applied sciences

MDPI

Article

NTT and Inverse NTT Quantum Circuits in CRYSTALS-Kyber for Post-Quantum Security Evaluation

Gyeongju Song, Kyungbae Jang, Siwoo Eum, Minjoo Sim and Hwajeong Seo *

Division of IT Convergence Engineering, Hansung University, Seoul 02876, Republic of Korea; thdrudwn98@hansung.ac.kr (G.S.); starj1234@hansung.ac.kr (K.J.); 21213203@hansung.ac.kr (S.E.); alswntla@hansung.ac.kr (M.S.)

* Correspondence: hwajeong@hansung.ac.kr; Tel.: +82-760-8033

Abstract: The emergence of quantum computers threatens current cryptographic systems, and NIST is preparing for the post-quantum era through the post-quantum cryptography (PQC) contest. CRYSTALS-Kyber is a lattice-based cipher suite that is used as a PQC standard. Lattice-based cryptography is considered quantum-safe for quantum computing because a quantum algorithm that can more efficiently solve the lattice problem of lattice-based cryptography compared to a classic algorithm has not been reported as yet. In this paper, we present quantum circuits tailored to NTT and inverse NTT, employed for optimized polynomial multiplication within CRYSTALS-Kyber. The proposed quantum circuits operate at $Z_{3329}[X]/(X^{256}+1)$, which are the parameters of CRYSTALS-Kyber. We provide an in-depth description of the NTT/InvNTT quantum circuit's operation and subsequently assess and analyze the quantum resources necessary for these functions. The NTT/InvNTT quantum circuits comprise four unique sub-functions, with the InvNTT additionally incorporating Barrett reduction. To the best of our knowledge, this represents the inaugural implementation of the CRYSTALS-Kyber NTT/InvNTT quantum circuits. We anticipate that our findings will aid in analyzing the security strengths of quantum computers for lattice-based cryptography.

Keywords: quantum circuit; CRYSTALS-Kyber; number theoretic transform (NTT); post-quantum cryptography (PQC)

Citation: Song, G.; Jang, K.; Eum, S.; Sim, M.; Seo, H. NTT and Inverse NTT Quantum Circuits in CRYSTALS-Kyber for Post-Quantum Security Evaluation. *Appl. Sci.* **2023**, *13*, 10373. https://doi.org/10.3390/app131810373

Academic Editors: Christos Bouras, Konstantinos Rantos, Konstantinos Demertzis and George Drosatos

Received: 10 July 2023
Revised: 30 August 2023
Accepted: 14 September 2023
Published: 16 September 2023

1. Introduction

The emergence of high-capacity quantum computers is anticipated to present a substantial threat to the existing cryptosystem. Public key cryptography maintains security based on the difficulty of factoring large integers. However, Shor's algorithm, a quantum algorithm, is known to pose a threat to public key cryptography (e.g., Rivest–Shamir–Adleman (RSA) and elliptic curve cryptography (ECC)) because it is possible to factor large integers within polynomial time [1]. The security level of symmetric key cryptography is contingent upon the length of the employed encryption key. Grover's algorithm can accelerate the attack by performing a brute force attack with $\sqrt{2^n}$ queries against symmetric key cryptography using n-bit keys. In essence, this results in a reduction of security potency from 2^n to $2^{\frac{n}{2}}$ when executed on quantum computers. A potential safeguard against quantum computer attacks involves doubling the length of the encryption key. Nonetheless, due to the disparity in operational methodologies, resource requisites, and feasible computations between classical and quantum computers, the level of security achieved on a classical computing system does not align seamlessly with the quantum security potency achievable on a quantum computer. Hence, to confirm the quantum security of cryptography on quantum computers, it is imperative to optimize and implement the cryptographic operations within quantum circuits. Also, based on the implemented quantum circuit, it is necessary to confirm the required quantum gate and the depth of the quantum circuit. In recent years, research has focused on ascertaining the quantum resistance of cryptography through implementing quantum circuits for symmetric cryptography and hash functions [2–21]. Upon

the alignment of available quantum computer resources with the requisites for executing a cryptographic attack, it is anticipated that the targeted cipher will be broken. Consequently, prevailing cryptographic systems are expected to be supplanted by algorithms in the realm of post-quantum cryptography (PQC). It is judged that the PQC algorithm can have quantum resistance because it is difficult to operate on a quantum computer, but discussions on this are ongoing.

The National Institute of Standards and Technology (NIST) held a competition to select the standard post-quantum cryptography (PQC) algorithm. The PQC algorithm is intended to replace the current cryptosystems in the post-quantum era [22]. In the one round conducted in 2017, 69 candidate algorithms were released. In 2019, 26 out of 69 algorithms advanced to round 2 (17: encryption/key-encapsulation (PKE/KEM), 9: signature schemes). In round 3, 15 candidate algorithms were released (7 finalists, 8 alternatives). In 2022, four finalists were announced. Candidate algorithms included CRYSTALS-Kyber (KEMs/Encryption), CRYSTALS-Dilithium, Falcon, and SPHINCS+ (Signature).

In this paper, we propose a quantum circuit for the number theoretic transform (NTT) and inverse number theoretic transform (InvNTT) used in CRYSTALS-Kyber. The implementation of encryption through quantum circuits is undertaken to leverage the distinctive properties of quantum computers. This approach involves investigating methods of attacking conventional cryptographic systems using quantum computational capabilities, thereby assessing the strength of quantum security. To ascertain the potency of quantum security, it is imperative to implement quantum algorithms on quantum computers and estimate the quantum resources. These efforts anticipate the commercialization of quantum computers, facilitating the development of robust security technologies and algorithms to effectively respond to quantum threats. The Montgomery reduction is used in NTT, and the Montgomery reduction and Barrett reduction are used in InvNTT. The inner function used in the InvNTT operation is similar to NTT but also uses the Barrett reduction. This paper implements all of these functions used in NTT and InvNTT as part of a quantum circuit that can operate on a quantum computer. Various methods were used to construct an efficient quantum circuit, and details are covered in Section 3.

We analyzed the quantum resources used through the implementation of quantum circuits for the internal operations of NTT/InvNTT. CRYSTAL-Kyber [23] is an IND-CCA2-secure KEM, underpinned by the hardness of Module-LWE [24] on a lattice. In the sub-operation of CRYSTALS-Kyber, a polynomial multiplication operation is performed with $n = 256$ and $q = 3329$ parameters for $Z_q[X]/(X^n + 1)$. Regarding this, NTT is employed to execute modular multiplication efficiently for polynomials a and b. The ring N is represented as a product of sub-rings in the NTT operation, and it is divided into polynomials for processing. Each a and b polynomial is expressed with 256 coefficients in the sub-ring using NTT transformation. These transformed polynomials are then multiplied point-wise. Unlike the traditional schoolbook multiplication with a complexity of $O(n^2)$, NTT-based multiplication has a complexity of $O(n \log n)$. Therefore, NTT is a good way to reduce the computational complexity of multiplication operations in general computers. To the best of our knowledge, the quantum circuits that have been proposed are the first NTT/InvNTT quantum circuits tailored to the parameters of CRYSTALS-Kyber. Our quantum circuit implemented NTT/InvNTT multiplication for $Z_q[X]/(X^n + 1)$ to operate at CRYSTALS-Kyber parameters $n = 256$ and $q = 3329$. The NTT operation employs the Montgomery reduction, while the InvNTT operation utilizes both the Montgomery and Barrett reductions. We implemented a negative integer by expressing it using two's complement in qubits, and $n \times 32$ qubits are used to express $Z_q[X]/(X^n + 1)$ coefficients. Since we express the intermediate computation process in two's complement, we do not allocate extra qubits to determine negative and positive integers. However, only 1-bit qubits are used for the intermediate conditionals. Finally, we estimate and analyze the quantum resources required for sub-functions to NTT/InvNTT.

The structure of this paper is as follows: In Section 2, we present research on quantum computing, the Grover algorithm, and CRYSTALS-Kyber. Section 3 describes the implementation of the proposed NTT and InvNTT quantum circuits. Section 4 estimates and analyzes the resources required for the proposed quantum circuits. Finally, Section 5 concludes the paper.

1.1. Contributions

1. First Quantum Circuit Implementation for NTT and inverse NTT: To the best of our knowledge, this is the first implementation of Kyber NTT and InvNTT in quantum circuits. We designed a structure for operating NTT and inverse NTT in quantum circuits.
2. Quantum Circuit for Optimal Utilization of Quantum Resources: We propose NTT and inverse NTT quantum circuits to maximize the utilization of quantum resources. The design of an efficient quantum circuit aims to curtail the consumption of quantum resources, including qubits and quantum gates, throughout the operational process.
3. Offers the Foundation for Quantum Strength Assessment of Kyber: The quantum circuit introduced in this paper serves as a foundational tool for assessing the quantum robustness of the Kyber cryptographic scheme. As we anticipate the emergence of powerful quantum computers in the future, it will become imperative to evaluate post-quantum resistance for post-quantum cryptography (PQC) protocols.

1.2. Extended Version of MobiSec 2022

The work presented in MobiSec 2022 is revisited in this paper. Research on the NTT quantum circuits in CRYSTALS-Kyber was presented in [25]. This time, we further present an efficient inverse NTT quantum circuit. Using this implementation, we conducted research on estimating, comparing, and analyzing all quantum resources required for inverse NTT in CRYSTALS-Kyber.

1.3. Notation

In this paper, we use a common notation for various parameters to describe the operation of *NTT* and *Inv NTT* quantum circuits, as shown in Table 1.

Table 1. Meanings for symbols used in this paper.

Symbols	Meaning
FFM	This means multiplication in a finite field.
Inv FFM	This represents the inverse operation for FFM.
Mont	This refers to the Montgomery reduction function. It is used in NTT and InvNTT.
$NTT_{sub1,sub2}$	These are sub-functions used in NTT. These perform addition or subtraction on two qubits.
Barret	This refers to the Barrett reduction function. It is used in InvNTT.
$\oplus$	Perform XOR operation.
$\circ$	Connect the algorithms from left to right.

2. Background

2.1. Quantum Computing

Quantum computers leverage two fundamental quantum mechanical properties: superposition and entanglement. Superposition allows a qubit to exist simultaneously in multiple states, rather than being confined to a binary state as with classical bits. Entanglement, a uniquely quantum phenomenon, links qubits in such a manner that the state of one instantly influences the state of another, irrespective of distance. When n qubits are in a superposition state, these can represent 2^n values. Operations in all states can be performed at once. A qubit in a quantum superposition can be represented as follows:

$$|\psi\rangle = \alpha|1\rangle + \beta|0\rangle$$

Any measurement on these qubits will always yield one of two eigenvalues, although we cannot predict which one will be observed [26]. α and β are the probability amplitudes, where the probability of a result $|1\rangle$ with a value 1 is $|\alpha|^2$ and the probability of a result $|0\rangle$ with a value 0 is $|\beta|^2$. Normalizing that state guarantees this equation:

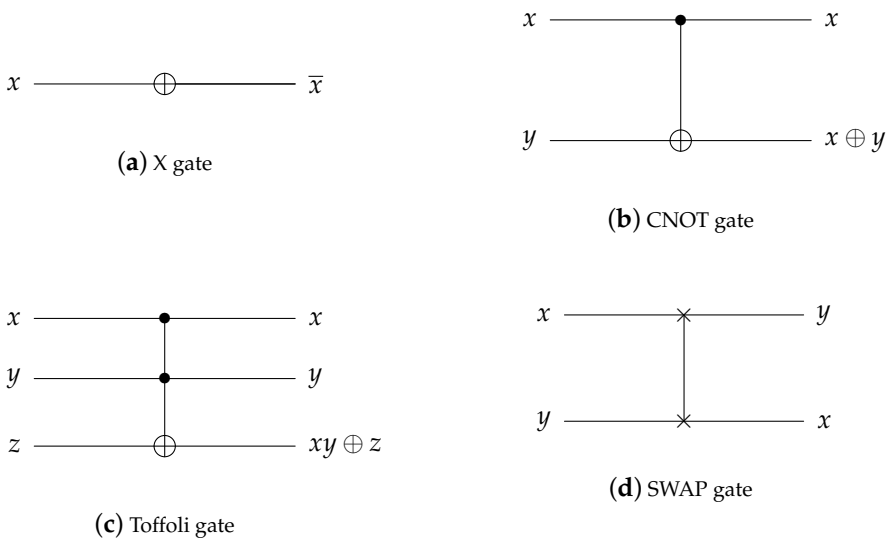
$$|\alpha|^2 + |\beta|^2 = 1$$

In quantum computing, all operations are reversible, barring measurements. This reversibility means that one can recover the original state from the resultant state without extra information, requiring the quantum gates to possess equal numbers of input and output bits. Below are the X, CNOT, Toffoli, and SWAP matrices, which are representative quantum gates utilized for the manipulation of qubits within quantum computing:

$$X = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 1 visually presents the quantum gate operations corresponding to each aforementioned gate. The (a) X(NOT) gate works on a single qubit, flipping its state. The (b) CNOT gate uses two qubits: a control and a target qubit. The target's state is inverted only if the control is set to one. The (c) Toffoli gate employs three qubits: two as controls and one as a target. Only when both control qubits are set to one does the target's state reverse. The (d) SWAP gate interchanges the physical positions of two qubits.



(**a**) X gate

(**b**) CNOT gate

(**c**) Toffoli gate

(**d**) SWAP gate

Figure 1. Quantum logic gates (https://en.wikipedia.org/wiki/Quantum_logic_gate, accessed on 30 August 2023).

2.2. CRYSTALS-Kyber

Quantum computers pose significant threats to classical cryptographic schemes, necessitating a transition to quantum-resistant algorithms. The post-quantum cryptography (PQC) competition was launched to respond to this. Its primary goals involve discovering, evaluating, and standardizing cryptographic methods resistant to quantum threats. CRYSTAL-Kyber is one of the candidate algorithms submitted to PQC. CRYSTAL-Kyber [23] is an IND-CCA2-secure key encapsulation mechanism (KEM) with the hardness of Module-LWE [24] on a lattice. This cryptographic, which is designed to be robust in the post-quantum era, was one of the finalists of the post-quantum cryptography project conducted by NIST. The security foundation is based on the hardness of resolving learning-with-error (LWE) problems for module lattices.

2.3. Number Theoretic Transform (NTT)

Theorem 1. *The Fourier transform (FT) is a mathematical transformation that translates a function from its original domain (often time or space) to a representation in the frequency domain. For instance, an audio signal can be decomposed into multiple sine and cosine waves through the Fourier transform (FT). For an oscillation function, $f(t)$, which changes over time t, the oscillation encompasses various frequency components. If the frequency component is extracted and expressed as the frequency intensity distribution $F(\omega)$ for the frequency ω, the nature of vibration can be easily analyzed. The mathematical expression for FT follows:*

$$F(\omega) = \frac{1}{\sqrt{2\pi}} = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

Theorem 2. *The discrete Fourier transform (DFT) operates on finite N complex fields, in contrast to the continuous interval (−∞,∞) addressed by the Fourier transform (FT). The N complex number sequence $x_N = x_0, \ldots, x_{(n-1)}$ is converted into another complex number sequence $X_k = X_0, \ldots, X_t$. Equation (1) shows the DFT process of a complex number field. k is the index of frequency, n is the index of the time sample, and N is the total number of samples in the input signal.*

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n / N} \tag{1}$$

The number theoretic transform (NTT) generalizes the domain of the discrete Fourier transform (DFT) to operate over integer fields. It uses the n-th primitive root of unity based on a quotient ring instead of the complex field of DFT. When performing multiplication on two n-bit length polynomials, typical schoolbook multiplication has a computational complexity of $O(n^2)$ whereas that of NTT multiplication is $O(nlogn)$. In lattice-based ciphers, NTT is used for efficient multiplication. The NTT multiplication of two polynomials, a and b, is calculated as follows: $NTT^{-1}(NTT(a) \cdot NTT(b))$. For the polynomial transformation, $a \in Z_q[X]/(X^{256}+1)$ is split into $a_L^{(1)} \in Z_q[X]/(X^{256} - \zeta^{128})$ and $a_R^{(1)} \in Z_q[X]/(X^{256} + \zeta^{128})$. The $a_L^{(1)} \in Z_q[X]/(X^{256} - \zeta^{128})$ and $a_R^{(1)} \in Z_q[X]/(X^{256} + \zeta^{128})$ polynomials that are split into sub-rings are transformed, as seen in Equation (2).

$$a \in Z_q[X]/(X^{256}+1) \begin{cases} a_L^{(1)} = (a_0 + \zeta^{128} a_{128}) + (a_1 + \zeta^{128} a_{129}) + \cdots \\ a_R^{(1)} = (a_0 - \zeta^{128} a_{128}) + (a_1 - \zeta^{128} a_{129}) + \cdots \end{cases} \tag{2}$$

2.4. Montgomery Reduction

The Montgomery reduction is a technique used in modular arithmetic to efficiently compute the modulo operation (remainder) of large integers [27]. The Montgomery reduction is used to enhance the multiplication speed within NTT/InvNTT computations. Utilizing the Montgomery reduction in modular multiplication can replace division op-

erations with shift operations. This multiplication converts integers a and b into the Montgomery form to calculate $a \cdot b = \text{mod } N$.

2.5. Barrett Reduction

The Barrett reduction, introduced by P.D. Barrett in 1986 [28], is an optimization in modular arithmetic, which facilitates modular reduction without direct division. This technique, primarily used in cryptographic algorithms, employs multiplication and shifts, making it efficient for large-number arithmetic. When computing instances at $r = k \bmod n$ (n: constant, $k < n^2$) for a fixed modulus n, long division causes slowness. For these calculations, the Barrett reduction performs modular reductions. The Barrett reduction replaces division with multiplication for the $r = k \bmod n$ operation.

3. Proposed Method

This paper introduces quantum circuits designed for NTT/InvNTT operations within a lattice-based algorithm. The formulated NTT/InvNTT quantum circuits are tailored to accommodate CRYSTALS-Kyber parameters $n = 256$ and $q = 3329$, facilitating multiplication within the modulus $Z_q[X]/(X^n + 1)$. Within these quantum circuits, an optimization technique was implemented to reduce the consumption of quantum resources. To perform addition within the quantum circuit, the ripple-carry adder proposed by [29] was appropriately used. Figure 2 illustrates the quantum circuit for the proposed NTT, with the sequence of operations detailed as follows ($\circ$: Connect the algorithms from left to right):

$$\text{NTT} = NTT_{sub} \circ Montgomery\ reduction \circ FFM_{1,2}$$

Figure 3 illustrates the quantum circuit for the proposed InvNTT, with the sequence of operations detailed as follows:

$$\text{InvNTT} = InvFFM_2 \circ Montgomery\ reduction \circ InvFFM_1 \circ Barret\ reduction$$

- *FFM and Inv FFM*: These perform multiplication in a finite field. It outputs 32-qubit results by multiplying the two 16-qubit values. In detail, it is divided into FFM_1, FFM_2, $Inv\ FFM_1$, and $Inv\ FFM_2$ functions. FFM_1 and FFM_2 are used in the NTT quantum circuit, and $Inv\ FFM_1$ and $Inv\ FFM_2$ are used in the InvNTT quantum circuit.
- *Montgomery reduction*: The process executes the Montgomery reduction multiplication on the product of a 32-qubit input and zeta; (input $\times$ *zeta*). This operation is achieved in the quantum circuit via multiplication, subtraction, and shifting, employing values $Q = 3329$ and $QINV = -3327$. The outcome of the Montgomery reduction is a 16-qubit value.
- NTT_{sub}: This function in the NTT quantum circuit manages addition and subtraction between the Montgomery reduction multiplication outcome and the input. Specifically, it branches into NTT_{sub1} and NTT_{sub2}. NTT_{sub1} yields the subtraction result while NTT_{sub2} produces the addition result.
- *Barrett reduction*: The function executes the Barrett reduction multiplication on a 16-bit input. Within the quantum circuit, this operation involves both multiplication and addition. Additionally, this function utilizes an extra 16 qubits.

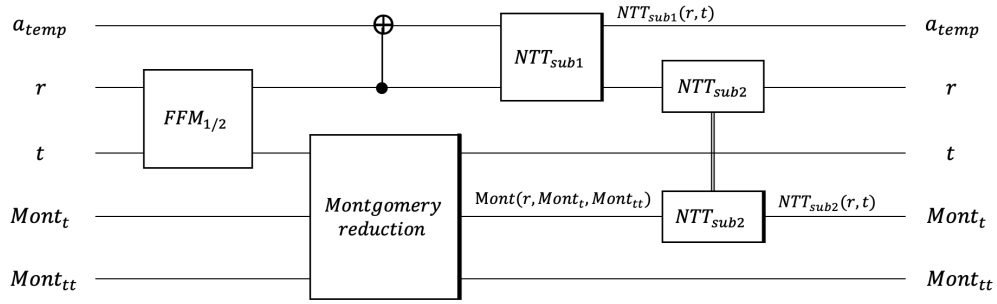


Figure 2. Process for the NTT quantum circuit.

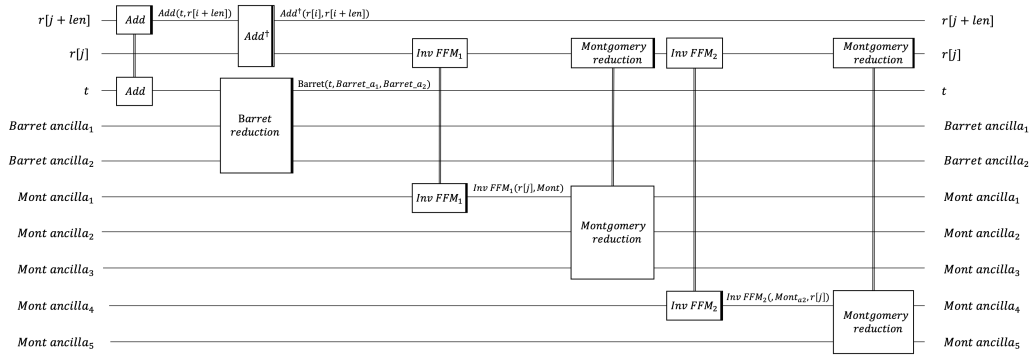


Figure 3. Process for the inverse NTT quantum circuit; † indicates reversible gates.

In qubit representation, negative numbers are denoted using the two's complement method. Since zeta is a pre-calculated value, there is no separate calculation. The NTT quantum circuit is performed using the Montgomery reduction. We allocate 32 qubits for each term to represent the multiplication result as a binary number. In the CRYSTALS-Kyber parameter $n = 256$, $q = 3329$, $32 \times n$-qubits are used to store the coefficients in the $Z_q[X]/(X^n + 1)$. The original input must be used in the last NTT_{sub} function, so the function proceeds while maintaining the input. Therefore, each function uses the temp qubit to store the operation result and proceeds to the next function. Source codes for the NTT and inverse NTT quantum circuits in CRYSTALS-Kyber are available in [30]. A description of each function follows.

3.1. FFM and Inv FFM

FFM and *Inv FFM* are divided into four sub-functions: FFM_1, FFM_2, $Inv\ FFM_1$, $Inv\ FFM_2$. FFM_1 and FFM_2 are used in NTT, $Inv\ FFM_1$ and $Inv\ FFM_2$ are used in InvNTT. FFM_1, FFM_2, and $Inv\ FFM_1$ output 32-bit results by multiplying the input with a constant *zeta* (input $\times$ *zeta*), while $Inv\ FFM_2$ multiplies the input with a constant f (input $\times$ f). In detail, it is divided into FFM_1, FFM_2, $Inv\ FFM_1$, and $Inv\ FFM_2$ functions. In NTT, FFM_1 and FFM_2 are used, and in InvNTT, $Inv\ FFM_1$ and $Inv\ FFM_2$ are used. In this quantum circuit, the ripple-carry adder [29] proposed by Cuccaro et al. was used. The difference between each function is as follows:

The FFM_1 function yields a 32-qubit output by multiplying a 32-qubit input with the constant *zeta*; (input $\times$ *zeta*). Importantly, within this FFM_1 function, the signs of both the input and *zeta* are ascertained, determining the sign of the resultant multiplication output. Similarly, the FFM_2 function generates a 32-qubit output by multiplying a 32-qubit input with the constant *zeta*; (input $\times$ *zeta*).

Within the FFM_2 function, while the sign of *zeta* is known, the sign of the input remains undetermined. Consequently, an additional step is necessary within the FFM_2 quantum circuit to identify whether the sign of the input is the same as or different from that of *zeta*. (If equal, the result is positive, if different, the result is negative). To meet this step, a 1-qubit *check* serves the purpose of determining the sign. The $Inv\ FFM_1$ function yields a 32-qubit result by multiplying a 16-qubit input with the value *zeta*; (input $\times$ *zeta*). Within this $Inv\ FFM_1$ function, the sign of *zeta* is known, while the sign of the input remains undetermined. To determine the output's sign, it is necessary to ascertain the sign of the input. For this, an additional step is introduced to check the input's sign (whether it matches or differs from that of *zeta*). To meet this step, a 1-qubit *check* is used to determine the sign (similar to FFM_2). Since *zetas* are fixed constants, we can reduce the number of qubits used by performing an addition to the input up to the value of *zetas* without assigning a value to the qubit. It is not a qubit-to-qubit multiplication, but an iteration of the qubit-to-qubit addition up to the value of *zeta*. Finally, $Inv\ FFM_2$ outputs a 32-bit result by multiplying a 32-qubit input with the constant f (input $\times$ f). Exceptionally, in this function, the value of f is always fixed as a positive number. Consequently, there is no need to judge its sign. One task involves checking the sign of the input (whether the sign of the input is negative or positive). The $FFM_{1,2}$ functions mean that each function operates in the first NTT cycle and other cycles. In the first NTT cycle ($C = 1$), the sign of the input and *zeta* are known, so the process of determining the sign is not necessary. On the other hand, in $C \geq 2$, since the sign of the input needs to be confirmed, the 1-qubit *check* is used to confirm the sign.

Algorithm 1 illustrates the procedure of the $FFM_{1,2}$ functions within NTT. To preserve the original value of the input, the result of the function is saved in a 32-qubit *ancilla*. The $FFM_{1,2}$ functions use CNOT gates to deposit input values into ancilla and then carry out multiplication. Within both FFM and $Inv\ FFM$, the signs of the input and *zeta* are confirmed. If the signs are the same, the result is a positive integer, and if the sign is different, the result is a negative integer. The outcome of the multiplication between 'input' and '*zeta*' is saved in the *ancilla* qubit. The $Inv\ FFM_{1,2}$ functions pertain to individual operations within InvNTT. $Inv\ FFM_1$ denotes the initial multiplication within a finite field, and $Inv\ FFM_2$ represents the subsequent multiplication within that field in InvNTT.

Algorithm 2 presents the quantum circuits for $Inv\ FFM_1$ and $Inv\ FFM_2$, both functioning within InvNTT. The $Inv\ FFM_{1,2}$ functions use CNOT gates to store input values in *ancilla* and perform multiplication. Specifically, the $Inv\ FFM_1$ function assesses the signs of both the input and zeta. When their signs align, the result is positive, but if they differ, the outcome is negative. The result of (input $\times$ *zeta*) is stored in an ancilla qubit. On the other hand, the $Inv\ FFM_2$ function carries out multiplication using the input and a constant f, which is pre-determined as a positive number. Hence, in this case, only the sign of the input is verified.

Algorithm 1: $FFM_{1,2}$ multiplication for NTT

Data: *zeta* , *r*(32bit), *check*(1-qubit)

[#FFM_1 function in NTT ($C = 1$**)]**

for i=0 **to** $length(r)$: $ancilla[i] \leftarrow$ CNOT(*r*[*i*], ancilla[*i*])
if $zeta \neq 1$ **then**
  if $zeta < 0$ *and* $input < 0$ **then**
    for i=0 **to** $-zeta + 1$: **Dagger**: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
  if $zeta < 0$ *and* $input > 0$ **then**
    for i=0 **to** $-zeta - 1$: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
  if $zeta \geq 0$ *and* $input > 0$ **then**
    for i=0 **to** $zeta - 1$: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
  if $zeta \geq 0$ *and* $input < 0$ **then**
    for i=0 **to** $zeta + 1$: **Dagger**: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
end

return *ancilla*

[#FFM_2 function in NTT ($C \geq 2$**)]**

check $\leftarrow$ CNOT(*r*[$length(r)$-1], *check*)
if $zeta \neq 1$ **then**
  if $zeta \geq 0$ **then**
    X(*check*)
    if *check=1* **then**
      for i=0 **to** $-zeta + 1$: **Dagger**: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
    end
    X(*check*)
    if *check=1* **then**
      for i=0 **to** $-zeta - 1$: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
    end
  end
else
  X(*check*)
  if *check=1* **then**
    for i=0 **to** $-zeta - 1$: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
  X(*check*)
  if *check=1* **then**
    for i=0 **to** $zeta + 1$: **Dagger**: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
end

return *ancilla*

Algorithm 2: $Inv\ FFM_{1,2}$ multiplication for InvNTT

Data: *zeta* , *r*(16bit), *check*(1-qubit)

[#$Inv\ FFM_1$ function in InvNTT]

$check \leftarrow$ CNOT($r[length(r)-1]$, *check*)
if $zeta \neq 1$ **then**
  if $zeta < 0$ **then**
    X | *check*
    if $check = 1$ **then**
      for i=0 **to** $-zeta - 1$:
      $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
    end
    X | *check*
    if $check = 1$ **then**
      for i=0 **to** $-zeta + 1$:
      Dagger: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
    end
  end
else
  X | *check*
  if $check = 1$ **then**
    for i=0 **to** $zeta + 1$:
    Dagger: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
  X | *check*
  if $check = 1$ **then**
    for i=0 **to** $zeta - 1$:
    $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
  end
end

return *ancilla*

[#$Inv\ FFM_2$ function in InvNTT]

$check \leftarrow$ CNOT($r[length(r)-1]$, *check*)
if $check = 1$ **then**
  for i=0 **to** $f - 1$:
  Dagger: $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
end
X | *check*
if $check = 1$ **then**
  for i=0 **to** $f - 1$:
  $ancilla \leftarrow$ **ADD**(*r*, *ancilla*)
end
X | *check*

return *ancilla*

3.2. Montgomery Reduction

This function performs the Montgomery reduction multiplication on the input $\times zeta$. For each term, it is calculated on the $Z_q[X]/(X^n + 1)$ field. The parameters of Kyber NTT are fixed as $q = 3329$, $n = 256$. Figure 4 and Algorithm 3 show the operations of the Montgomery reduction quantum circuit. In the for loop, Q is $q = 3329$ and $QINV$ is the inverse of Q: -3327. Since Q and $QINV$ are known values, the result of their multiplication with the

corresponding sizes can be obtained without allocating qubits to store their values. Input $\times$ Q and input $\times$ $QINV$ are quantum-to-classical operations, not quantum-to-quantum operations. Finally, index values [0] to [15] are discarded through the 16-bit left shift and the values of indexes [16] to [31] are returned.

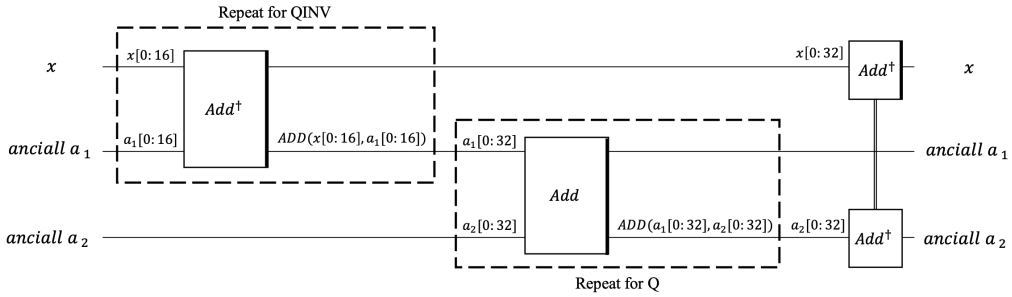


Figure 4. Quantum circuit for the Montgomery reduction; † indicates the reversible gate.

Algorithm 3: Quantum circuit for the Montgomery reduction.

Input: $a, ancilla_1, ancilla_2$

1: **for** $i = 0$ to $-QINV$ **do**
2: **Dagger**:
$tmp_1[0:16] \leftarrow$ **ADD**$(a[0:16], tmp_1[0:16])$
3: **end for**
4: **for** $i = 0$ to Q **do**
5: $tmp_2[0:32] \leftarrow$ **ADD**$(tmp_1[0:32], tmp_2[0:32])$
6: **end for**
7: **Dagger**:
$a[0:32] \leftarrow$ **ADD**$(ancilla_2[0:32], a[0:32])$

return $a[16:32]$

3.3. NTT_{sub}

The NTT_{sub} function carries out both addition and subtraction operations, contrasting the Montgomery reduction outcome with the input of a matching index. Specifically, it functions through the detailed operations of NTT_{sub1} and NTT_{sub2}. The calculations for NTT_{sub1} and NTT_{sub2} are as follows:

$$NTT = \begin{cases} NTT_{sub1} = input - Montgomery\ \ result \\ NTT_{sub2} = input + Montgomery\ \ result \end{cases}$$

In order to perform the sequential computations of the formula, it is imperative to retain both the initial input values and the Montgomery results subsequent to the NTT_{sub1} operation. Keeping all calculation targets (the input and Montgomery reduction results) is not feasible; thus, one practical solution involves storing the input in temporary qubits. Figure 5 shows the operations of the quantum circuit for NTT_{sub1} and NTT_{sub2}.

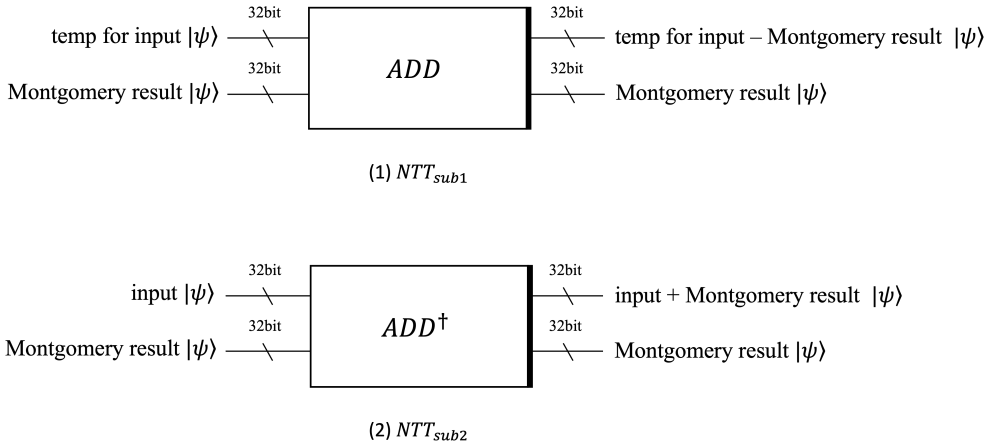


Figure 5. $NTT_{sub1,sub2}$ operation for the input and Montgomery results (above: (1) NTT_{sub1}, below: (2) NTT_{sub2}); † indicates the reversible gate.

In the NTT_{sub1}, it saves the subtraction of the temporary value from the Montgomery result in the temporary qubit, while keeping the Montgomery result as is. In the NTT_{sub2}, it stores the addition of the initial input value and the Montgomery result in the input qubit. The results of all operations are sorted in order according to the index in the NTT array.

Barrett Reduction

This function performs the Barrett reduction on the 16-bit input. The constant v used inside the Barrett reduction is calculated as $v = ((1 << 26) + q/2)$ and used for multiplication. In a quantum circuit, v denotes classic data and is used to determine the number of multiplications. Figure 6 shows the quantum circuit for the Barrett reduction. The quantum circuit operation for the Barrett reduction is the same as Algorithm 4. In the quantum circuit, the number 2^{25} used in the Barrett reduction is stored in the qubit as a binary number. The Barrett reduction is processed for the sum of two 16-bit inputs. The function employs the ripple-carry adder [29] for multiplication and addition. The multiplication with the constant v = 20,159 consumes the most quantum resources.

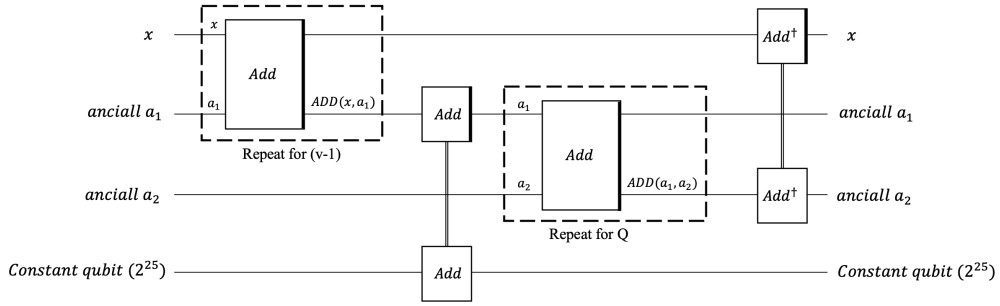


Figure 6. Quantum circuit for the Barrett reduction; † indicates the reversible gate.

Algorithm 4: Quantum circuit for the Barrett reduction.

Input: Input: a, temporary: $ancilla_1$, $ancilla_2$

$ancilla_1 \leftarrow a$

for $i = 0$ to v **do**
  $ancilla_1 \leftarrow$ **ADD**$(a, ancilla_1)$
end for

$ancilla_1 \leftarrow$ **ADD**$(2^{25}, ancilla_1)$

for $i = 0$ to 6 **do**
  $ancilla_2[i] \leftarrow$ CNOT$(ancilla_1[26+i], ancilla_2[i])$
end for

for $i = 0$ to $Q-1$ **do**
  $ancilla_2[i] \leftarrow$ CNOT$(ancilla_1[26+i], ancilla_2[i])$
end for

Dagger:
  $a \leftarrow$ **ADD**$(ancilla_2, a)$

return a

4. Evaluation

We conducted an estimation of quantum resources employing the quantum programming tool provided by ProjectQ. The NTT quantum circuit has three primary functions, while the InvNTT quantum circuit has four main functions. Each function operates in cycles (C). The quantum resources used in each sub-function of NTT are shown in Table 2. Table 3 presents the quantum resources allocated to each sub-function within the InvNTT. Since the table indicates the quantum resource used for one operation of each function, it represents the minimum resources used for operating the function at a smaller n parameter.

Table 2. Quantum resources for the NTT function.

Function	C	Quantum Gates				Depth
		CCCNOT	Toffoli	CNOT	X	
$fmul_1$	128	-	48,576	97,943	1	146,488
$fmul_2$	768	97,024	195,564	33	2	292,592
Montgomery reduction	896	-	306,270	639,184	-	945,438
NTT_{sub}	896	-	124	318	-	379

Table 3. Quantum resources for the InvNTT function.

Function	C	Quantum Gates				Depth
		CCCNOT	Toffoli	CNOT	X	
$fmul_3$	896	97,024	195,564	50	2	292,608
$fmul_4$	256	184,320	371,520	33	2	555,844
Barrett reduction	896	-	1,349,696	2,793,402	-	4,143,041
Montgomery reduction	1152	-	306,270	639,184	-	945,438

The functions FFM_1 and FFM_2 both perform multiplication using the input and the constant *zeta* in the NTT process. The main difference is that FFM_2 requires more

quantum resources than FFM_1. This is because FFM_2 needs to ascertain whether the input is positive or negative to determine the results expressed in the two's complement. In FFM_2, a multi-controlled gate is used to determine the operation based on the input's sign. In $InvFFM_1$, multiplication is conducted with the input and *zeta*, while in $InvFFM_2$, it is conducted with the input and the positive value f. $InvFFM_2$ uses more quantum resources because f is larger than *zeta*. The Montgomery reduction requires the most resources in NTT due to multiplying large numbers. NTT_{sub} uses the least amount of resources by performing simple addition and subtraction on 32-bit qubits. The Barrett reduction uses the most resources in InvNTT due to the multiplication of large numbers. The quantum resource of each function is repeated by as much as C. Consequently, the operation of NTT/InvNTT demands substantial resources. Given the quantum resources for the NTT/InvNTT within the Kyber operation, it is expected that a very large-scale quantum computer will be required to operate the entire cipher on the quantum computer. In this regard, an attempt to implement a quantum circuit for an efficient NTT/InvNTT architecture [31] may reduce quantum resources. This must be verified by efficiently implementing quantum circuits and estimating quantum resources. These efforts are anticipated to contribute to the establishment of benchmarks for efficient NTT/InvNTT quantum circuits. Therefore, this needs to be pursued in future research. According to IBM's quantum computer development roadmap, they plan to have over 1000 qubits by 2023. After 2024, the goal is to reach between 1000 and 1 million qubits. As a result, it is expected that large-scale quantum computers will appear in the future. Given this, it is expected that in the future we can evaluate the post-quantum security of CRYSTAL-Kyber using the large-scale quantum circuit.

5. Conclusions

In this paper, we propose a quantum circuit for NTT/InvNTT that is used to speed up polynomial multiplication in CRYSTALS-Kyber. The proposed quantum circuits are implemented as part of an optimization method to reduce quantum resources. The NTT/InvNTT quantum circuits elucidate the operation by showcasing pseudo-codes for the sub-functions. Finally, we estimate quantum resources based on the proposed quantum circuits and analyze them. As a result of the estimation of quantum resources, NTT has the largest depth in the Montgomery reduction function and uses the most quantum resources. In InvNTT, the Barrett reduction function has the largest depth, the most quantum resources are used, and the Montgomery reduction function uses the most qubits. In the event of the eventual development of large-scale quantum computers, it is foreseeable that the NTT operations could be executed through the utilization of the proposed quantum circuit. To the best of our knowledge, these are the first NTT/InvNTT quantum circuits to operate on the CRYSTALS-Kyber cryptographic algorithm. We anticipate that this will serve as a pivotal tool in assessing the post-quantum security robustness of CRYSTALS-Kyber.

Author Contributions: Software, G.S.; Investigation, S.E.; Writing—original draft, G.S.; Writing—review & editing, K.J. and M.S.; Supervision, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (<Q|Crypton>, Number 2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity, 40%) and this work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (Number 2018-0-00264, Research on Blockchain Security Technology for IoT Services, 50%) and This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (Number NRF-2020R1F1A1048478, 10%).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
2. Wang, Z.G.; Wei, S.J.; Long, G.L. A quantum circuit design of AES requiring fewer quantum qubits and gate operations. *Front. Phys.* **2022**, *17*, 41501. [CrossRef]
3. Zou, J.; Wei, Z.; Sun, S.; Liu, X.; Wu, W. Quantum circuit implementations of AES with fewer qubits. In Proceedings of the Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, Republic of Korea, 7–11 December 2020; Proceedings, Part II 26; Springer: Berlin/Heidelberg, Germany, 2020; pp. 697–726.
4. Jaques, S.; Naehrig, M.; Roetteler, M.; Virdia, F. Implementing Grover oracles for quantum key search on AES and LowMC. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 280–310.
5. Song, G.; Jang, K.; Seo, H. Improved Low-Depth SHA3 Quantum Circuit for Fault-Tolerant Quantum Computers. *Appl. Sci.* **2023**, *13*, 3558. [CrossRef]
6. Bathe, B.; Anand, R.; Dutta, S. Evaluation of Grover's algorithm toward quantum cryptanalysis on ChaCha. *Quantum Inf. Process.* **2021**, *20*, 394. [CrossRef]
7. Preston, R.H. Applying Grover's Algorithm to Hash Functions: A Software Perspective. *IEEE Trans. Quantum Eng.* **2022**, *3*, 1–10. [CrossRef]
8. Song, G.; Jang, K.; Kim, H.; Eum, S.; Sim, M.; Kim, H.; Lee, W.; Seo, H. SPEEDY Quantum Circuit for Grover's Algorithm. *Appl. Sci.* **2022**, *12*, 6870. [CrossRef]
9. Jang, K.; Baksi, A.; Song, G.; Kim, H.; Seo, H.; Chattopadhyay, A. Quantum Analysis of AES. *Cryptol. ePrint Arch.* **2022**. Available online: https://eprint.iacr.org/2022/683 (accessed on 13 September 2023).
10. Baksi, A.; Jang, K.; Song, G.; Seo, H.; Xiang, Z. Quantum implementation and resource estimates for rectangle and knot. *Quantum Inf. Process.* **2021**, *20*, 1–24. [CrossRef]
11. Huang, Z.; Sun, S. Synthesizing Quantum Circuits of AES with Lower T-depth and Less Qubits. *Cryptol. ePrint Arch.* **2022**. Available online: https://eprint.iacr.org/2022/620 (accessed on 13 September 2023).
12. Song, G.J.; Jang, K.B.; Seo, H.J. Resource Eestimation of Grover Algorithm through Hash Function LSH Quantum Circuit Optimization. *J. Korea Inst. Inf. Secur. Cryptol.* **2021**, *31*, 323–330.
13. Zou, J.; Li, L.; Wei, Z.; Luo, Y.; Liu, Q.; Wu, W. New quantum circuit implementations of SM4 and SM3. *Quantum Inf. Process.* **2022**, *21*, 181. [CrossRef]
14. Jang, K.; Choi, S.; Kwon, H.; Kim, H.; Park, J.; Seo, H. Grover on Korean Block Ciphers. *Appl. Sci.* **2020**, *10*, 6407. [CrossRef]
15. Lin, D.; Xiang, Z.; Xu, R.; Zeng, X.; Zhang, S. Quantum circuit implementations of SM4 block cipher based on different gate sets. *Quantum Inf. Process.* **2023**, *22*, 282. [CrossRef]
16. Luo, Q.; Li, Q.; Li, X.; Yang, G.; Shen, J.; Zheng, M. Quantum Implementaion of SM4 Block Cipher with Less Qubits. 2023. Available online: https://assets.researchsquare.com/files/rs-3105531/v1_covered_b86868c3-8eca-4d89-99c2-bd7b71258c0e.pdf?c=1688471985 (accessed on 13 September 2023).
17. Almazrooie, M.; Samsudin, A.; Abdullah, R.; Mutter, K.N. Quantum reversible circuit of AES-128. *Quantum Inf. Process.* **2018**, *17*, 1–30. [CrossRef]
18. Rahman, M.; Paul, G. Grover on KATAN: Quantum resource estimation. *IEEE Trans. Quantum Eng.* **2022**, *3*, 1–9. [CrossRef]
19. Anand, R.; Maitra, A.; Mukhopadhyay, S. Evaluation of quantum cryptanalysis on speck. In Proceedings of the Progress in Cryptology–INDOCRYPT 2020: 21st International Conference on Cryptology in India, Bangalore, India, 13–16 December 2020; Proceedings 21; Springer: Berlin/Heidelberg, Germany, 2020; pp. 395–413.
20. Chauhan, A.K.; Sanadhya, S.K. Quantum resource estimates of grover's key search on aria. In Proceedings of the Security, Privacy, and Applied Cryptography Engineering: 10th International Conference, SPACE 2020, Kolkata, India, 17–21 December 2020; Proceedings 10; Springer: Berlin/Heidelberg, Germany, 2020; pp. 238–258.
21. Song, G.; Jang, K.; Kim, H.; Lee, W.K.; Seo, H. Grover on Caesar and Vigenère Ciphers. *IACR Cryptol. ePrint Arch.* **2021**, *2021*, 554.
22. Chen, L.; Chen, L.; Jordan, S.; Liu, Y.K.; Moody, D.; Peralta, R.; Perlner, R.; Smith-Tone, D. *Report on Post-Quantum Cryptography*; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2016; Volume 12.
23. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 353–367.
24. Langlois, A.; Stehlé, D. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.* **2015**, *75*, 565–599. [CrossRef]
25. MobiSec2022. Available online: https://www.manuscriptlink.com/society/kiisc/conference/mobisec2022 (accessed on 13 September 2023).
26. Hey, T. Quantum computing: An introduction. *Comput. Control. Eng. J.* **1999**, *10*, 105–112. [CrossRef]
27. Montgomery, P.L. Modular multiplication without trial division. *Math. Comput.* **1985**, *44*, 519–521. [CrossRef]
28. Barrett, P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *Proceedings of the Advances in Cryptology—CRYPTO'86: Proceedings*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 311–323.

29. Cuccaro, S.A.; Draper, T.G.; Kutin, S.A.; Moulton, D.P. A new quantum ripple carry addition circuit. *arXiv* **2004**, arXiv:quant-ph/0410184.
30. GitHub: CRYSTALS-Kyber NTT Quantum Circuit. Available online: https://github.com/kyungzzu/CRYSTALS-Kyber_Quantum_Circuit.git (accessed on 13 September 2023).
31. Bisheh-Niasar, M.; Azarderakhsh, R.; Mozaffari-Kermani, M. High-speed NTT-based polynomial multiplication accelerator for CRYSTALS-Kyber post-quantum cryptography. *Cryptol. ePrint Arch.* **2021**. Available online: https://eprint.iacr.org/2021/563 (accessed on 13 September 2023).