Quantum Resource Estimates of Grover's Key Search on ARIA

Amit Kumar Chauhan(✉) and Somitra Kumar Sanadhya

Indian Institute of Technology Ropar, Rupnagar, India
amit.iitropar@gmail.com, somitra@iitrpr.ac.in

Abstract. Grover's algorithm provides a quantum attack against block ciphers by searching for a k-bit key using $O(\sqrt{2^k})$ calls to the cipher, when given a small number of plaintext-ciphertext pairs. Recent works by Grassl et al. in PQCrypto'16 and Almazrooie et al. in QIP'18 have estimated the cost of this attack against AES by analyzing the quantum circuits of the cipher.

We present a quantum reversible circuit of ARIA, a Korean standardized block cipher that is widely deployed in government-to-public services. Firstly, we design quantum circuits for the main components of ARIA, and then combine them to construct the complete circuit of ARIA. We implement Grover's algorithm-based exhaustive key-search attack on ARIA. For all three variants of ARIA-{128, 192, 256}, we establish precise bounds for the number of qubits and the number of Clifford+T gates that are required to implement Grover's algorithm.

We also estimate the G-cost as the total number of gates, and DW-cost as the product of circuit depth and width. To find the circuit depth of various circuits such as squaring, multiplier, and permutation layer, we implement them in an open-source quantum computing platform QISKIT developed by IBM.

Keywords: Quantum cryptanalysis · Quantum circuit · Grover's search algorithm · Block cipher · ARIA

1 Introduction

Recent advancements in quantum computing technologies have enhanced the viability of a large-scale quantum computer. Consequently, much of the traditional public-key cryptosystems such as RSA, ECDSA, ECDH will be completely broken due to Shor's algorithm [22]. However, it is widely believed that symmetric cryptosystems like block ciphers and hash functions are quantum-immune. The only known principle is the square-root speed-up over classical key search or pre-image search attacks with Grover's algorithm [11].

The national institute of standards and technology (NIST) has also initiated a process to standardize the cryptographic primitives that are designed to remain secure in the presence of quantum computers. NIST [19] defined various security categories defined based on the concrete cost of an exhaustive key search on the

L. Batina et al. (Eds.): SPACE 2020, LNCS 12586, pp. 238–258, 2020.
https://doi.org/10.1007/978-3-030-66626-2_13

block cipher AES and collision search for the hash function SHA-3 as a reference point. The relevant cost metrics include the number of qubits, the number of Clifford+T gates, and the T-depth and overall circuit-depth. The NIST proposal derives security categories on gate cost estimates from the gate-level descriptions of the AES oracle by Grassl et al. [10].

Related Work. Grassl et al. [10] studied the quantum circuits of AES and estimated the cost of quantum resources needed to apply Grover's algorithm to the AES oracle for key search. Almazrooie et al. [1] improved the quantum circuit of AES-128. The work by Grassl et al. [10] focused on minimizing the number of qubits. In contrast, Almazrooie et al. [1] focused on reducing the total number of Toffoli gates by saving one multiplication in a binary field inversion circuit. Amy et al. [3] estimated the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. Later, Kim et al. [15] discussed the time-space trade-off cost of quantum resources on block ciphers in general and used AES as an example.

Recently, Langenberg et al. [17] developed the quantum circuits for AES that demonstrate the significant improvements over the works by Grassl et al. [10] and Almazrooie et al. [1]. Their work was based on the different S-box design derived by Boyar and Peralta [6], which significantly reduces the number of Toffoli gates in the S-box and its Toffoli-depth. Jaques et al. [13] further studied the quantum key-search attacks under a depth restriction. As a working example, they implemented the AES Grover oracle in Q# quantum programming language. They provided lower-cost estimates by considering different time-space trade-offs based on the NIST security categories for maximum depth.

Bonnetain et al. [5] also studied the post-quantum security of AES within a new framework for classical and quantum structured search. They used the work by Grassl et al. [10] for deducing concrete gate counts for reduced-round attacks.

Our Contribution. In this work, we present a reversible quantum circuit of the block cipher ARIA-k [16], where $k = 128, 192, 256$ are the different key sizes. To implement the full quantum circuit of ARIA, we separately present the quantum circuits for squaring, multiplier, S-box, and the diffusion layer. For the invertible linear map, we adopt an in-place PLU decomposition algorithm as implemented in SageMath[1] [24]. For each circuit used in ARIA, we establish the cost of quantum resources for the number of qubits and the number of Pauli-X gates, controlled-NOT gates, and Toffoli gates. We also compute their circuit depth by implementing them in an open-source quantum computing platform Qiskit[2] [9]. The source code of Qiskit and Sagemath implementations of squaring, multiplier, S-boxes, and diffusion matrix for ARIA is publicly available[3] under a free license to allow independent verification of our results. For all three variants of ARIA-{128, 192, 256}, we first provide the entire cost of these oracles for the number

[1] https://www.sagemath.org.
[2] https://qiskit.org.
[3] https://github.com/amitcrypto/ARIA-Blocks-Qiskit.git.

of qubits, Pauli-X gates, controlled-NOT gates, Toffoli gates, and Toffoli-depth and overall circuit depth. Later, when we apply Grover's search algorithm [11] to all three ARIA oracles, we consider the decomposition of reversible circuits into a universal fault-tolerant gate set that can be implemented as the Clifford+T gate set.

The Clifford group consists of Hadamard gate, Phase gate, and controlled-NOT gate. An important non-Clifford gate is a Toffoli gate, which is universal and composed of a few T gates and Clifford gates. For implementing the Toffoli gate, we use Amy et al. [2]'s Toffoli decomposition that requires 7 T-gates and 8 Clifford gates with T-depth of 4 and a total depth of 8. However, Selinger [21] offers a Toffoli decomposition with 7 T-gates and 18 Clifford gates, with T-depth 1 and 4 ancillae. To realize an ℓ-fold controlled-NOT gate in terms of T-gates, we use the result by Wiebe and Roetteler [23] and the estimated cost as $(32 \cdot \ell - 84)$.

We then provide the precise cost estimate of quantum resources for Grover's based key search attack in the Clifford+T model. We also compute the G-cost as the total number of gates and DW-cost as the product of circuit depth and width (the number of qubits) as defined by Jaques and Schanck [14]. We provide the results in Table 6. We believe that like the work by Grassl et al. [10], our work will further help to assess the security of ARIA against more advanced quantum reduced-round attacks.

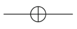
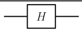
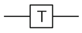
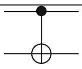
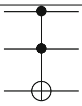
Organization. First, we briefly discuss quantum computation and Grover's algorithm in Sect. 2. Next, we recall the structure of block cipher ARIA in Sect. 3. We rewrite the SubBytes function to get an advantage over the reversible implementation of ARIA. We then separately evaluate the cost of each operation for one round of ARIA, and the cost of round subkeys generation. In Sect. 4, we present the full quantum reversible circuit of ARIA-128, and give overall cost estimates of resources used in the quantum reversible circuit of ARIA-{128, 192, 256}. In Sect. 5, we provide the total resource estimates of an exhaustive key search with Grover's algorithm for ARIA-{128, 192, 256}. In Sect. 6, we compare the quantum resource estimates of ARIA and AES [10,14]. In Sect. 7, we conclude our work.

2 Preliminaries

2.1 Quantum Computation

A quantum computer acts on quantum states by applying quantum gates to its quantum bits (qubits). A qubit ($|0\rangle$ or $|1\rangle$) is a quantum system defined over a finite set $B = \{0, 1\}$. The state of a 2-qubit quantum system $|\psi\rangle$ is the superposition defined as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. In general, the states of an n-qubit quantum system can be described as unit vectors in $\mathbb{C}^{2^n}$ under the orthonormal basis $\{|0\ldots00\rangle, |0\ldots01\rangle, \ldots |1\ldots11\rangle\}$, alternatively written as $\{|i\rangle : 0 \leq i < 2^n\}$. Any quantum algorithm is described by a sequence of gates in the form of a quantum circuit, and all quantum computations are reversible. For circuit design, we use the standard quantum circuit model [18] and adopt the basic gate set {Pauli-X, H, CNOT, T, Toffoli}. We briefly define the basic gates in Table 1. To estimate the cost of resources, we consider the decomposition of reversible circuits over the Clifford+T gate set.

Table 1. Commonly used 1-, 2-, 3-qubit quantum gates, along with their corresponding unitary matrices, circuit symbols, and a description of their actions.

Gate type	**Qubits**	**Circuit symbol**	**Unitary matrix**	**Description**
Pauli-X	1		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Analogous to classical NOT gate, switches $\|0\rangle$ to $\|1\rangle$ and vice-versa.
H	1		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Transforms a basis state into an even superposition of the two basis states.
T	1		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	Adds a relative phase shift of $\pi/4$ between contributing basis states.
CNOT	2		$\begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{bmatrix}$	Controlled-not, a reversible analog to classical XOR gate. The input connected to solid dot is a controlled input to make the operation reversible.
Toffoli	3		$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$	Controlled-controlled-not, a 3-qubit gate that switches the 3rd bit for states where the first two bits are 1 (that is, switches $\|110\rangle$ to $\|111\rangle$ and vice-versa).

2.2 Grover's Search Algorithm

We briefly recall the interface that we need to provide for realizing a key search, namely Grover's algorithm [11]. The Grover searching procedure takes as an input a quantum circuit implementing a Boolean function $f : \{0,1\}^k \to \{0,1\}$ in the usual way, i.e., via a quantum circuit U_f that implements $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$, where $x \in \{0,1\}^k$ and $y \in \{0,1\}$. The basic Grover's algorithm finds an element x_0 such that $f(x_0) = 1$. The Grover's algorithm consists of repeatedly applying the operation G to the initial state $|\psi\rangle \otimes |\phi\rangle$, where $|\psi\rangle = \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |x\rangle$ and $|\phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The Grover operator G is defined as

$$G = U_f \left(\left(H^{\otimes k} \left(2|0\rangle\langle 0| - \mathbf{1}_{2^k} \right) H^{\otimes k} \right) \otimes \mathbf{1}_2 \right)$$

where $|0\rangle$ denotes the all zero basis state of the appropriate size. In order to find a solution x_0 such that $f(x_0) = 1$, G has to be applied $\mathcal{O}(\sqrt{N})$ times to the cipher, where $N = 2^k$ is the total number of possible solutions, and provided that there is only one solution. This means that we can find a solution by applying $H^{\otimes k+1}$ to the initial state $|0\rangle^{\otimes k} \otimes |1\rangle$ and then applying G^{ℓ}, where $\ell = \lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$, followed by a measurement of the entire quantum register which will yield a solution x_0 with high probability [7].

In finding the unique solution using Grover's algorithm, we study the number of gates and space requirements needed to apply Grover's algorithm to the block cipher ARIA.

3 Quantum Circuits to Implement ARIA

ARIA [16] is a 128-bit block cipher standardized as a Korean standard block cipher. Its design is based on a substitution-permutation network such as AES. ARIA supports three different key sizes – 128, 192 and 256 bits with different number of rounds – 12, 14, 16 respectively. The 128-bit internal state and key state are treated as a bytes matrix of 4×4 size, where the bytes are numbered from 0 to 15 column-wise.

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

We will devote 128 qubits to hold the current internal state.

The ARIA cipher consists of two parts:

1. **Round transformation** – Each round of the ARIA cipher consists of the following three basic operations, except the last round.
   - **Substitution layer (SL):** a non-linear byte-wise substitution applied to every byte of the state matrix in parallel, where two different substitution layers exist for odd and even rounds.
   - **Diffusion layer (DL):** a linear matrix multiplication of the state matrix with a 16×16 involution matrix.
   - **Add round key (ARK):** simply XORing of the state and a 128-bit round key, which is derived from the master key.

   Before the first round, an initial ARK operation is applied, and the DL operation is omitted in the last round.

2. **Key scheduling algorithm** – It generates the round keys for different rounds from a master key K of 128, 192, or 256 bits. It works in two phases:
   - **Initialization phase:** Four 128-bit values W_0, W_1, W_2, W_3 are generated from the master key MK, by using a 3-round Feistel cipher.
   - **Round key generation phase:** By combining four values W_0, W_1, W_2, W_3, the round subkeys for encryption are generated. The number of round keys for ARIA-{128, 192, 256} are 13, 15, 17 respectively.

Next, we separately describe each operation of ARIA and measure the quantum resource estimates for its reversible implementation.

3.1 Add Round Key (ARK)

In the implementation of the key expansion, we ensure that the current round key is available on 128 dedicated wires. Then we simply XOR the 128-bit round key with the current state.

Quantum Resource Estimates for ARK. Implementing the bit-wise XOR of the round key needs 128 CNOT gates, which can be executed in parallel, and therefore the circuit depth is 1.

3.2 Substitution Layer (SL)

ARIA has two different substitution layers for even and odd rounds. In each odd round, the substitution layer consist of four 8-bit S-boxes in the following order $(S_1, S_2, S_1^{-1}, S_2^{-1})$, and in each even round the substitution layer consists of the following 4 S-boxes $(S_1^{-1}, S_2^{-1}, S_1, S_2)$ operating on one column. Each S-box replaces one byte of the current state with a new value. We treat a state byte as an element $\alpha \in \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.

1. The first S-box $S_1 : GF(2^8) \rightarrow GF(2^8)$ is defined as

$$S_1(\alpha) := \mathbf{A}.\alpha^{-1} + \mathbf{a} \tag{1}$$

where $\mathbf{A} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{bmatrix}$.

The inverse of S-box S_1 can be defined as

$$S_1^{-1}(\alpha) := (\mathbf{A}^{-1}.(\alpha + \mathbf{a}))^{-1} \tag{2}$$

where $\mathbf{A}^{-1} = \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{bmatrix}$.

2. The second S-box $S_2 : GF(2^8) \rightarrow GF(2^8)$ is defined as

$$S_2(\alpha) := \mathbf{B}.\alpha^{247} + \mathbf{b} \tag{3}$$

where $\mathbf{B} = \begin{bmatrix} 0&1&0&1&1&1&1&0 \\ 0&0&1&0&1&1&1&1 \\ 1&0&0&1&0&1&1&1 \\ 1&1&0&0&1&0&1&1 \\ 1&1&1&1&0&0&1&0 \\ 0&1&1&1&1&0&0&1 \\ 1&0&1&1&1&1&0&0 \\ 0&1&0&1&1&1&1&0 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$.

We rewrite the Eq. (3) as

$$\begin{aligned} S_2(\alpha) &:= \mathbf{B}.(\alpha^{-1})^8 + \mathbf{b} = \mathbf{B}.\mathbf{C}.\alpha^{-1} + \mathbf{b} \\ &= \mathbf{D}.\alpha^{-1} + \mathbf{b} \end{aligned} \tag{4}$$

where $\mathbf{D} = \begin{bmatrix} 0&1&0&1&0&1&1&1 \\ 0&0&1&1&1&1&1&1 \\ 1&1&1&0&1&1&0&1 \\ 1&1&0&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1 \\ 1&1&0&0&1&1&1&0 \\ 0&1&1&0&0&0&1&1 \\ 1&1&1&1&0&1&1&0 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$.

The inverse of S-box S_2 can be defined as

$$S_2^{-1}(\alpha) = (\mathbf{D}^{-1}.(\alpha + \mathbf{b}))^{-1} \tag{5}$$

where $\mathbf{D}^{-1} = \begin{bmatrix} 0&0&0&1&1&0&0&0 \\ 0&0&1&0&0&1&1&0 \\ 0&0&0&0&1&0&1&0 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&0&1&1&0&0 \\ 0&1&1&0&1&0&1&1 \\ 1&0&1&1&1&1&0&1 \\ 1&0&0&1&0&0&1&1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$.

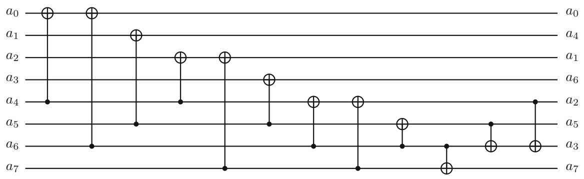
Quantum Resource Estimates for Substitution Layer (SL). In the implementation of substitution layer, we first compute α^{-1} used in SubBytes functions S_1, S_1^{-1} and S_2, S_2^{-1} given in Eqs. (1), (2), (3), (5) and then we apply the corresponding affine transformations.

To compute α^{-1}, we use Itoh-Tsujii multiplier [12] that is a series of multiplication and squaring operations in $\mathsf{GF}(2^n)$. ARIA's operations work in $\mathsf{GF}(2^8)$ with irreducible polynomial $P(x) = x^8 + x^4 + x^3 + x + 1$. Specifically, we can write α^{-1} as

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

The squaring in $\mathsf{GF}(2^8)$ can be implemented with only 12 CNOT gates, and the circuit depth is 7. The resulting circuit is shown in Fig. 1.

The multiplication in $\mathsf{GF}(2^8)$ can be realized by using a classical Mastrovito multiplier, which is adopted by Maslov et al. [8]. For the inputs $\mathbf{a} = [a_0, \ldots, a_7]^\intercal$ and $\mathbf{b} = [b_0, \ldots, b_7]^\intercal$ in $\mathsf{GF}(2^8)$, let the product of $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\mathbf{c} = [c_0, \ldots, c_7]^\intercal$. The multiplier circuit for computing $\mathbf{c}$ can be implemented with 64 Toffoli gates and 21 CNOT gates, and the circuit depth is 37 and Toffoli-depth is 28. The resulting circuit is shown in Fig. 2.



a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
a_0 a_4 a_1 a_6 a_2 a_5 a_3 a_7

Fig. 1. Circuit for squaring in $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.

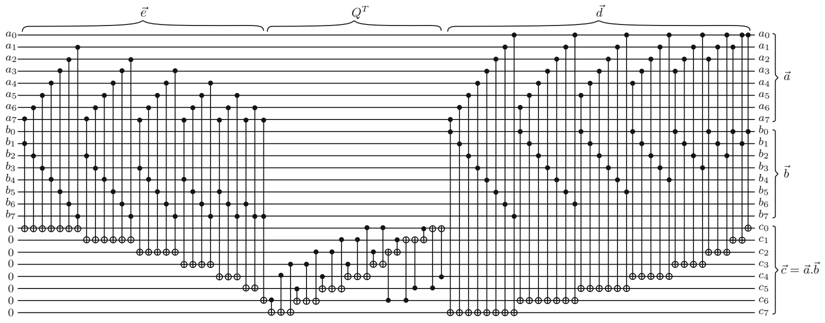


Fig. 2. Circuit for multiplier in $\mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$.

The SubBytes functions S_1 and S_2 given in Eqs. (1) and (3) can be implemented with 33 squarings and 7 multiplications and the required number of qubits are only 40. The resulting circuit is shown in Fig. 3. The same circuit with 40 qubits was also given by Kim et al. [15].

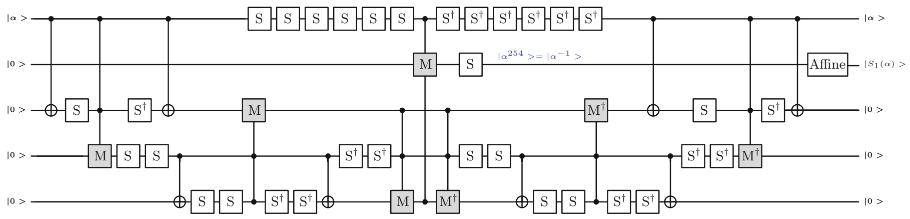


Fig. 3. Circuit for SubBytes functions S_1 and S_2. The gates labelled with S represent the squaring circuit shown in Fig. 1, and the gates labelled with M represent the multiplier circuit shown in Fig. 2. The gates labelled with $S^\dagger$ and $M^\dagger$ represent the inverse squaring and inverse multiplication respectively. Different affine transformations are used for computing S_1 and S_2 (see Fig. 4 and Fig. 6). The first wire represents the input, the second wire represents the S-box output by computing the multiplicative inverse α^{-1} which is equivalent to α^{254} in Galois field $\mathsf{GF}[2^8]$, and other three wires are used as workspace (ancillary qubits).

The affine function of S-box S_1 can be implemented with only 26 CNOT gates and 4 Pauli-X (NOT) gates. The resulting circuit is shown in Fig. 4. The affine function of S-box S_1^{-1} can be implemented with only 18 CNOT gates and 4 Pauli-X (NOT) gates. The resulting circuit is shown in Fig. 5.

The affine function of S-box S_2 can be implemented with only 35 CNOT gates and 4 Pauli-X (NOT) gates. The resulting circuit is shown in Fig. 6. The affine function of S-box S_2^{-1} can be implemented with only 27 CNOT gates and 4 Pauli-X (NOT) gates. The resulting circuit is shown in Fig. 7.

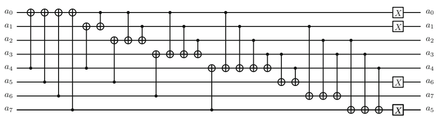


Fig. 4. Circuit for affine function of S_1.

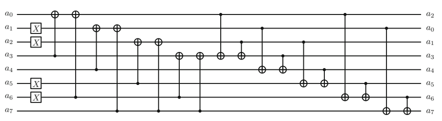


Fig. 5. Circuit for affine function of S_1^{-1}.

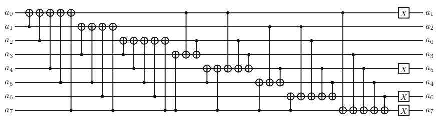


Fig. 6. Circuit for affine function of S_2.

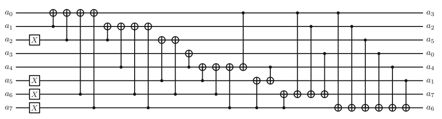


Fig. 7. Circuit for affine function of S_2^{-1}.

3.3 Diffusion Layer (DL)

The diffusion layer is defined by an invertible map $D : GF(2^8)^{16} \rightarrow GF(2^8)^{16}$ which is given by

$$(x_0, x_1, \ldots, x_{15}) \mapsto (y_0, y_1, \ldots, y_{15}), \tag{7}$$

where $(y_0, y_1, \ldots, y_{15})$ can be computed by a matrix multiplication as follows.

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

Quantum Resource Estimates for Diffusion Layer (DL). In the implementation of the diffusion layer, we convert the diffusion matrix of 16×16 size into a matrix of 128×128 size by replacing 0 with a zero matrix of 8×8 size and 1 with an identity matrix of 8×8 size. Using the PLU decomposition algorithm, this modified diffusion matrix can be implemented with only 768 CNOT gates, and the circuit depth is 26.

3.4 Key Scheduling Algorithm

ARIA's key scheduling algorithm consists of two parts: initialization and round key generation, which we describe as follows.

In the initialization phase, four 128-bit values W_0, W_1, W_2, W_3 are generated from the master key MK, using a 3-round 256-bit Feistel cipher. MK can be of 128, 192, 256-bit sizes. The 256-bit value input to 256-bit Feistel is defined as

$$KL||KR = MK||00\ldots0$$

where KL is a 128-bit value with bits from MK, and the remaining bits of MK are the part of 128-bit value KR by appending zeroes to it.

Let F_o and F_e be odd and even round functions of ARIA round transformation. The round function takes 128-bit predefined constants CK_i to be the rational part of π^{-1} and are given as follows:

$$CK_1 = \texttt{0x517cc1b727220a94fe12abe8fa9a6ee0}$$
$$CK_2 = \texttt{0x6db14acc9e21c820ff28b1d5ef5de2b0}$$
$$CK_3 = \texttt{0xdb92371d2126e970324977504e8c90e0}.$$

The generation of four quantum words is as follows:

$$W_0 = KL, \qquad W_1 = F_o(W_0, CK_1) \oplus KR,$$
$$W_2 = F_e(W_1, CK_2) \oplus W_0, \qquad W_3 = F_o(W_2, CK_3) \oplus W_1.$$

The initialization process of key schedule of ARIA is shown below in Fig. 8.

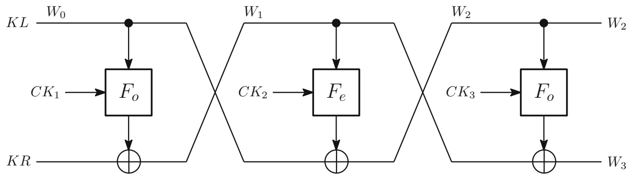


Fig. 8. Initialization phase : 3-round Feistel scheme.

In the round key generation phase, we combine the four values W_0, W_1, W_2, W_3 to obtain the encryption round keys RK_i of 128-bit size each. Note that number of rounds used in ARIA is 12, 14, 16, corresponding to the key sizes 128, 192, 256 of the master key, respectively. Since one extra key is required for the last round key addition, the number of round keys needed is 13, 15, and 17, respectively. The round subkeys are generated as follows:

$$\begin{aligned}
RK_1 &= (W_0) \oplus (W_1^{\ggg 19}), & RK_2 &= (W_1) \oplus (W_2^{\ggg 19}),\\
RK_3 &= (W_2) \oplus (W_3^{\ggg 19}), & RK_4 &= (W_0^{\ggg 19}) \oplus (W_3),\\
RK_5 &= (W_0) \oplus (W_1^{\ggg 31}), & RK_6 &= (W_1) \oplus (W_2^{\ggg 31}),\\
RK_7 &= (W_2) \oplus (W_3^{\ggg 31}), & RK_8 &= (W_0^{\ggg 31}) \oplus (W_3),\\
RK_9 &= (W_0) \oplus (W_1^{\lll 61}), & RK_{10} &= (W_1) \oplus (W_2^{\ggg 61}),\\
RK_{11} &= (W_2) \oplus (W_3^{\lll 61}), & RK_{12} &= (W_0^{\lll 61}) \oplus (W_3),\\
RK_{13} &= (W_0) \oplus (W_1^{\lll 31}), & RK_{14} &= (W_1) \oplus (W_2^{\lll 31}),\\
RK_{15} &= (W_2) \oplus (W_3^{\lll 31}), & RK_{16} &= (W_0^{\lll 31}) \oplus (W_3),\\
RK_{17} &= (W_0) \oplus (W_1^{\lll 19}). & &
\end{aligned} \tag{8}$$

Quantum Resource Estimates for Round Key Generation. In the implementation of key generation, the main cost comes from the initialization part of the key schedule of ARIA in generating the four quantum words W_0, W_1, W_2, W_3 by using a 256-bit Feistel cipher. The Feistel uses different inner round functions for odd and even rounds, which is one round of ARIA, i.e., AddRoundKey, SubBytes, and Diffusion operations.

1. **Add Round Key (ARK):** We simply XOR the 128-bit key with the current state. Thus, only 128 CNOT gates are required to implement this operation.

2. **Substitution Layer (SL):** Four S-boxes $S_1, S_1^{-1}, S_2, S_2^{-1}$ are used in two different substitution layers for odd and even rounds. Each layer uses four times $S_1, S_1^{-1}, S_2, S_2^{-1}$, i.e., 16 S-boxes. We count the required quantum gates to implement each of these S-boxes.

   – Computing S_1 : To find the multiplication inverse of input α, we require 33 squarings and 7 multiplications, including the uncomputing wires (see Fig. 3). Squaring operation requires 12 CNOT gates (see Fig. 1), and multiplication operation requires 64 Toffoli gates plus 21 CNOT gates (see Fig. 2). To apply affine transformation, we require 26 CNOT gates and 4 Pauli-X gates (see Fig. 4). Thus, the total cost is given as:
     - Number of Toffoli gates $= 64 \times 7 = 448$
     - Number of CNOT gates $= 12 \times 33 + 21 \times 7 + 26 = 569$
     - Number of Pauli-X gates $= 4$.

   To compute S-boxes S_1^{-1}, S_2, S_2^{-1}, we use the same circuit to find the inverse as given in Fig. 3. However, the affine functions used in S_1^{-1}, S_2, S_2^{-1} are different from S_1. Therefore, we discuss the cost of affine functions separately while other costs remain same.

- Computing S_1^{-1} : Here, applying affine transformation requires only 18 CNOT gates and 4 Pauli-X gates (see Fig. 5). Thus, the total cost is given as:
    - Number of Toffoli gates $= 64 \times 7 = 448$
    - Number of CNOT gates $= 12 \times 33 + 21 \times 7 + 18 = 561$
    - Number of Pauli-X gates $= 4$.

- Computing S_2 : Here, applying affine transformation requires only 35 CNOT gates and 4 Pauli-X gates (see Fig. 6). Thus, the total cost is given as:
    - Number of Toffoli gates $= 64 \times 7 = 448$
    - Number of CNOT gates $= 12 \times 33 + 21 \times 7 + 35 = 578$
    - Number of Pauli-X gates $= 4$.

- Computing S_2^{-1} : Here, applying affine transformation requires only 27 CNOT gates and 4 Pauli-X gates (see Fig. 7). Thus, the total cost is given as:
    - Number of Toffoli gates $= 64 \times 7 = 448$
    - Number of CNOT gates $= 12 \times 33 + 21 \times 7 + 27 = 570$
    - Number of Pauli-X gates $= 4$.

Therefore, the total number of quantum gates needed to implement the substitution layer are as follows.
- Total number of Toffoli gates $= 448 \times (4 \times 4) = 7,168$
- Total number of CNOT gates $= (569 + 561 + 578 + 570) \times 4 = 9,112$
- Total number of Pauli-X gates $= 4 \times (4 \times 4) = 64$.

3. **Diffusion Layer (DL):** It is a linear operation and implementing it requires only 768 CNOT gates.

Therefore, one round of ARIA requires the following number of gates:

- Total number of Toffoli gates $= 7,168$
- Total number of CNOT gates $= 128 + 9,112 + 768 = 10,008$
- Total number of Pauli-X gates $= 64$.

The round subkeys RK_i are generated using four quantum keywords W_0, W_1, W_2, W_3 as given in the expression (8). It consists of only left or right circular rotation and XOR operations. The circular rotations of words are implemented for free because it is just a particular permutation of the state of the quantum keywords W_0, W_1, W_2, W_3. The only gates required for subkeys generation RK_i are CNOT gates. The number of CNOT gates required for each round subkey is 512 since we are using only one qubit state $|W_4\rangle$ to compute the round subkeys and uncompute the states to save the qubits. For counting of CNOT gates, one can refer to Fig. 9 for the generation of subkey RK_1 working on the state $|W_4\rangle$.

Table 2 demonstrates the estimated cost of generating W_0, W_1, W_2, W_3 in the initialization phase, and the round subkeys RK_i. Some extra Pauli-X gates are also used to generate the round constants CK_1, CK_2, CK_3 of 3-round Feistel.

Table 2. Quantum cost of generating four quantum words and round subkeys for the key schedule of ARIA-{128, 192, 256}.

KeyWords (W_i)	# Pauli-X	# CNOT	# Toffoli
W_0	0	128	0
W_1	64 + 65 = 129	$10,008 + 128 = 10,136$	7,168
W_2	64 + 65 = 129	$10,008 + 128 = 10,136$	7,168
W_3	64 + 57 = 121	$10,008 + 128 = 10,136$	7,168
Total	379	30,536	21,504
Round Subkeys	# Pauli-X	# CNOT	# Toffoli
RK_i for each i	0	$128 \times 4 = 512$	0

4 Resource Estimates: Reversible ARIA Implementation

In this section, we provide the quantum resource estimates to implement the ARIA-{128, 192, 256} for the number of qubits and the number of Pauli-X gates, CNOT gates, and Toffoli gates. We present the full quantum reversible circuit of ARIA-128, as shown in Fig. 9. Note that arranging 12 encryption rounds of ARIA-128 in pipeline fashion requires $12 \times 128 = 1536$ qubits. In order to save qubits, we, therefore, rearrange the encryption rounds in a "zigzag" fashion as adopted by Grassl et al. [10] and Almazrooie et al. [1], to design the reversible circuit of ARIA-128, which requires only 640 qubits of storage. Using the "zigzag" strategy of designing the reversible circuit, we are able to reduce the width but at the cost of increasing depth of the circuit.

The entire circuit of ARIA-128 in Fig. 9 makes use of 1408 qubits (number of qubits required for both key generation and encryption algorithms) and 24 ancilla qubits, which have been used to calculate the multiplicative inverse in the SubBytes function. These 24 ancilla qubits can be placed in between the subkeys generation and the encryption rounds. Every vertical line represents one quantum word of 128-qubits, and every horizontal line indicates 128 connections of the corresponding gates or subroutines. The words KL and KR are assigned for the master key, and the words CK_1, CK_2, CK_3 are assigned to Feistel round constants, which are used to generate four quantum keywords W_0, W_1, W_2, W_3 that are further used for subkeys generation. We also use the extra workspace W_4 of 128-qubits size for generating round subkeys. Each subkey needs only 4 CNOT and few left or right circular rotations, which are considered to be free to implement. The words W_5 to W_9 are devoted to the encryption rounds of ARIA.

Note that ARIA-128 and ARIA-192 require 640 qubits of storage since we need to reverse the encryption rounds on 640 qubits, and the reversing process

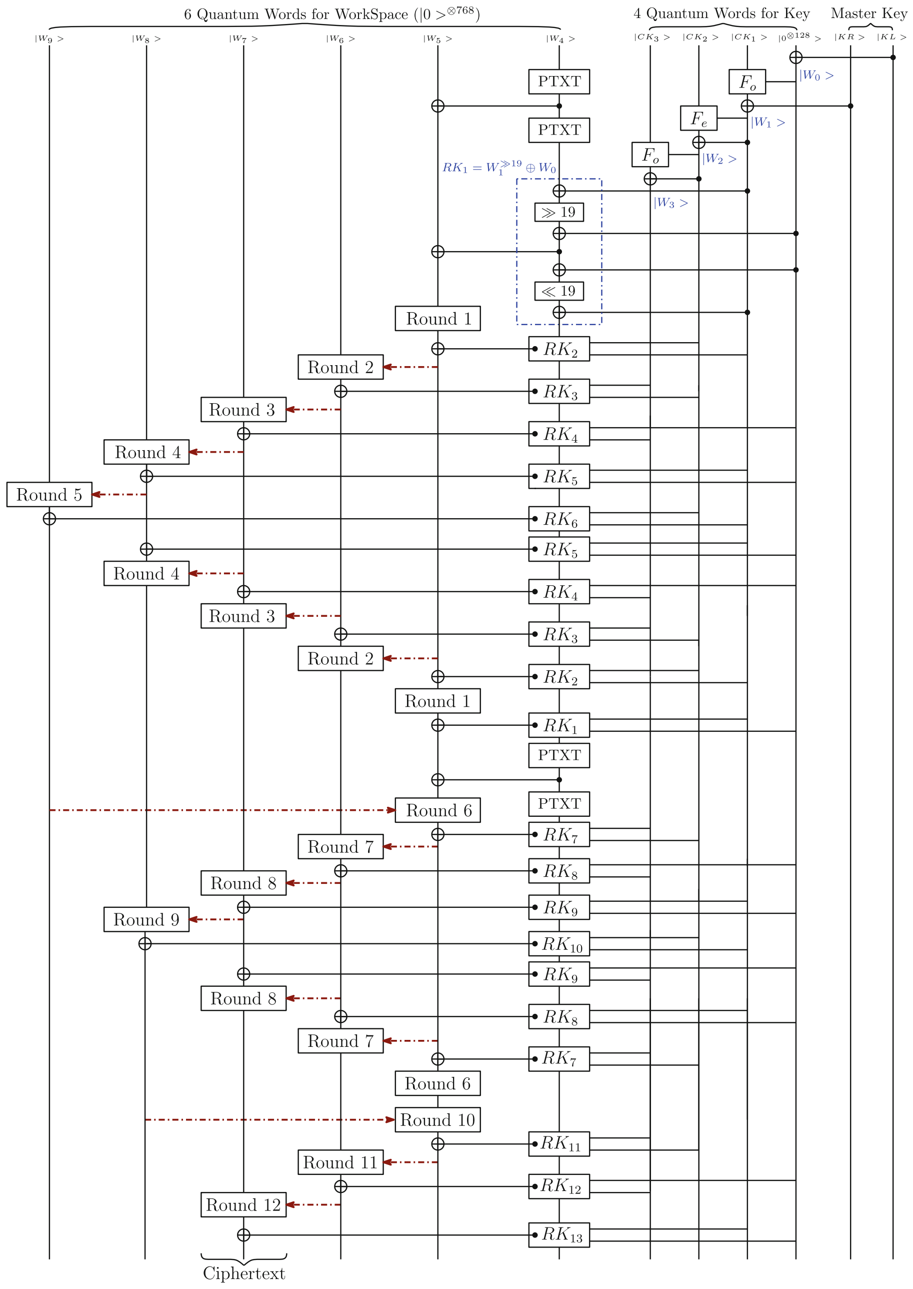

6 Quantum Words for WorkSpace ($|0>^{\otimes 768}$)
4 Quantum Words for Key
Master Key
$|w_9>$ $|w_8>$ $|w_7>$ $|w_6>$ $|w_5>$ $|w_4>$
$|CK_3>$ $|CK_2>$ $|CK_1>$ $|0^{\otimes 128}>$ $|KR>$ $|KL>$
PTXT
F_o F_e
$|W_0>$ $|W_1>$ $|W_2>$ $|W_3>$
$RK_1 = W_1^{\gg 19} \oplus W_0$
$\gg 19$
$\ll 19$
Round 1 … Round 12
RK_1 … RK_{13}
Ciphertext

Fig. 9. Quantum circuit of ARIA-128. Each vertical line (wire) constitutes 128 qubits. The wires $|KL\rangle$ and $|KR\rangle$ represent the master key, next four wires are used for generating keywords $|W_0\rangle, |W_1\rangle, |W_2\rangle, |W_3\rangle$, the wire $|w_4\rangle$ is used as ancillas for round key generation, and the wires from $|w_5\rangle$ to $|w_9\rangle$ are used for main round operations. The dark-red colored dash-lines represent the outputs from one round to another. (Color figure online)

is performed after rounds 5, 9, and 12. However, ARIA-256 requires 768 bits of storage for reversing the rounds, and the reversing process is performed after rounds 6, 11, and 15, requiring only 128 qubits more than ARIA-128.

In Tables 3, 4, 5, we provide the overall cost of quantum resource estimates needed to implement full reversible circuits of ARIA-{128, 192, 256}.

Table 3. Quantum resource estimates for the implementation of ARIA-128.

Phase	#Quantum Gates			Depth		#Qubits	
	#Pauli-X	#CNOT	#Toffoli	Toffoli	Overall	Storage	Ancilla
Initial	0	0	0	0	0	256	0
Key Gen	379	41,228	21,504	588	1,342	512	128
Encryption	1,216	189,896	136,192	3,724	7,918	640	24
Total	1,595	231,124	157,696	4,312	9,260	1,408	152

Table 4. Quantum resource estimates for the implementation of ARIA-192.

Phase	#Quantum Gates			Depth		#Qubits	
	#Pauli-X	#CNOT	#Toffoli	Toffoli	Overall	Storage	Ancilla
Initial	0	0	0	0	0	256	0
Key Gen	379	43,336	21,504	588	1,358	512	128
Encryption	1,472	229,928	164,864	4,508	9,590	640	24
Total	1,851	273,264	183,368	5,096	10,948	1,408	152

Table 5. Quantum resource estimates for the implementation of ARIA-256.

Phase	#Quantum Gates			Depth		#Qubits	
	#Pauli-X	#CNOT	#Toffoli	Toffoli	Overall	Storage	Ancilla
Initial	0	0	0	0	0	256	0
Key Gen	379	45,384	21,504	588	1,374	512	128
Encryption	1,792	279,968	200,704	5,488	11,680	768	24
Total	2,171	325,352	222,208	6,076	13,054	1,536	152

5 Grover Oracle and Key Search Resource Estimates

Assume that we have a quantum adversary who makes the use of Grover's algorithm to find a key on given a small number of plaintext-ciphertext pairs. To apply Grover's algorithm for an exhaustive key search on ARIA, we need a circuit to implement unitary operator U_f based on ARIA. Since ARIA works as a PRF, it is possible that multiple keys leads to the same ciphertext,

$$\mathsf{ARIA}(k_0, m) = \mathsf{ARIA}(k_1, m) = \dots,$$

where $k_0, k_1 \in \{0,1\}^{128}$ are different keys and m is given plaintext. To ensure key uniqueness among other desirable solutions, more than a single pair of plaintext and ciphertext are required. Grassl et al. [10] proposed a method to calculate the required number of pairs (r_k) such that $r_k > \lceil 2k/n \rceil$, where k and n denote the length of the key and plaintext respectively. Thus, to implement ARIA-128, one needs 3 pairs of plaintext-ciphertext, and ($3 \times 1560 = 4680$) qubits. However, following the work by Langenberg et al. [17], we assume that $r_k = \lceil k/n \rceil$ known plaintext-ciphertext pairs are sufficient to avoid false positives in an exhaustive key search for ARIA-k ($k \in \{128, 192, 256\}$). Thus, taking into account "cleaning up" of wires, we need to implement:

- 2 ARIA instances (for $r_{128} = 1$ plaintext-ciphertext pair) for ARIA-128
- 4 ARIA instances (for $r_{192} = 2$ plaintext-ciphertext pairs) for ARIA-192
- 4 ARIA instances (for $r_{256} = 2$ plaintext-ciphertext pairs) for ARIA-256

5.1 Number of Qubits

As adopted in [10,17], to make the smaller T-depth, we can test the multiple plaintext-ciphertext pairs in parallel. The total number of qubits needed is $r_k.q_k + 1$, where q_k is the number of qubits needed to implement ARIA.

- ARIA-128: $1 \cdot 1,560 + 1 = 1,561$ qubits for a Grover-based key search.
- ARIA-192: $2 \cdot 1,560 + 1 = 3,121$ qubits for a Grover-based key search.
- ARIA-256: $2 \cdot 1,688 + 1 = 3,377$ qubits for a Grover-based key search.

5.2 Gate Counts

Operator U_f. Inside the operator U_f, we need to compare the 128-bit outputs of ARIA instances with r_k given ciphertexts. We use a $128 \cdot r_k$-controlled NOT gates. We can also budget $2 \cdot (r_k - 1) \cdot k$ CNOT gates to make the input key available to all r_k parallel ARIA instances (and uncomputing this operation at the end). However, we need to implement the actual ARIA instances. From Tables 3, 4, 5, we obtain the following resource estimates:

- ARIA-128 : Two ARIA instances require $2 \cdot 157,696 = 315,392$ Toffoli gates with a Toffoli depth of $2 \cdot 4,312 = 8,624$. Additionally, we need $2 \cdot 1,595 = 3,190$ Pauli-X gates and $2 \cdot 231,124 = 462,248$ CNOT gates.
- ARIA-192 : Four ARIA instances require $4 \cdot 183,368 = 733,472$ Toffoli gates with a Toffoli depth of $4 \cdot 5,096 = 20,384$. Additionally, we need $4 \cdot 1,851 = 7,404$ Pauli-X gates and $4 \cdot 273,264 = 1,093,056$ CNOT gates.
- ARIA-256 : Four ARIA instances require $4 \cdot 222,208 = 888,832$ Toffoli gates with a Toffoli depth of $4 \cdot 6,076 = 24,304$. Additionally, we need $4 \cdot 2,171 = 8,684$ Pauli-X gates and $4 \cdot 325,352 = 1,301,408$ CNOT gates.

Grover Operator G. Grover's algorithm repeatedly applies the operator

$$G = U_f \left(\left(H^{\otimes k} \left(2 \left|0\right\rangle \left\langle 0\right| - \mathbf{1}_{2^k} \right) H^{\otimes k} \right) \otimes \mathbf{1}_2 \right)$$

where $|0\rangle$ is the all-zero basis state of appropriate size. So in addition to U_f, further gates are needed. Following [10], for the operator $(2\left|0\right\rangle\left\langle 0\right| - \mathbf{1}_{2^k})$, we also budget a k-fold controlled-NOT gate. With $\lfloor \frac{\pi}{4} \cdot \sqrt{2^k} \rfloor$ number of Grover iterations for ARIA-k, we can now give estimates in the Clifford+T model (Fig. 10).

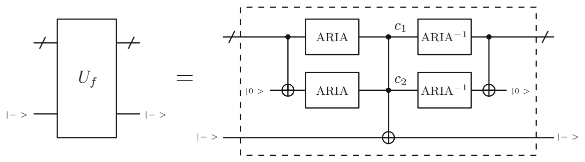


Fig. 10. The reversible implementation of the function U_f for ARIA-128 is shown for which $r = 2$ invocations of ARIA-128 suffice in order to make the target key unique.

5.3 Overall Cost

Here, we provide the overall quantum resource estimates for Grover's based key search in the Clifford+T model. We use the following results/observations:

- By Amy et al. [3], one Toffoli gate requires 7 T-gates and 8 Clifford gates, a T-depth of 4, and a total depth of 8.
- By Wiebe and Roetteler [23], the number of T-gates to realize an ℓ-fold controlled-NOT gate ($\ell \geq 5$) is estimated as $(32 \cdot \ell - 84)$.
- To estimate the total number of Clifford gates, we count only the Clifford gates in the ARIA instances, plus the $2 \cdot (r_k - 1) \cdot k$ CNOT gates inside U_f for the parallel processing of plaintext-ciphertext pairs.
- To estimate the T-depth and overall circuit depth, we take into account only T-depth and circuit depth of ARIA-k. For the S-box used in ARIA, the circuit depth is 391. However, when we represent Toffoli gates in Clifford+T model, the S-box circuit depth is 1,692.

Therefore, the estimated total cost for a Grover-based attack against ARIA-k for $k \in \{128, 192, 256\}$ is as follows.

1. **ARIA**-128:
   - T-gates: $\lfloor \frac{\pi}{4} \cdot 2^{64} \rfloor \cdot (7 \cdot 315,392 + 32 \cdot 128 - 84 + 32 \cdot 128 - 84) \approx 1.65 \cdot 2^{84}$ T-gates with a T-depth of $\lfloor \frac{\pi}{4} \cdot 2^{64} \rfloor \cdot 4 \cdot 8,624 \approx 1.65 \cdot 2^{78}$.
   - Clifford gates: $\lfloor \frac{\pi}{4} \cdot 2^{64} \rfloor \cdot (8 \cdot 315,392 + 3,190 + 462,248) \approx 1.11 \cdot 2^{85}$.
   - Circuit depth: $\lfloor \frac{\pi}{4} \cdot 2^{64} \rfloor \cdot 2 \cdot (22 \cdot 1,692 + 21 \cdot 26 + 112) \approx 1.81 \cdot 2^{79}$.

2. **ARIA**-192:
   - T-gates: $\lfloor \frac{\pi}{4} \cdot 2^{96} \rfloor \cdot (7 \cdot 733,472 + 32 \cdot 192 - 84 + 32 \cdot 192 - 84) \approx 1.92 \cdot 2^{117}$ T-gates with a T-depth of $\lfloor \frac{\pi}{4} \cdot 2^{96} \rfloor \cdot 4 \cdot 20,384 \approx 1.95 \cdot 2^{111}$.
   - Clifford gates: $\lfloor \frac{\pi}{4} \cdot 2^{96} \rfloor \cdot (8 \cdot 733,472 + 7,404 + 1,093,056) \approx 1.30 \cdot 2^{118}$.
   - Circuit depth: $\lfloor \frac{\pi}{4} \cdot 2^{96} \rfloor \cdot 2 \cdot (26 \cdot 1,692 + 25 \cdot 26 + 132) \approx 1.07 \cdot 2^{112}$.

3. **ARIA**-256:
   - T-gates: $\lfloor \frac{\pi}{4} \cdot 2^{128} \rfloor \cdot (7 \cdot 888,832 + 32 \cdot 256 - 84 + 32 \cdot 256 - 84) \approx 1.16 \cdot 2^{150}$ T-gates with a T-depth of $\lfloor \frac{\pi}{4} \cdot 2^{128} \rfloor \cdot 4 \cdot 24,304 \approx 1.16 \cdot 2^{144}$.
   - Clifford gates: $\lfloor \frac{\pi}{4} \cdot 2^{128} \rfloor \cdot (8 \cdot 888,832 + 8,684 + 1,301,408) \approx 1.57 \cdot 2^{150}$.
   - Circuit depth: $\lfloor \frac{\pi}{4} \cdot 2^{128} \rfloor \cdot 2 \cdot (30 \cdot 1,692 + 29 \cdot 26 + 152) \approx 1.23 \cdot 2^{144}$.

6 Cost Comparison of ARIA and AES

For fairness and correctness, we choose the block cipher AES as a reference point because of a well-studied quantum resource estimates for Grover's search on AES. In addition, ARIA and AES share many resemblances like SPN structure, similar S-Box computations defined over the same irreducible polynomial, albeit different mixcolumn operations and key-scheduling algorithm.

In Table 6, we compare the results on S-box computations for the number of qubits, squaring, and multiplication operations to compute the multiplicative inverse. The number of Toffoli gates is directly proportional to the required number of multiplications in a multiplier circuit. For ARIA, one S-box computation requires 7 multiplications (see Fig. 3), which leads to 448 Toffoli gates. We also compare the total number of S-box computations needed in the reversible implementations of AES-128 and ARIA-128.

Table 6. Comparison of S-Box computations for AES and ARIA.

Block ciphers	Single S-box (Multiplicative Inverse)			Total S-box computations
	#Qubits	#Multiplication	#Squaring	
AES-128 (Grassl et al. [10])	40	8	23	320
AES-128 (Almazrooie et al. [1])	48	7	14	320
ARIA-128 (this work)	40	7	33	304

Next, we consider the G-cost and DW-cost (the metrics proposed by Jaques and Schanck [14]) for AES and ARIA. By G-cost, we mean the total number of gates, and by DW-cost, we mean the product of circuit depth and width. Finally, we compare overall cost estimates of quantum resources needed for Grover's algorithm with $\lfloor \frac{\pi}{4} \cdot \sqrt{2^k} \rfloor$ AES and ARIA oracles iterations for exhaustive key search attacks, without a depth restriction. On a lighter note, it is clear from the comparison that the G-cost of ARIA and AES is almost the same, but the DW-cost of ARIA is lower than the AES.

Jaques et al.'s work [13]									
Scheme	r_k	#Clifford	$\#T$	T-depth	full depth	width	G-cost	DW-cost	p_s
AES-128	1	$1.03 \cdot 2^{85}$	$1.59 \cdot 2^{84}$	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	984	$1.83 \cdot 2^{85}$	$1.11 \cdot 2^{91}$	$1/e$
AES-192	2	$1.17 \cdot 2^{118}$	$1.81 \cdot 2^{117}$	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	2224	$1.04 \cdot 2^{119}$	$1.44 \cdot 2^{124}$	1
AES-256	2	$1.46 \cdot 2^{150}$	$1.13 \cdot 2^{150}$	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	2672	$1.30 \cdot 2^{151}$	$1.02 \cdot 2^{157}$	$1/e$

This Work									
Scheme	r_k	#Clifford	$\#T$	T-depth	full depth	width	G-cost	DW-cost	p_s
ARIA-128	1	$1.11 \cdot 2^{85}$	$1.65 \cdot 2^{84}$	$1.65 \cdot 2^{78}$	$1.81 \cdot 2^{79}$	1561	$1.93 \cdot 2^{85}$	$1.37 \cdot 2^{90}$	$1/e$
ARIA-192	2	$1.30 \cdot 2^{118}$	$1.92 \cdot 2^{117}$	$1.95 \cdot 2^{111}$	$1.07 \cdot 2^{112}$	3121	$1.13 \cdot 2^{119}$	$1.63 \cdot 2^{123}$	1
ARIA-256	2	$1.57 \cdot 2^{150}$	$1.16 \cdot 2^{150}$	$1.16 \cdot 2^{144}$	$1.23 \cdot 2^{144}$	3377	$1.36 \cdot 2^{151}$	$1.01 \cdot 2^{156}$	$1/e$

7 Conclusion

We investigated the security of the block cipher ARIA-{128, 192, 256} against Grover's search algorithm. Firstly, we estimated the cost of quantum resources needed to implement different components of ARIA such as S-box computations (multiplicative inverse, squaring, multiplication operations), diffusion layer by analyzing theoretically for the number of qubits and the number of Pauli-X gates, CNOT gates, and Toffoli gates. We also estimated the T-depth and the overall circuit depth. We then evaluated the overall cost of one Round of ARIA and key expansion mechanism. We finally constructed a full quantum circuit of ARIA-128, and provided the cost of quantum resources for full quantum circuits of ARIA-{128, 192, 256}. We also provided the cost of Grover's key search attack against ARIA-{128, 192, 256} in the Clifford+T model. However, the established quantum resource estimates remain far beyond the currently available technology and resources. As a future research work, it might be interesting to reduce Grover's key search attack complexity against ARIA by introducing more optimizations to the circuit, in particular the S-box circuit optimization technique by Boyer and Peralta [6]. It would also be interesting to implement the Grover oracle for ARIA in any quantum programming language for automatic resource estimation like the works [13,20]. Another interesting open problem remains to evaluate the cost of block ciphers like AES, ARIA by implementing them against the multi-target attacks [4].

Acknowledgment. We would like to thank the anonymous reviewers of SPACE 2020 for their insightful comments and suggestions, which has significantly improved the presentation and technical quality of this work. The second author would also like to thank MATRICS grant 2019/1514 by the Science and Engineering Research Board (SERB), Dept. of Science and Technology, Govt. of India for supporting the research carried out in this work. We would also like to thank Dr. Kai-Min Chung for initial discussions on quantum computing.

References

1. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. Quantum Inf. Process. **17**(5), 1–30 (2018). https://doi.org/10.1007/s11128-018-1864-3
2. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. Comput.-Aided Design Integr. Circu. Syst. **32**(6), 818-830 (2013)
3. Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 317–337. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_18
4. Banegas, G., Bernstein, D.J.: Low-communication parallel quantum multi-target preimage search. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 325–335. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72565-9_16

5. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. **2**, 2019 (2019)
6. Boyar, J., Peralta, R.: A small depth-16 circuit for the AES S-Box. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) SEC 2012. IAICT, vol. 376, pp. 287–298. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30436-1_24
7. Boyer, M., Brassard, G., Hoeyer, P., Tapp, A.: Tight bounds on quantum searching (1996). arXiv:quant-ph/9605034
8. 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: Kawano, Y., Mosca, M. (eds.) TQC 2008. LNCS, vol. 5106, pp. 96–104. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89304-2_9
9. Abraham, H., et al.: Qiskit: An open-source framework for quantum computing (2019. https://qiskit.org
10. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) PQCrypto 2016. LNCS, vol. 9606, pp. 29–43. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29360-8_3
11. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: ACM Symposium on the Theory of Computing (1996)
12. Guajardo, J., Paar, C.: Itoh-Tsujii inversion in standard basis and its application in cryptography and codes. Design Codes Cryptogr. **25**(2), 207–216 (2002). https://doi.org/10.1023/A:1013860532636
13. 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.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 280–310. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_10
14. Jaques, S., Schanck, J.M.: Quantum cryptanalysis in the RAM Model: claw-finding attacks on SIKE. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 32–61. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_2
15. Kim, P., Han, D., Jeong, K.C.: Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Inf. Process. **17**(12), 1–39 (2018). https://doi.org/10.1007/s11128-018-2107-3
16. Kwon, D., et al.: New block cipher ARIA. In: Information Security and Cryptology - ICISC (2003)
17. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Trans. Quantum Eng. **1**, 1–12 (2020)
18. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. 10th, Anniversary edn. Cambridge Univ, Press (2011)
19. NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2017). https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf/
20. Ramos-Calderer, S., Bellini, E., Latorre, J.I., Manzano, M., Mateu, V.: Quantum search for scaled hash function preimages. IACR Cryptol. ePrint Arch. 1062 (2020). https://eprint.iacr.org/2020/1062
21. Selinger, P.: Quantum circuits of T-depth one. Phys. Rev. A **87**, 042302 (2013)

22. Shor, P.W.: Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In: Adleman, L.M., Huang, M.-D. (eds.) ANTS 1994. LNCS, vol. 877, pp. 289–289. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58691-1_68
23. Wiebe, N., Roetteler, M.: Quantum arithmetic and numerical analysis using repeat-until-success circuits. Quantum Inf. Comput. **16**(1&2) (2016)
24. William, S., et al.: Sagemath, the Sage Mathematics Software System Version 8.1 (2017). https://www.sagemath.org