Research Article

Feasibility Analysis of Cracking RSA with Improved Quantum Circuits of the Shor’s Algorithm

Xia Liu,1,2 Huan Yang,3 and Li Yang1

1State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
2School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
3Beijing Youzhuju Network Technology Co., Ltd., Beijing, China

Correspondence should be addressed to Li Yang; yangli@iie.ac.cn

Received 22 February 2023; Revised 14 June 2023; Accepted 2 November 2023; Published 24 November 2023

Academic Editor: Nadeem Sarwar

Copyright © 2023 Xia Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Since the RSA public key cryptosystem [1] has given rise to numerous scholarly discussions on this problem [2–6], in which security relies on the integer factorization. The problem of integer factorization is to find a nontrivial factor of a given composite number. It is believed to be a hard mathematical problem for which no classical polynomial-time algorithm has yet been discovered. However, with the development of quantum computing, the emergence of quantum algorithms poses a threat to cryptosystem like RSA. The most representative and compelling quantum algorithm is Shor’s algorithm [7, 8]. It can factor integers in only polynomial time in theory, offering an exponential speedup over its classical counterpart (the number field sieve [9]). Even though the Shor’s algorithm can theoretically solve problems in polynomial time that classical algorithms require exponential time to solve, the number of gates contained in the quantum circuit of a quantum algorithm determines the time to run the quantum algorithm on a quantum computer. Since the current gate estimation of the Shor’s algorithm in the existing paper does not really reflect the real implementation on the ion-trap quantum computer [10, 11], it is debatable whether it can pose a threat to RSA, which is exactly what we are trying to do.

Quantum computers implement quantum computation, which takes as input quantum states representing the superposition of all different possible inputs and simultaneously evolves them into the corresponding outputs using a sequence of unitary transformations [12–18]. Quantum computation can be described as a quantum circuit in which the unitary transformations are represented by quantum gates. Since the appearance of the Shor’s algorithm in 1994, much effort has been devoted to construct quantum circuit
of the algorithm and improvement of its quantum circuit in terms of the number of qubits and the depth of the circuits. The first work on this topic is [19], in which Vedral et al. provided an explicit quantum circuit construction of basic arithmetic operations from addition to modular exponentiation. Beckman et al. [20] estimated the number of qubits and operations required by the Shor’s algorithm: an $n$-bit integer can be factored in $O(n^3)$ operations with $5n+1$ qubits. Miquel et al. [21] analysed the impact of losses and decoherence on the quantum circuit of Shor’s algorithm. In terms of the number of qubits required, Beauregard [22] constructed a quantum circuit of the Shor’s algorithm with $2n+3$ qubits using a quantum Fourier transform (QFT)-based adder [23] and semiclassical QFT [24]; Takahashi and Kunihiro [25] reduced the number of qubits to $2n+2$ with a comparator in the modular addition operation, which is the lowest known number of qubits so far. Häner et al. [26] also constructed a quantum circuit of the Shor’s algorithm with $2n+2$ qubits based on a purely Toffoli gate-based adder that eliminates most of the cost overheads originating from rotation synthesis and enables testing and debugging. In terms of the depth of the circuit, Zalka [27] reduced the circuit depth of the Shor’s algorithm to $2^{17}n^2$, but required $24n \sim 96n$ qubits with fast Fourier transform (FFT)-based fast multiplication; and Pavlidis and Gizopoulos’s circuit [28] implemented division to compute the modular multiplication. This circuit has a depth of $2000n^2$ and requires $9n+2$ qubits.

Although there have been many attempts to improve the qubit number or the circuit depth of the Shor’s algorithm, their focus has not been on optimizing the number of CNOT gates, which greatly affects the time to run the algorithm in an ion-trap quantum computer [29, 30].

The Cliford + T set is a general method for approximating an arbitrary unitary transformation with arbitrary precision in quantum computation. The running time of the CNOT gate, which is the only two-qubit gate in the Cliford + T set, when acting on the nonadjacent qubits is much higher than that of other single-qubit gates in the ion-trap quantum computer. Furthermore, CNOTs cannot be parallel in an ion-trap quantum computer, even if they act on completely unrelated qubits. Therefore, the total CNOT count in a quantum circuit primarily determines the running time of a quantum algorithm on an ion-trap quantum computer.

### 1.1. Our Contribution

In this paper, we improve the circuit of basic operations, such as addition and multiplication, further improve the quantum circuit of the Shor’s algorithm, and then combine window technique to focus on optimizing the number of CNOT gates. Further we analyse the running time of the Shor’s algorithm on ion-trap quantum computers according to the CNOT gates number we obtained. To be specific, we modify the quantum circuit of the Shor’s algorithm to reduce the CNOT count by applying a window technique [10], Montgomery multiplication [31], and pebbling technique [32] to the modular exponentiation operation, which is the most computationally intensive component of the Shor’s algorithm. In addition, we estimated the time to run the Shor’s algorithm once in the ion-trap quantum computer and analysed the feasibility of the Shor’s algorithm based on the CNOT count of our improved circuit and the lower time limit of the CNOT gate in the ion-trap quantum computer reported in [29]. We study whether the Shor’s algorithm can be completed in a reasonable running time under the premise that the fault-tolerant quantum computer has enough space, further illustrating whether the Shor’s algorithm can really pose a threat to cryptosystems such as RSA, so our focus is not on the qubits number.

### 1.2. Outline of the Paper

The rest of this paper is organized as follows. The preliminaries section describes the Shor’s algorithm and previously constructed basic arithmetic circuits. The method section presents our work on the arithmetic circuits and the circuit of the modular exponentiation operation as well as an analysis of the corresponding CNOT count. We design a new circuit based on existing circuits to implement the Shor’s algorithm in this section. At the same time, our circuit has a smaller CNOT count. In the last section, we discuss the CNOTs required to run the Shor’s algorithm and give a result on the time estimation and feasibility of the Shor’s algorithm.

### 2. Preliminaries

#### 2.1. Ion-Trap Quantum Computing

At present, the implementation schemes of quantum computing mainly include ion traps, superconductors, linear optics, nuclear magnetic resonance, and so on [33]. The ion-trap quantum computing was proposed by Cirac and Peter in 1995 [34], in which they discussed the characteristics of ion traps and the realization of quantum gates and measurement with them. Ion-trap quantum computing exploits the interaction between electric charges and magnetic fields to restrain the motion of charged particles. Two different energy levels, e.g., the ground state and the excited state, of a confined ion are used as the two basis states of a qubit, and a quantum state is evolved by shooting different laser beams on different ions [34–36]. Compared with other physical implementations such as superconducting quantum computing, ion-trap quantum computing has attracted much attention due to its long coherence time and high qubit preparation and readout efficiency [37].

Quantum computation is usually modelled as quantum circuits, consisting of quantum gates. Hadamard gate ($H$), phase gate ($\phi$), controlled-NOT (CNOT), and $T$ gate constitute a set of universal quantum gates, that is, the Cliford + $T$ set [38, 39]. As the only two-qubit gate in the Cliford + $T$ set, the CNOT gate has two input qubits, called the control qubit and the target qubit. The target qubit will be flipped if the control qubit is $|1\rangle$ and remain unchanged if the control qubit is $|0\rangle$. The CNOT gate on an ion-trap quantum computer is realized by exciting the collective quantized motion of ions with laser. If a CNOT gate acts on two nonadjacent qubits, the time to implement it is much higher than that of other single-qubit gates. Even worse, CNOT gates can only be executed in series on an ion-trap quantum computer. Hence, the number of CNOT gates in
2.2. Shor’s Algorithm. The purpose of the Shor’s algorithm is to decompose a composite number into the product of two prime numbers, which actually solves the problem of period (or order) finding [7]. An efficient period-finding algorithm can be used to factor integers efficiently too. Given an integer \( N \) to be factored, \( a < N \) is a randomly chosen integer satisfying \( \gcd(a, N) = 1 \), the order finding problem is to find a least positive integer \( r \) such that \( a^r \equiv 1 \mod N \). The quantum order finding is shown in Figure 1, which requires two quantum registers. The first quantum register consists of \( 2n \) qubits, which qubits are initially set to \( |0\rangle \), and the second quantum register consists of \( n \) qubits, which qubits are initially set to \( |1\rangle \), where \( n = \lceil \log N \rceil \) is the number of bits needed to represent \( N \). The algorithm for quantum order finding is shown in Algorithm 1. Note that in the figures in this paper, we use the black triangles on the right side of the gate symbols to indicate quantum registers that are modified and holding the result of the computation.

The most expensive operation in the quantum order finding is the modular exponentiation by the classical known constant \( a \) as \( ME(a) \), which performs the following transform on its input quantum states:

\[
|x\rangle|0\rangle \rightarrow |x\rangle|ax \mod N\rangle.
\]

(4)

Because the \( MM(\cdot) \) operation multiplies the input quantum state \( |x\rangle \) by the classical known constant to a different quantum register that is initially set to \( |0\rangle \), the direct approach to computing \( a^x \mod N \) with \( 2n \) \( MM(\cdot) \) operations accumulates the intermediate data of each \( MM(a^2) \) operation. The following method proposed by Vedral et al. [19] is widely adopted to implement in-place modular multiplication on the input quantum state \( |x\rangle \) by the classical known constant \( a \):

(i) Apply \( MM(a) \) to the input quantum register and the ancillary quantum register, which is initially set to \( |0\rangle \) as follows:

\[
|x\rangle|0\rangle \rightarrow |x\rangle|ax \mod N\rangle.
\]

(5)

(ii) Swap the quantum states of the input quantum register and ancillary quantum register as follows:

\[
|ax \mod N\rangle|x\rangle.
\]

(6)

(iii) Apply \( MM^{-1}(a^{-1}) \), i.e., the inverse of \( MM(a) \), to the input quantum register and the ancillary quantum register as follows:

\[
|ax \mod N\rangle|0\rangle.
\]

(7)

Therefore, as shown in Figure 4, the \( ME(a) \) operation can be constructed using \( MM(\cdot) \) operations.

2.3. Previous Work on Basic Arithmetic. We compare different types of basic arithmetic circuits and use the circuits that require the fewest CNOTs to construct the Shor’s algorithm.

2.3.1. Addition. Of the different types of addition circuits, we found that Cuccaro et al. addition [23] (hereinafter the CDK adder, which is shown in Figure 5) is the one with the lowest known CNOT count. Figure 5 shows that each \( MAJ \) (i.e., compute the majority of three bits in-place) and each \( UMA \)

![Figure 1: Overall quantum circuit of the Shor’s algorithm for order finding problem.](image-url)
(i.e., UnMajority and Add) contain two CNOTs and one Toffoli gate, respectively. Thus, we conclude that an \(n\)-qubit CDK adder has a total \(n\) MAJ s and \(n\) UMA s, that is, \(4n+1\) CNOTs and \(2n\) Toffoli gates. Furthermore, based on the standard decomposition of the Toffoli gate into the Clifford + T set, which contains six CNOTs [13], we obtain that the CNOT count of the CDK adder for \(n\)-bit binary integers is \(16n+1\), and the number of qubit is \(2n+2\).

### 2.3.2. Addition by a Constant

The difference between addition and addition by a constant is that the \(y\) (i.e., the constant \(a\)) in the circuit is known, so we can simplify the addition circuit. More specifically, addition by a constant \(a\) can be constructed from the CDK adder, as shown in Figure 6. First, the constant \(a\) is encoded in an input quantum register of the CDK adder initially in \(|0\rangle\), then the CDK adder (i.e., ADD in Figure 6) is used to compute the sum, and finally, the input

---

**Algorithm 1**: The algorithm of quantum order finding.

1. Prepare the initial states \(|0\rangle|1\rangle\).
2. Apply the Hadamard transform \(H^{\otimes 2n}\) to the first quantum register to obtain \(|0\rangle|1\rangle \rightarrow 1/2^n\sum_{x=0}^{2^n-1} |x\rangle|1\rangle\).
3. Calculate the modular exponentiation by the constant \(a: 1/2^n\sum_{x=0}^{2^n-1} |x\rangle|a^x\mod N\rangle \rightarrow 1/2^n\sum_{x=0}^{2^n-1} \sum_y \exp(2\pi i xy/2^n) |y\rangle|a^x\mod N\rangle\).
4. Apply the quantum Fourier transform, \(\text{QFT}^{\otimes 2n}\), to the first quantum register: \(1/2^n\sum_{x=0}^{2^n-1} |x\rangle|a^x\mod N\rangle \rightarrow 1/2^{2n} \sum_{x=0}^{2^n-1} y \exp(\pi i xy/2^n) |y\rangle|a^x\mod N\rangle\).
5. Measure the first quantum register and determine the order of \(a\) with high probability by classical postprocessing on the measured results.

---

**Input**: \(a\) and \(N\), such that \(\gcd(a, N) = 1\).

**Output**: Integer \(r\) such that \(ar \equiv 1 \mod N\).

- (1) Prepare the initial states \(|0\rangle|1\rangle\).
- (2) Apply the Hadamard transform \(H^{\otimes 2n}\) to the first quantum register to obtain \(|0\rangle|1\rangle \rightarrow 1/2^n\sum_{x=0}^{2^n-1} |x\rangle|1\rangle\).
- (3) Calculate the modular exponentiation by the constant \(a: 1/2^n\sum_{x=0}^{2^n-1} |x\rangle|a^x\mod N\rangle \rightarrow 1/2^n\sum_{x=0}^{2^n-1} \sum_y \exp(2\pi i xy/2^n) |y\rangle|a^x\mod N\rangle\).
- (4) Apply the quantum Fourier transform, \(\text{QFT}^{\otimes 2n}\), to the first quantum register: \(1/2^n\sum_{x=0}^{2^n-1} |x\rangle|a^x\mod N\rangle \rightarrow 1/2^{2n} \sum_{x=0}^{2^n-1} y \exp(\pi i xy/2^n) |y\rangle|a^x\mod N\rangle\).
- (5) Measure the first quantum register and determine the order of \(a\) with high probability by classical postprocessing on the measured results.
quantum register is recovered to $|0\rangle$ in the same manner as in the first step. The encoding operation of a constant $a$ applies $X$ gates to the appropriate qubits that correspond to 1 in the binary representation of $a$. Different from the circuit of the addition with two unknown addends in the form of a quantum state rather than unknown constants, the adder can be simplified by a known constant, which is shown in Figure 7. Therefore, combining Figures 6 with 7, we conclude that the CNOT count of addition by a constant for $n$-bit binary integers is $13n+1$ and the number of qubits is $2n+2$.

2.3.3. Controlled Addition. We construct the controlled CDK adder [41] by controlling all the MAJ and UMA blocks of the CDK adder, which is shown in Figure 8. The CNOT counts of the controlled MAJ block and the controlled UMA block are both 13, such that the CNOT count of the controlled CDK adder for $n$-bit integers is $26n+6$. Since there is one more control qubit, the number of qubits is $2n+3$.

2.3.4. Controlled Addition by a Constant. Based on the addition of a constant, we control the two encoding operations of the constant with the intermediate uncontrolled CDK adder, as shown on the left of Figure 9, instead of the controlled CDK adder only shown on the right of Figure 9. Considering that the average number of CNOT for encoding operations of 0 or 1 is 1/2, then two $n$-qubit encoding operations for $a$ need $n$ CNOTs. Since we do not know exactly what values are encoded, we need to consider the addition using $16n+1$ of ADD in Figure 8. Thus, the CNOT count of the controlled addition for $n$-bit integers by a constant are $17n+1$. Compared to Figure 8, the number of qubits in Figure 9 has not changed, that is, $2n+3$.

2.3.5. Comparison. We use the comparison in Figure 10 based on the MAJ blocks given in [23], of which the CNOT count is $16n+1$ and qubits number is $2n+2$ for $n$-bit integers.
Figure 8: Controlled form of CDK.

Figure 9: Controlled addition by a constant based on addition by a constant.

Figure 10: Quantum circuit of comparison based on the MAJ block.
3. Method

In this section, we first give the implementation of the basic arithmetic with the lowest known number of CNOTs and then construct an improved modular exponentiation of a constant by accumulating intermediate data and using a windowing technique to reduce the CNOT number from $O(n^3)$ to $O(n^3/\log n)$. Moreover, we give the specific number of CNOT using in the Shor’s algorithm.

3.1. Improvement of the Basic Arithmetic. Using the basic arithmetic circuits described in the previous section, we improve the modular addition, shift, and modular doubling circuits. We also construct new controlled comparison and controlled modular addition circuits according to the previous comparison and modular addition circuits. Both have the fewest CNOT number.

3.1.1. Controlled Comparison. Based on Figure 10, we constructed a controlled comparison by controlling the CNOT and X gates and holding the result of the comparison, which is shown in Figure 12. The CNOT count of the controlled comparison for $n$-bit integers is $16n+7$ and the number of qubits is $2n+3$.

3.1.2. Modular Addition. We improved the modular addition (as shown in Figure 13) using the basic arithmetic of addition and comparison constructed before, in which the subtraction is the inverse of addition. The CNOT count of our modular addition is $61n+6$. Since the two auxiliary bits in ADD can be restored, they can be reused in later operations. Therefore, the qubits number of modular addition is $3n+3$.

3.1.3. Controlled Modular Addition. We constructed the controlled modular addition by controlling the first addition and the last comparison of modular addition, as shown in Figure 14. The CNOT count and the qubits number of our controlled modular addition is $71n+17$ and $3n+4$, respectively.

3.1.4. Shift. The functions of left shift and right shift are as follows.

- Left shift: $|0x_{n-1}\cdots x_1x_0\rangle \rightarrow |x_{n-1}\cdots x_1x_00\rangle$.
- Right shift: $|x_{n-1}\cdots x_1x_0\rangle \rightarrow |0x_{n-1}\cdots x_1x_0\rangle$.

The method to implement shift is to use the SWAP gate. However, we note that there is no need to swap two qubits with a SWAP operation if a qubit is known to be in the state of $|0\rangle$. Hence, we construct the left shift and right shift for an $n$-qubit quantum register, as shown in Figure 15, for which the CNOT count are both $2n$ with $n+1$ qubits.

3.1.5. Modular Doubling. As shown in Figure 16, we improve the modular doubling, by replacing the subtraction of the constant $N$ in [43] with a comparison of the constant $N$. The CNOT count and the qubits number of our modular doubling is $31n+15$ and $2n+2$, respectively.

3.1.6. Modular Multiplication. Roetteler et al. [41] reported quantum circuits of two approaches to computing the modular multiplication of two factors both in the form of quantum states: fast modular multiplication [44] and Montgomery modular multiplication [31]. Based on the basic arithmetic we constructed, we improved the circuits for these two modular multiplications to have fewer CNOT numbers, which are shown in Figures [17] and [19], respectively.

By the binary expansion of the first factor $x$, product $x \cdot y$ can be written as follows:

$$x \cdot y = \sum_{i=0}^{n-1} 2^i x_i y$$

$$= x_0y+2(x_1y+2(x_2y+\cdots+2x_{n-1}y+\cdots)).$$

Therefore, fast modular multiplication decomposes $x \cdot y \bmod N$ into a sequence of conditional modular additions and modular doublings. Based on our construction of basic arithmetic, we improved the circuit of fast modular multiplication, as shown in Figure 17 and obtain a CNOT count of $102n^2-64n-32$. Since the qubits used in ModDbl is $2n+2$, the number of qubits in Figure 17 is $4n+2$.

Montgomery modular multiplication computes $x \cdot y/2^n \bmod N$ instead of $x \cdot y \bmod N$ for the input factors $x$ and $y$ in the form of quantum state. Figure 18 shows the whole quantum circuit of Montgomery modular multiplication, which consists of forward and backward Montgomery modular multiplications. If an input factor $x$ is in the form of Montgomery representation $(x \cdot 2^n \bmod N)$, then the result of the Montgomery modular multiplication will be $x \cdot y \bmod N$, which is the actual form required.

Based on our construction of basic arithmetic, the CNOT count of forward Montgomery modular multiplication, which is shown in Figure 19, is $45n^2+17n-4$ with $5n+2$ qubits, whereas the CNOT count of the whole Montgomery modular multiplication is $90n^2+35n-8$ with $6n+2$ qubits.

3.2. Proposed Circuit for the Shor’s Algorithm. Here, we modify the quantum circuit of the Shor’s algorithm to reduce the CNOT count from $O(n^3)$ to $O(n^3/\log n)$ by applying a window technique [10], Montgomery multiplication [31], and pebbling technique [32] to the modular exponentiation operation, which is the most computationally intensive component of the Shor’s algorithm. In addition, we are the first to estimate the time to run the Shor’s algorithm once in the ion-trap quantum computer and analyse the feasibility of the Shor’s algorithm based on the CNOT count of our improved circuit and the lower time limit of the CNOT gate in the ion-trap quantum computer reported in [29]. We further study whether the Shor’s algorithm can be completed in a reasonable running time under the premise that the fault-
\[ y_i = 1 \]
\[ y_i = 0 \]

**Figure 11:** The form of MAJ when \( y \) is known.

\[
\begin{array}{ccc}
|c_i\rangle & \quad & |c_i \oplus y_i\rangle \\
|x_i\rangle & \quad & |x_i \oplus y_i\rangle \\
|y_i\rangle & \quad & |c_{i+1}\rangle
\end{array}
\]

**Figure 12:** Controlled comparison.

**Figure 13:** Modular addition. The box ADD represents the quantum circuit of addition in Figure 5. The COMP (\( N \)) represents the circuit of comparison by a constant in Figure 11, and the COMP is the circuit of comparison with \( 16n+1 \) CNOT count. The SUB is the inverse of the ADD with \( 16n+1 \) CNOT count.

**Figure 14:** Controlled modular addition. The meaning of the boxes here are the same as in Figure 13, except that there is a control qubit \( |\text{ctrl}\rangle \).
Figure 15: Quantum circuit of (a) left shift and (b) right shift.

Figure 16: Quantum circuit of modular doubling.

Figure 17: Fast modular multiplication. We replace the first controlled modular addition with a controlled copying operation because the modular sum equals one addend if the other addend is 0. The box of ModDb1 is the circuit shown in Figure 16 with 31n + 15 CNOT count. The circuit of controlled ModAdd is shown in Figure 14 with 71n + 17 CNOT count.

Figure 18: Whole quantum circuit of Montgomery modular multiplication which performs $|x\rangle|y\rangle|0\rangle \rightarrow |x\rangle|y\rangle|x \cdot y/2^n \mod N\rangle$. The forward Montgomery modular multiplication computes $x \cdot y/2^n \mod N$ with the information of the intermediate result in each round held in ancillary qubits. To recover the ancillary qubits of the forward Montgomery modular multiplication, we can copy the result $x \cdot y/2^n \mod N$ to another new quantum register and apply the backward Montgomery modular multiplication, that is, run the circuit of forward Montgomery modular multiplication backwards.
tolerant quantum computer has enough space, which illustrates whether the Shor’s algorithm can really pose a threat to cryptosystems such as RSA.

3.2.1. Accumulation of the Intermediate Data. As shown in Figure 20, instead of erasing the intermediate data in each controlled MM(·) operation, as implemented in Vedral et al.’s method, we accumulate the intermediate data until the modular exponentiation $a^x$ is computed by the last controlled MM($a^{2^{m-1}} \mod N$) operation. We also erase the intermediate data using all controlled inverse MM(·) operations in reverse order, except for the last controlled MM($a^{2^{m-1}} \mod N$) operation.

3.2.2. Windowing Technique. Here we use the window form described in [10] to design the quantum circuit of the Shor’s algorithm, reducing the CNOT number $N$ from $O(n^3)$ to $O(n^2)$. Windowing techniques, such as the fast implementation of the cyclic redundancy check parity check, are widely used to reduce the number of operations in classical computation [44]. Gidney [10] showed that such techniques are also useful for optimizing quantum circuits in quantum computation and presented various windowed quantum arithmetic circuits, including a windowed modular exponentiation with nested windowed modular multiplications.

The key to the windowing technique in quantum computation is to merge several controlled operations acting on the target qubits into a single operation acting on the target qubits and the corresponding qubits which is created and recovered by the table lookup operation.

To be specific, the ME($a$) operation can be decomposed into a series of controlled MM(·) operations, which is the equation (2), and the windowing technique can be used to compute $m$ ME($a$) operation first. That is, $m$ $x_i$ are selected first and the $2^m$ cases $T_i = a^{x_i}$ represented by $m$ $x_i$ can be calculated and stored in an $n$-qubit register, where $x = \sum_{j=1}^{m} 2^j x_j$. For example, when the window size $m = 3$, then there is 8 cases $T_i$ from $x_1$ to $x_3$, that is,

$$T_i = a^{2 \times x_1 + 2 \times x_2 + 2 \times x_3}.$$  

We iterate all control qubits in groups with a window size of $m$ instead of iterating them individually. For the $m$-controlled-qubit MM(·) operations in each group, we merge them as follows:

(i) Retrieve the actual value of the factor used to obtain the result from the precomputed table using $m$ control qubits. Create a special quantum state corresponding to the value found in the ancillary quantum register.

(ii) Perform modular multiplication of the value of the target quantum state by the value of the special quantum state.

(iii) Retrieve the actual value of the factor used to obtain the result from the precomputed table using $m$ control qubits. The special quantum state corresponding to the value found in the ancillary quantum register is recovered.

The table lookup operations in steps i and iii perform

$|x\rangle|0\rangle \rightarrow |x\rangle|x_0\rangle$, where $|x_0\rangle$ is the value obtained from the classical precomputed table addressed by $x$. We provide the quantum circuit of the table lookup operation without the control qubit based on [10] in Figure 21.

Modular multiplication with both factors in the form of a quantum state can be computed by a modular multiplication operation, denoted as $MM$, that performs the following transform on its input quantum states:

$$|x\rangle|y\rangle|0\rangle \rightarrow |x\rangle|y\rangle|x \cdot y \mod p\rangle,$$

The quantum circuit of the windowed ME($a$) operation based on a construction with accumulated intermediate data is shown in Figure 22, which greatly reduces the CNOT number of the quantum circuit for the Shor’s algorithm from $O(n^3)$ to $O(n^3/\log n)$ with $O(n^2)$ qubits.

Now, we analyse the whole circuit of the Shor’s algorithm to obtain a specific CNOT number. Given the $n$-bit integer to be factored and the window size $m$, the precise number of CNOTs in our quantum circuit for modular exponentiation that minimizes window size $m$ for input bit size $n$ is $(\left\lceil \frac{2n}{m} \right\rceil - 1)((n+1)2^n + 90n^2 + 34n - 10) + (n+1)2^{m-m(\left\lceil \frac{2n}{m} \right\rceil - 1) + 102n^2 - 54n - 42}$. Note that when $m = O(\log n)$, $N$ is a polynomial.
Figure 20: Construction of the $\text{ME}(a)$ operation by $\text{MM}(\cdot)$ operations with the intermediate data accumulated.

Figure 21: Quantum circuit of table lookup operation with the window size $m = 3$. If the control qubits contain the value $i$, then we bind the value $T_i$ from the classical precomputed table addressed by $i$ to the lowest register by applying the $X$ gate to the appropriate qubits of the lowest register. The binding operation is denoted as a rectangle with the value used for binding.

Figure 22: Windowed ME($a$) operation based on a construction with accumulated intermediate data.
of $n$. We calculate $\partial N(n, m)/\partial m$, and for each $n_i \in (1024, 4096)$, we use Matlab to approximate the zero $m_i$ of $\partial N(n, m)/\partial m$ to obtain a pair $(n_i, m_i)$. Because $m$ should be an integer, we round each $m_i$ up and down to get $m_i$ and $m_i'$, respectively. Then letting $N_{min} = \min(N(n_i, m_i'), N(n_i, m_i'))$, for each $i$ and fitting $N$ with respect to $n$ based on all the pairs $(n_i, N_{min})$, we obtain $N = 217n^3/\log_2 n$. Adding the $4n^2 + n$ CNOTs used for QFT on 2qubits, we conclude that the total number of CNOTs in our implementation of the Shor’s algorithm is

$$N = 217 \frac{n^3}{\log_2 n} + 4n^2 + n. \quad (10)$$

Reference [29] gives the lower limit of the time needed to execute a CNOT gate in an ion-trap quantum computer, which is approximately $2.85 \times 10^{-4}$ s. Combined with the number of CNOTs needed to run the Shor’s algorithm, the time to break 1024-bit RSA would be at least 72 years after three levels of coding.

4. Discussion and Conclusion

Although there have been many attempts to improve the qubit number or the circuit depth of Shor’s algorithm, their focus has not been on optimizing the number of CNOT gates, which greatly affects the time to run the algorithm in an ion-trap quantum computer. In this paper, we improve the quantum circuit of basic arithmetic operations, including addition, controlled addition, and comparison. Table 1 summarizes the CNOT numbers for the basic arithmetic. We constructed a circuit for controlled comparison, controlled modular addition, and modular multiplication. Based on this work, the quantum circuit that can run Shor’s algorithm was greatly improved. We reduce the CNOT gates number from $O(n^3)$ to $O(\log n)$ and calculate the specific number of CNOT gates required by the algorithm. The time required by the Shor’s algorithm to attack 1024-bit RSA was estimated, i.e., 72 years, which means it is hard to attack the RSA using the ion-trap quantum computer in a reasonable time. However, this estimated time does not consider the fault tolerance of the circuit, which we will study in the future.

According to the results of the CNOT number, we can consider the lower bound of CNOT number required by the Shor’s algorithm. Assume that the time required to execute one CNOT is $t$, and the lower bound of the CNOT-count of the algorithm is $N$, which is a function of the number of qubits $n$. Then, the lower bound of the runtime of the Shor’s algorithm can be expressed as $T = N(n)t$. Modular exponentiation can be constructed using modular multiplication, and modular multiplication can be constructed using modular addition. Therefore, the number of CNOTs required for modular exponentiation must be greater than that required for modular addition. Because the quantum circuit of modular addition adds a modular operation, the number of CNOTs required by modular addition must be larger than that of the addition circuit. For two $n$ qubits $x, y$, we have that $c_{i+1} = x_i + (x_i + y_i)(x_i + c_i)$, $s_i = x_i + y_i + c_i$, where $x_i$ and $y_i$ are the $i$-th bits of the binary representation of $x, y$. $c_{i+1}$ is the $i$-th carry, and $s_i$ is the sum of the $i$-th bits. Therefore, each qubit addition requires at least one Toffoli and three CNOTs. Thus, the addition of $n$ qubits requires at least $9n$ CNOTs. However, we have not obtained a specific lower bound for the number of CNOT gates required to run the Shor’s algorithm, so we plan to calculate the lower bound tighter in our next work.

Data Availability

The data used in this study are available within the article.

Disclosure

A preprint has previously been published in [30].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Key Research Program of the Chinese Academy of Sciences (Grant no. ZDRW- XX-2022-1), National Natural Science Foundation of China (Grant no. 61672517), and National Natural Science Foundation of China (Key Program, Grant no. 61732021).

References

Security and Communication Networks


