverse the encryption, returning the state to its pre-encryption. As a result, the total cost of the oracle is effectively twice the cost of implementing the encryption quantum circuit, excluding the qubit cost. When it comes to the diffusion operator, its contribution to the overall quantum resource cost is negligible compared to the oracle. This is because the most of the quantum resources are utilized in implementing the encryption circuit.

Additionally, the cost of Grover's attack can be calculated by r $\times$ 2 $\times$ $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$. In this context, r represents $\frac{key\ size}{block\ size}$, as indicated by [7], which explains that a minimum of r plaintext-ciphertext pairs are required to identify a unique key. As a result, the cost of Grover's attack on ARIA is approximately Table 4 $\times$ r $\times$ 2 $\times$ $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$.

Table 5: Cost of the Grover's key search for ARIA

Cipher	Source	#Gate (G)	Full depth (FD)	T-depth (Td)	#Qubit (M)	G-FD	FD-M	Td-M	FD^2-M	Td^2-M
ARIA-128	[3]	$1.99 \cdot 2^{85}$	$1.82 \cdot 2^{79}$	$1.65 \cdot 2^{78}$	1,561	$1.81 \cdot 2^{165}$	$1.38 \cdot 2^{90}$	$1.25 \cdot 2^{89}$	$1.26 \cdot 2^{170}$	$1.04 \cdot 2^{168}$
	[16]	$1.12 \cdot 2^{84}$	$1.78 \cdot 2^{76}$	$1.47 \cdot 2^{72}$	29,217	$1.99 \cdot 2^{160}$	$1.58 \cdot 2^{91}$	$1.31 \cdot 2^{87}$	$1.41 \cdot 2^{168}$	$1.92 \cdot 2^{159}$
	[12]	$1.30 \cdot 2^{83}$	$1.13 \cdot 2^{76}$	$1.47 \cdot 2^{72}$	26,865	$1.47 \cdot 2^{159}$	$1.85 \cdot 2^{90}$	$1.21 \cdot 2^{87}$	$1.05 \cdot 2^{167}$	$1.77 \cdot 2^{159}$
	Ours	$\mathbf{1.87 \cdot 2^{82}}$	$\mathbf{1.77 \cdot 2^{74}}$	$1.47 \cdot 2^{72}$	**9,537**	$\mathbf{1.65 \cdot 2^{157}}$	$\mathbf{1.03 \cdot 2^{88}}$	$\mathbf{1.70 \cdot 2^{85}}$	$\mathbf{1.82 \cdot 2^{162}}$	$\mathbf{1.25 \cdot 2^{158}}$
ARIA-192	[3]	$1.15 \cdot 2^{119}$	$1.07 \cdot 2^{112}$	$1.95 \cdot 2^{110}$	3,121	$1.23 \cdot 2^{231}$	$1.63 \cdot 2^{123}$	$1.48 \cdot 2^{122}$	$1.74 \cdot 2^{235}$	$1.45 \cdot 2^{233}$
	[16]	$1.20 \cdot 2^{117}$	$1.01 \cdot 2^{109}$	$1.67 \cdot 2^{104}$	65,857	$1.22 \cdot 2^{226}$	$1.01 \cdot 2^{125}$	$1.67 \cdot 2^{120}$	$1.02 \cdot 2^{234}$	$1.39 \cdot 2^{225}$
	[12]	$1.47 \cdot 2^{116}$	$1.28 \cdot 2^{108}$	$1.67 \cdot 2^{104}$	60,449	$1.89 \cdot 2^{224}$	$1.18 \cdot 2^{124}$	$1.54 \cdot 2^{120}$	$1.51 \cdot 2^{232}$	$1.28 \cdot 2^{225}$
	Ours	$\mathbf{1.06 \cdot 2^{116}}$	$\mathbf{1.00 \cdot 2^{107}}$	$1.67 \cdot 2^{104}$	**20,865**	$\mathbf{1.07 \cdot 2^{223}}$	$\mathbf{1.27 \cdot 2^{121}}$	$\mathbf{1.06 \cdot 2^{119}}$	$\mathbf{1.27 \cdot 2^{228}}$	$\mathbf{1.77 \cdot 2^{223}}$
ARIA-256	[3]	$1.38 \cdot 2^{151}$	$1.24 \cdot 2^{144}$	$1.17 \cdot 2^{143}$	3,377	$1.71 \cdot 2^{295}$	$1.02 \cdot 2^{156}$	$1.93 \cdot 2^{154}$	$1.27 \cdot 2^{300}$	$1.13 \cdot 2^{298}$
	[16]	$1.34 \cdot 2^{149}$	$1.14 \cdot 2^{141}$	$1.86 \cdot 2^{136}$	72,081	$1.52 \cdot 2^{290}$	$1.25 \cdot 2^{157}$	$1.02 \cdot 2^{153}$	$1.43 \cdot 2^{298}$	$1.90 \cdot 2^{289}$
	[12]	$1.64 \cdot 2^{148}$	$1.44 \cdot 2^{140}$	$1.86 \cdot 2^{136}$	67,553	$1.18 \cdot 2^{289}$	$1.48 \cdot 2^{156}$	$1.92 \cdot 2^{152}$	$1.07 \cdot 2^{297}$	$1.78 \cdot 2^{289}$
	Ours	$\mathbf{1.20 \cdot 2^{148}}$	$\mathbf{1.12 \cdot 2^{139}}$	$1.86 \cdot 2^{136}$	**22,657**	$\mathbf{1.34 \cdot 2^{287}}$	$\mathbf{1.55 \cdot 2^{153}}$	$\mathbf{1.28 \cdot 2^{151}}$	$\mathbf{1.73 \cdot 2^{292}}$	$\mathbf{1.19 \cdot 2^{288}}$

5 Conclusion

This paper focused on optimizing the quantum circuit for ARIA and estimating the cost of Grover's attack on the algorithm. We implemented all ARIA S-boxes using the Boyar-Perlata method, which provides an efficient approach to S-box realization. By leveraging parallel processing, we minimized the circuit depth and further reduced the qubit count through the use of reversible computation techniques.

Using the implemented circuit, we estimated the quantum resources required for ARIA. Our implementation achieves the optimal performance in these metrics. Specifically, our approach improves upon the most optimized implementation described in [12], achieving reductions of over 61% in overall depth and more than 65.5% in qubit count. Furthermore, we analyzed the computational costs for executing Grover's search attack on ARIA. The Grover attack costs for ARIA-128, ARIA-192, and ARIA-256 are estimated to be $1.65 \cdot 2^{157}$, $1.07 \cdot 2^{223}$, and $1.34 \cdot 2^{287}$, respectively. The results indicate that ARIA-128, ARIA-192, and ARIA-256 correspond to NIST-defined quantum security levels 1, 3, and 5, respectively.

6 Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT).(No. RS-2023-00277994, Quantum Circuit Depth Optimization for ARIA, SEED, LEA, HIGHT, and LSH of KCMVP Domestic Cryptographic Algorithms, 90%) and This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (<Q|Crypton>, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity, 10%).

References

1. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Festa, P. (ed.) Experimental Algorithms. pp. 178–189. Springer Berlin Heidelberg, Berlin, Heidelberg (2010) 8
2. Boyar, J., Peralta, R.: A depth-16 circuit for the AES S-box. Cryptology ePrint Archive, Report 2011/332 (2011), https://eprint.iacr.org/2011/332 8
3. 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) 8, 10, 13, 14
4. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: Quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography. pp. 29–43. Springer International Publishing, Cham (2016) 4
5. Grover, L.K.: 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) 1

6. Jang, K., Baksi, A., Kim, H., Song, G., Seo, H., Chattopadhyay, A.: Quantum analysis of AES. Cryptology ePrint Archive, Paper 2022/683 (2022), https://eprint.iacr.org/2022/683, https://eprint.iacr.org/2022/683 10
7. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing Grover Oracles for quantum key search on AES and LowMC. In: Canteaut, A., Ishai, Y. (eds.) 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. Lecture Notes in Computer Science, vol. 12106, pp. 280–310. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_10, https://doi.org/10.1007/978-3-030-45724-2_10 4, 10, 14
8. 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: International conference on information security and cryptology. pp. 432–445. Springer (2003) 5
9. Liu, Q., Preneel, B., Zhao, Z., Wang, M.: Improved quantum circuits for AES: Reducing the depth and the number of qubits. Cryptology ePrint Archive, Paper 2023/1417 (2023), https://eprint.iacr.org/2023/1417, https://eprint.iacr.org/2023/1417 10
10. 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 4
11. 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 4
12. Oh, Y., Jang, K., Yang, Y., Seo, H.: Quantum implementation and analysis of aria. In: 2024 Silicon Valley Cybersecurity Conference (SVCC). pp. 1–7. IEEE (2024) 8, 10, 11, 12, 13, 14, 15
13. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. IEEE (1994) 1
14. 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) 12
15. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Trans. Symmetric Cryptol. **2020**(2), 120–145 (2020). https://doi.org/10.13154/tosc.v2020.i2.120-145, https://doi.org/10.13154/tosc.v2020.i2.120-145 12
16. Yang, Y., Jang, K., Oh, Y., Seo, H.: Depth-optimized quantum implementation of aria. Cryptology ePrint Archive (2023) 8, 10, 11, 13, 14
17. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of AES with fewer qubits. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 697–726. Springer International Publishing, Cham (2020) 10