Quantum Fourier Transform and Its Application in Shor's Algorithm

Zhongwei Wang[1, *], Xirui Gou[2], Ruiqing Fu[3], Zhixuan Fu[4]

[1]Department of Physics, Nanjing University, Nanjing, 210093, China

[2]Watkinson School, Hartford, CT, 06115, USA

[3]Department of Physics, Beihang University, Beijing, 102206, China

[4]Shanghai Starriver Bilingual School, Shanghai, 201108, China

*Corresponding Author's E-mail: 171840742@smail.nju.edu.cn

Keywords: Quantum, Fourier Transform, Shor's Algorithm

Abstract: Quantum Fourier transform (QFT) plays an eminent role in quantum computation. It creates a superposition of different quantum states, allowing simultaneous calculation, which would take many steps were the same program implemented on a classical computer. The computational speed of a quantum computer is thus boosted dramatically. In this article, we explain the application of QFT in Shor's algorithm, which was proposed by Peter Shor to factor large integers on quantum computers. Specifically, the principle and design of quantum Fourier transform are explained. We stress the subtle distinction between QFT and its inverse. Since former articles did not emphasize it, we hope it could be a supplement to former articles. Our next work is to verify the existing articles on executing Shor's algorithm by conducting two experiments on IBMQ of factoring N = 15 in two ways (a = 7 and a = 11). We find that the effect of quantum entanglement might be crucial to the speed boost of factoring large integers in Shor's algorithm.

1. Introduction

Fourier transform is one of the most powerful tools in math and physics, eluding almost no theoretical work from the theory of deferent and epicycles proposed in 3rd century BCE, to the latest theory of solid state physics, signal processing, and optical imaging. Based on the most simple but also most profound belief that everything can be decomposed into a superposition of waves of certain pure frequencies, Fourier transform enables us to perceive the unperceivable space to our naked eyes – the reciprocal space, which is also characterized by a four-vector (ω, k_x, k_y, k_z), as opposed to the four-vector $(t,\ x,\ y,\ z)$ which characterizes real space. Owing to the mighty power of Fourier transform shown in myriad fields of math and physics, there is no surprise that its quantum version, or rather the discrete version, could also play an eminent role in quantum computation, such as in quantum phase estimation algorithm [1] for estimating the eigenvalues of a unitary operator, and in algorithms for hidden subgroup problem [2].

One of the most prominent roles quantum Fourier transform has played in quantum computation is that in Shor's algorithm, proposed in 1994. Running on the quantum circuit, Shor's algorithm can factor large numbers into the multiplication of two integers in polynomial time $\mathrm{O}((\log N)^k)$ [3], much faster than the classical algorithm which takes up sub-exponential time, $\mathrm{O}(e^{1.9(\log N)^{2/3}(\log\log N)^{1/3}})$, at best to factor the same number [4]. The contrast between the two algorithms will become strikingly stark as N goes up. For example, to factor a 200-digit number N, it will take a common PC 10^3 years. However, it would be nearly instantly broken down by Shor's algorithm were there an available quantum computer. The reason why factoring large numbers is crucial is that modern public key cryptography used in internet communication, such as the RSA scheme [5], is based on the assumption that large integer factorization is computationally intractable. However, this assumption holds for non-quantum computers, but not for quantum computers. Consequently, anyone with a quantum computer can pose an immediate threat to the privacy and security of the Internet nowadays. The efficiency of Shor's algorithm is attributed to two key elements: quantum Fourier transform and modular exponentiation. The former takes advantage of quantum superposition to do multiple calculations simultaneously. The latter creates a quantum entanglement between register A and register B in Figure so that the measuring of register B will affect that of register A. The explanation will become clear in the latter context.

However, the distinction between quantum Fourier transform and the inverse quantum Fourier transform is not sufficiently illustrated in the previous research. And most research on Shor's algorithm mainly focus on the theoretical basis, not on the specific construction of the circuit. Thus, we illustrate the distinction between QFT and IQFT both in the formula and circuit design. Furthermore, this article is dedicated to the specific construction of Shor's algorithm with the experiments of factoring N = 15 with a = 7 and a = 11 separately conducted on IBMQ. We also made a flow chart (See Figure) of Shor's algorithm, hoping it could make the procedure easier to understand.

2. QFT and IQFT

Mathematically speaking, quantum Fourier transform is a special case of discrete Fourier transform in Hilbert space. It projects a vector in Hilbert space to another vector in the same space.

$$QFT(|1\rangle) = |0\rangle + e^{i\pi/4}|1\rangle + e^{i\pi/2}|2\rangle + e^{i3\pi/4}|3\rangle + e^{i\pi}|4\rangle + e^{i5\pi/4}|5\rangle + e^{i3\pi/2}|6\rangle + e^{i7\pi/4}|7\rangle \quad (1)$$

$$QFT(|2\rangle) = |0\rangle + e^{i\pi/2}|1\rangle + e^{i\pi}|2\rangle + e^{i3\pi/2}|3\rangle + e^{i2\pi}|4\rangle + e^{i5\pi/2}|5\rangle + e^{i3\pi}|6\rangle + e^{i7\pi/2}|7\rangle \quad (2)$$

If state $|1\rangle$ is put in, the output will be a superposition of all states from $|0\rangle$ to $|7\rangle$ with the same amplitude. But the phase difference between any two adjacent states is $ei\pi/4$. So the frequency of the output is 1. If state $|2\rangle$ is put in, the amplitude of all possible states is also the same, but the phase difference between any two adjacent states doubles, namely, the output has a frequency of 2. If the input is a superposition state, say $|1\rangle$ and $|2\rangle$, then the output is the superposition of the wave of frequency 1 and the wave of frequency 2. See Figure 1 (4-qubit QFT)

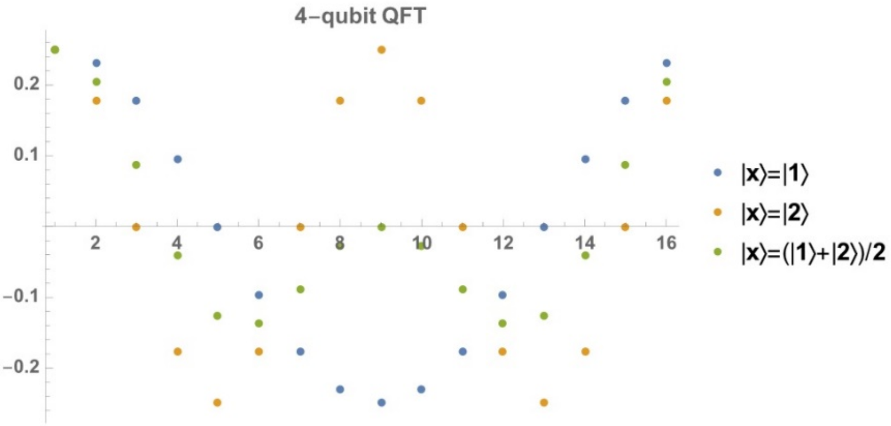

Figure 1 Demonstration of how QFT works

The inverse quantum Fourier transform will do the opposite. It analyzes the periodicity of the input states and outputs their frequencies. (Generally, the input states have multiple frequency components.)

$$IQFT(|0\rangle + e^{i\pi/4}|1\rangle + e^{i\pi/2}|2\rangle + e^{i3\pi/4}|3\rangle + e^{i\pi}|4\rangle + e^{i5\pi/4}|5\rangle + e^{i3\pi/2}|6\rangle + e^{i7\pi/4}|7\rangle) = |1\rangle \quad (3)$$

$$IQFT(|0\rangle + e^{i\pi/2}|1\rangle + e^{i\pi}|2\rangle + e^{i3\pi/2}|3\rangle + e^{-2\pi}|4\rangle + e^{i5\pi/2}|5\rangle + e^{i3\pi}|6\rangle + e^{i7\pi/2}|7\rangle) = |2\rangle \quad (4)$$

In general, q-qubit QFT is defined as

$$U_{QFT}(|x\rangle) = \frac{1}{Q}\sum_{y=0}^{Q-1} \omega^{xy}|y\rangle \quad (5)$$

where $Q = 2^q$, ω is q-th root of unity $e^{2\pi i/q}$. The definition of IQFT only differs from QFT by a negative sign in the exponent.

$$U_{IQFT}(|x\rangle) = \frac{1}{Q}\sum_{y=0}^{Q-1} \omega^{-xy}|y\rangle \quad (6)$$

3. Realization of QFT and IQFT on IBMQ

3.1 Design of QFT [6]

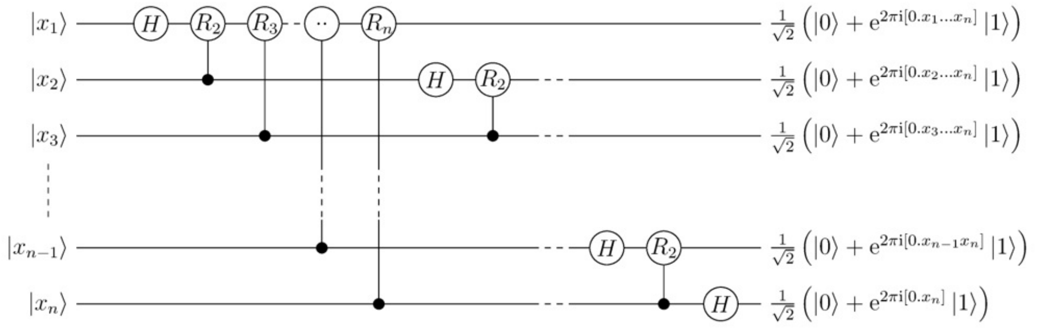
$|x_1\rangle$
$|x_2\rangle$
$|x_3\rangle$
$|x_{n-1}\rangle$
$|x_n\rangle$
H
R_2
R_3
R_n
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_1 \dots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_2 \dots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_3 \dots x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_{n-1} x_n]}|1\rangle\right)$
$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle\right)$

Figure 2 Circuit of QFT [7]

The circuit of QFT is shown in Figure 2. The quantum gates used in this circuit are the Hadamard gate, which is designed for creating superposition states and defined as follow.

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (7)$$

And the controlled-Rz gate, which is defined as follow.

$$R_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^m} \end{bmatrix} \quad (8)$$

Assume the input is $|x_1 x_2 \dots x_n\rangle$. The first operation of this circuit is to apply the Hadamard gate to the first qubit, which produces the quantum state.

$$\frac{1}{2^{1/1}}(|0\rangle + e^{2\pi i[0.x_1]}|1\rangle) \otimes |x_2 \dots x_n\rangle \quad (9)$$

Note that $[x_1 x_2 \dots x_n]$ is the expression for binary number $x_1 2^{n-1} + x_2 2^{n-2} + \dots + x_n 2^0$, and $\otimes$ denotes the state product. Then apply the controlled-R_2, R_3 through R_n gate successively to the first qubit, producing the state.

$$\frac{1}{2^{1/2}}(|0\rangle + e^{2\pi i[0.x_1 x_2 \dots x_n]}|1\rangle) \otimes |x_2 \dots x_n\rangle \quad (10)$$

By the same token, apply the Hadamard gate and the controlled-R_3, R_4 *through* R_n gate to the second qubit, outputting the state.

$$\frac{1}{2^{2/2}}(|0\rangle + e^{2\pi i[0.x_1 x_2 \dots x_n]}|1\rangle) \otimes (|0\rangle + e^{2\pi i[0.x_2 x_3 \dots x_n]}|1\rangle) \otimes |x_3 \dots x_n\rangle \quad (11)$$

Operate the rest of the qubits in the same manner. That is, generally, apply the Hadamard gate and the controlled-R_{m+1}, R_{m+2} through R_n gate to the m-th qubit. After applying all quantum gates, it can be thus concluded that the final state that the circuit above produces is as follow.

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i[0.x_1 x_2 \dots x_n]}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i[0.x_2 x_3 \dots x_n]}|1\rangle\right) \otimes \dots \otimes \left(|0\rangle + e^{2\pi i[0.x_n]}|1\rangle\right) \quad (12)$$

On the other hand, from the definition of QFT:

$$\mathrm{U_{QFT}}(|\mathrm{x}\rangle) = \frac{1}{2^{n/2}} \sum_{\mathrm{y}=0}^{2^n-1} e^{2\pi\mathrm{i}\frac{xy}{2^n}} |\mathrm{y}\rangle$$
$$= \frac{1}{2^{n/2}} \sum_{\mathrm{y}=0}^{2^n-1} e^{2\pi\mathrm{i}\frac{x[y_1y_2\dots y_n]}{2^n}} |y_1y_2..y_n\rangle$$
$$= \frac{1}{2^{n/2}} \sum_{y_1=0}^{1} e^{2\pi\mathrm{i}\frac{x[y_1 0\dots 0]}{2^n}} |y_1\rangle \otimes \sum_{y_2=0}^{1} e^{2\pi\mathrm{i}\frac{x[0 y_2\dots 0]}{2^n}} |y_2\rangle \otimes \dots \otimes \sum_{y_n=0}^{1} e^{2\pi\mathrm{i}\frac{x[0\dots 0 y_n]}{2^n}} |y_n\rangle \tag{13}$$

Note that $\frac{[x_1x_2\dots x_n]\cdot[y_1 0\dots 0]}{2^n} = y_1 \cdot [x_1 \dots x_{n-2}x_{n-1}.x_n]$ and $e^{2\pi i y_1\cdot[x_1\dots x_{n-2}x_{n-1}.x_n]} = e^{2\pi i y_1\cdot[0.x_n]}$.

$$\mathrm{U_{QFT}}(|\mathrm{x}\rangle) = \frac{1}{2^{\mathrm{n}/2}} \left(|0\rangle + \mathrm{e}^{2\pi\mathrm{i}[0.\mathrm{x_n}]}|1\rangle\right)\left(|0\rangle + \mathrm{e}^{2\pi\mathrm{i}[0.\mathrm{x_2x_3\dots x_n}]}|1\rangle\right) \otimes \dots \otimes \left(|0\rangle + \mathrm{e}^{2\pi\mathrm{i}[0.\mathrm{x_1x_2\dots x_n}]}|1\rangle\right) \tag{14}$$

It seems that the order in (14) is reversed with respect to (7). However, the output should be read from bottom to top, just It shows that the circuit in Figure 2 is indeed the circuit for QFT. Obviously, QFT is a unitary transform because each gate in the circuit is unitary.

3.2 Implementation of QFT and IQFT [8]

The structure of IQFT is similar to QFT. One only needs to reverse the sequence of QTF and replace each gate with its conjugate. This is because if $QFT = \mathrm{U_1} \dots \mathrm{U_n}$, where each $\mathrm{U_i}$ is an individual gate, and $\mathrm{QFT}^{-1} = (U_1U_2...U_n)^{-1} = \mathrm{U_n^{-1}} \dots \mathrm{U_1^{-1}} = \mathrm{U_n^\dagger} \dots \mathrm{U_1^\dagger}$. IQFT for n = 3 is the right half part of Figure .

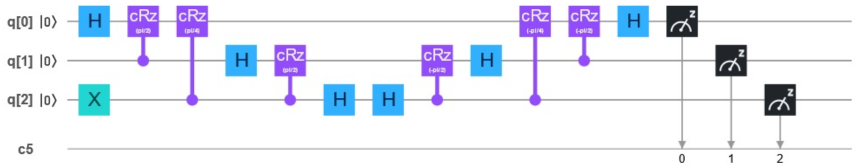

Figure 3 QTF(IQFT(|x⟩))= |x⟩

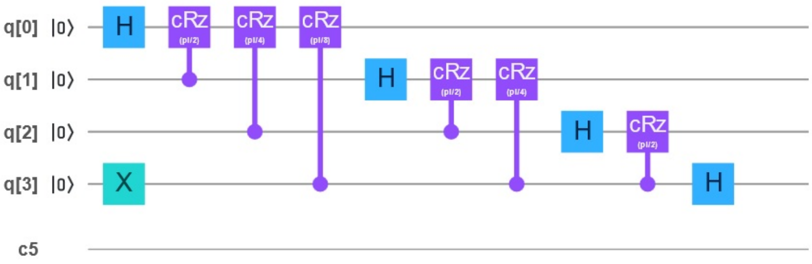

Figure 4 Circuit for 4-qubit QFT

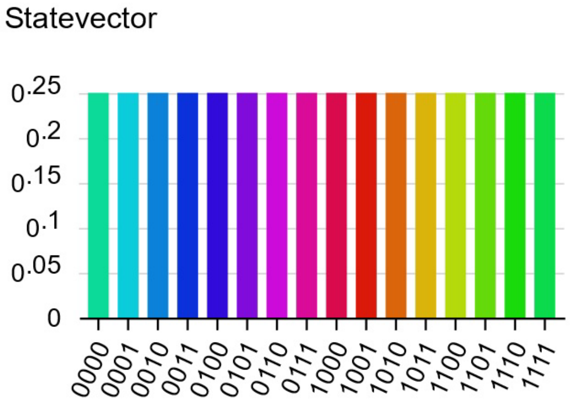

Figure 5 Result of 4-qubit QFT

Different colors stand for different phase from 0 to 2π. As is shown Figure 5, when the input is x=1, the output is a superposition of all 8 possible states which have the same amplitude, only differ from adjacent states in phase by $\pi/4$.

4. Shor's Algorithm [9]

Shor's algorithm consists of two subroutines: the classical one and the quantum one. The classical subroutine is operated on a traditional circuit while the quantum part works on a quantum circuit, using the property of quantum superposition to drastically boost the computational speed.

4.1 Classical Subroutine

Suppose a large integer N is given to be factored. The classical circuit generates a random integer $a < N$ and calculates the great common divisor between a and N by the Euclidean algorithm.

•If $\gcd(\mathrm{a,N}) \neq 1$, then $\gcd(\mathrm{a,N})$ is a non-trivial factor of N. The factorization is thus completed.

•If $\gcd(\mathrm{a,N}) = 1$, a and N are coprime. Then use the quantum period-finding subroutine in Figure to find the period p of the following equation

$$f(\mathrm{x}) = \mathrm{a}^{\mathrm{x}} \bmod N \tag{15}$$

which means $\mathrm{a}^{\mathrm{x+p}} = \mathrm{a}^{\mathrm{x}} \bmod N$ so $\mathrm{a}^{\mathrm{p}} \equiv 1$ (mod N). According to Euler's theorem, $\mathrm{a}^{\varphi(\mathrm{N})} \equiv 1(\mathrm{mod\ N})$, where a and N are coprime and $\varphi(\mathrm{N})$ denotes Euler's totient function. So p is a factor of or equal to $\varphi(\mathrm{N})$. For a = 7, N = 15, $\varphi(15) = 15 \cdot (1 - 1/3)(1 - 1/5) = 8$. It can be verified that p = 4 is the least exponent of 7 to make $7^{\mathrm{p}} = 2401 \equiv 1$ (mod 15). And 4 is indeed one factor of 8. For a = 13, N = 42, $\varphi(42) = 42 \cdot (1 - 1/2)(1 - 1/3)(1 - 1/7) = 12$. $13^2 = 169 \equiv 1$ (mod 42). So p = 2 is the least exponent of 13 that satisfy $f(p) = 1$. And 2 is unsurprisingly a factor of 12.

Rewrite this equation as $N|(a^{\mathrm{p}} - 1) = (a^{p/2} + 1)(a^{p/2} - 1)$. So either $\left(a^{p/2} + 1\right)$ or $\left(a^{p/2} - 1\right)$ is likely to contain the factors of N. If so,$\gcd\left(a^{p/2} + 1, N\right)$ and $\gcd\left(a^{p/2} - 1, N\right)$ are the factors of N, and we are done.

However, the following two cases are exceptions. If the period p is odd, $a^{p/2}$ is not an integer. Or if $\mathrm{N}|(a^{p/2} + 1)$ or $\mathrm{N}|(a^{p/2} - 1)$, then $\gcd\left(a^{p/2} + 1, \mathrm{N}\right) = \mathrm{N}$ or $\gcd\left(a^{p/2} - 1, \mathrm{N}\right) = \mathrm{N}$, which tells us nothing about the factors of N. In these two cases, another random number a will be generated and the process above will iterate. Nevertheless, the probability of finding the number which satisfies the previous two conditions is about 25%, not bad [6].

4.2 Quantum Subroutine (Period-Finding)

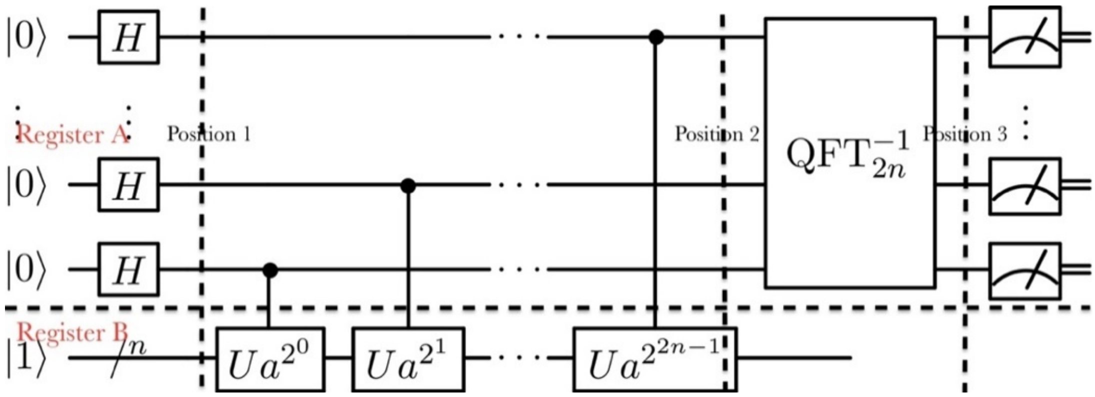

Figure 6 Circuit for Shor's Algorithm [12]

Figure 6 is composed of two registers: A and B. Suppose A contains q qubits, and B contains n qubits. Denote 2q = Q. There are some conditions for q and n to meet. q should at least make Q − 1 > 2p. The reasons will become clear later (See Figure). 2^n should at least be greater than N because register B is used to store ax mod N, which has the multitude of N. Conveniently, as is shown in Figure , q is chosen to be 2n to guarantee Q − 1 > 2p since the period p is always less than N.

First, implement n Hadamard gates on register A. To see the results, recall the Hadamard gate on a single qubit:

$$H(|0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{16}$$

It converts |0⟩ to a superposition of all possible states with the same amplitude. Hadamard gates on q qubits are similar, transforming the initial state into all possible states with the same amplitude. So, at position 1,

$$|\psi_1\rangle = U_{H^{\otimes q}}(0_q,\ 0_n\rangle) = \frac{1}{\sqrt{Q}}\sum_{x=0}^{Q-1}|x,\ 0\rangle \tag{17}$$

Uf gate is built as follow.

$$U_f(|x,\ 0\rangle) = |x,\ \mathrm{f(x)}\rangle \tag{18}$$

where f(x) = ax mod N. We can see Uf is dependent on specific a and N. Implement Uf to register B. So at position 2.

$$|\psi_2\rangle = \mathrm{U_f}(|\psi_1\rangle) = \frac{1}{\sqrt{\mathrm{Q}}}\sum_{\mathrm{x=0}}^{\mathrm{Q-1}}|\mathrm{x, f(x)}\rangle \tag{19}$$

Where $\mathrm{f(x)} = a^x\ mod\ N$. Unluckily, the specific design of $\mathrm{U_f}$ is contingent on the specific a and N, which means we have to redesign or adjust the whole structure on register B every time a new set of (a, N) is given. This unsatisfactory fact is the bottleneck to Shor’s algorithm. But instead of getting bogged down in the specific structure of $\mathrm{U_f}$, nor the method to adjust the circuit, let’s just assume $\mathrm{U_f}$ has been constructed a priori.

Apply an IQFT to |ψ2⟩. So at position 3,

$$|\psi_3\rangle = U_{\mathrm{IQF}T^{-1}}(|\psi_2\rangle) = \frac{1}{\mathrm{Q}}\sum_{\mathrm{x=0}}^{\mathrm{Q-1}}\sum_{\mathrm{y=0}}^{\mathrm{Q-1}}\omega^{-xy}|\mathrm{y, f(x)}\rangle \tag{20}$$

Substitute z for f(x), where all possible z’s constitute a subset S of $Z_N^* = \{1, 2\ , \ldots, \mathrm{N}-1\}$. For simplicity, let z runs from 0 to N-1. If it falls into S, $\sum_{x;\mathrm{f}(x)=z}\omega^{-xy}$ is nonzero; otherwise, $\sum_{x;\mathrm{f}(x)=z}\omega^{-xy} = 0$.

$$|\psi_3\rangle = \frac{1}{\mathrm{Q}}\sum_{\mathrm{z=0}}^{N-1}\sum_{\mathrm{y=0}}^{\mathrm{Q-1}}\left(\sum_{x;\mathrm{f}(x)=z}\omega^{-xy}\right)|\mathrm{y, z}\rangle \tag{21}$$

The probability of observing the state |y, z⟩ is

$$P(|\mathrm{y, z}\rangle) = \left|\frac{1}{\mathrm{Q}}\sum_{x;\mathrm{f}(x)=z}\omega^{-xy}\right|^2 \tag{22}$$

$\mathrm{P(|y, z\rangle)}$ is the conditional probability of measuring state |y⟩ in the register A, given that |z⟩ in the register B was observed. It implies that |z⟩ must be measured before |y⟩. Then the wavefunction in register B collapsed from a superposition of all possible |z⟩ states into a certain |z⟩. Due to the entanglement established by U_f between register A and register B, the probability of measuring register A is consequently changed [10]. In reality, the circuit in Figure is connected to classical circuit at both ends, so we do not have to measure which state will |z⟩ will turn out to be in register B. Recall that p is the period of f(x), so x = x0 + np, where x0 is the first integer that makes f(x)=z, and n = 0, 1, 2, ...m – 1, where m is the upper bound for n. Since Q is usually much larger than p, m is roughly estimated as Q/p for every possible z falling into the set S.

$$P(|\mathrm{y, z}\rangle) = \left|\frac{1}{\mathrm{Q}}\sum_{x;\mathrm{f}(x)=z}\omega^{-xy}\right|^2 = \frac{1}{\mathrm{Q}^2}\left|\sum_{n=0}^{m-1}\omega^{-(x_0+np)y}\right|^2 = \frac{1}{\mathrm{Q}^2}\left(\frac{\sin\left(\frac{\mathrm{\pi mpy}}{\mathrm{Q}}\right)}{\sin\left(\frac{\mathrm{\pi py}}{\mathrm{Q}}\right)}\right)^2 \tag{23}$$

If y happens to make $\frac{py}{Q}$ close to an integer, namely $\omega^{\frac{py}{Q}}$ is close to 1, which leads to constructive interference, P r(|y, z⟩) is significantly larger than otherwise. Denote $\frac{py}{Q}$ as s, and consider

$$P(s) = \left(\frac{sin(m\pi s)}{m\ sin(\pi s)}\right)^2 \tag{24}$$

As is shown in Figure 7, it's better to make q sufficiently large. That's because as m ~ Q/p approaches to infinity, P(s) approaches to

$$\sum_{\mathrm{i}=-\infty}^{\mathrm{i}=\infty} \delta_{\mathrm{i,s}} \tag{25}$$

It becomes nearly impossible to find $\mathrm{s} = \frac{py}{Q}$ between any two adjacent peaks because s is not close to an integer then. Thus we can calculate $\mathrm{p} = \frac{Q}{\Delta y}$, where $\Delta \mathrm{y}$ is the minimal positive period of y measured in register A.

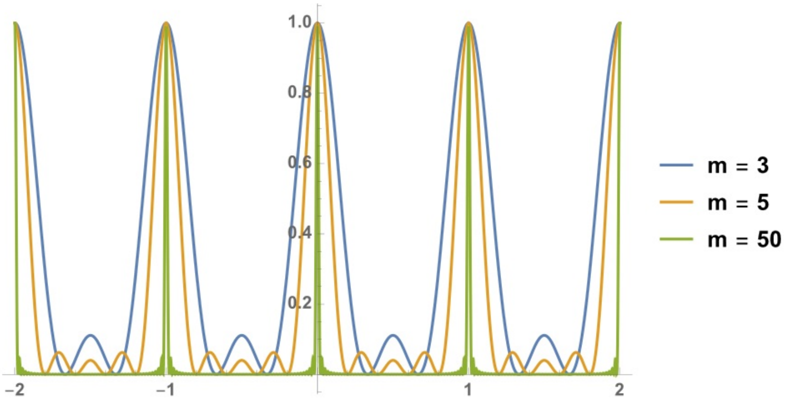

Figure 7 $P(\mathrm{z}) = \left(\frac{\sin(\mathrm{m}\pi\mathrm{s})}{\mathrm{m}\sin(\pi\mathrm{s})}\right)^2$

In summary, Uf gate selects the x that make f(x) =a certain z. They are the very x that are put in the IQFT. Note that x's are separated by a constant value p, and the IQFT detects the periods of the input states. Accordingly, the measurement of the output will reflect the periodicity of the input by showing yet another periodicity. We can utilize the period of the measurement in the output to deduce the period of the input.

4.3 The procedure of Shor's Algorithm

To wrap up, Shor's algorithm is composed of two parts: the classical subroutine and the quantum subroutine. The flowchart is summarized in Figure 8.

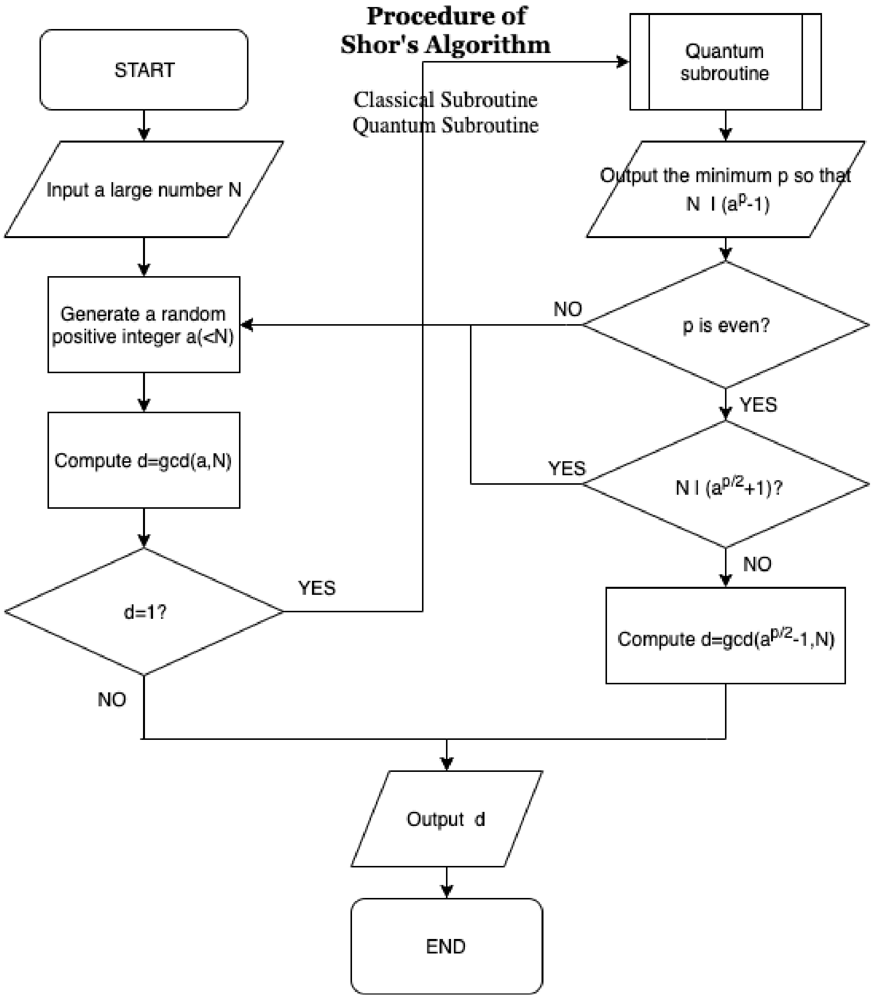

Figure 8 Procedure of Shor's Algorithm

5. Experimental Results [8]

5.1 Case 1: N=15, a=7 [11]

The circuit for Shor’s algorithm is designed as Figure 9. Qubits 0-2 form register A; qubits 3-6 form register B.

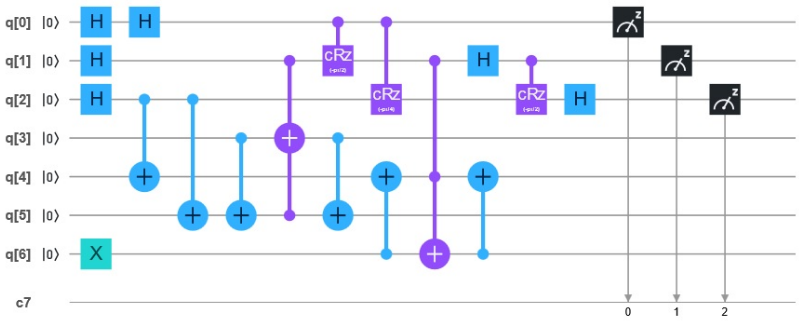

Figure 9 N=15, a=7, IQFT

Denote $x = [0.\mathrm{x}_1\mathrm{x}_2\mathrm{x}_3]$. $7^{\mathrm{x}} \equiv 7^{4\mathrm{x}_1} \cdot 7^{2\mathrm{x}_2} \cdot 7^{\mathrm{x}_3}$ mod N. Calculate.

$$\begin{cases} 7^4 \equiv 1 \bmod 15 \\ 7^2 \equiv 4 \bmod 15 \\ 7^1 \equiv 7 \bmod 15 \end{cases} \tag{26}$$

As a result, the first qubit q[0] does not make a difference in the output of U_f because whether or not q[0] is 0, register B remains the same. Consider the situation below:

x = 1 (q[0] = 0, q[1] = 0, q[2] = 1)
x = 2 (q[0] = 0, q[1] = 1, q[2] = 0)
x = 3 (q[0] = 0, q[1] = 1, q[2] = 1)

The outputs should be 1, 4, 7 for each, which match with the experiment results shown in Figure , Figure and Figure 12. Keep in mind that the output should be counted bottom up, which means the largest digit is at the bottom and the smallest digit is at the top, the reverse of the input.

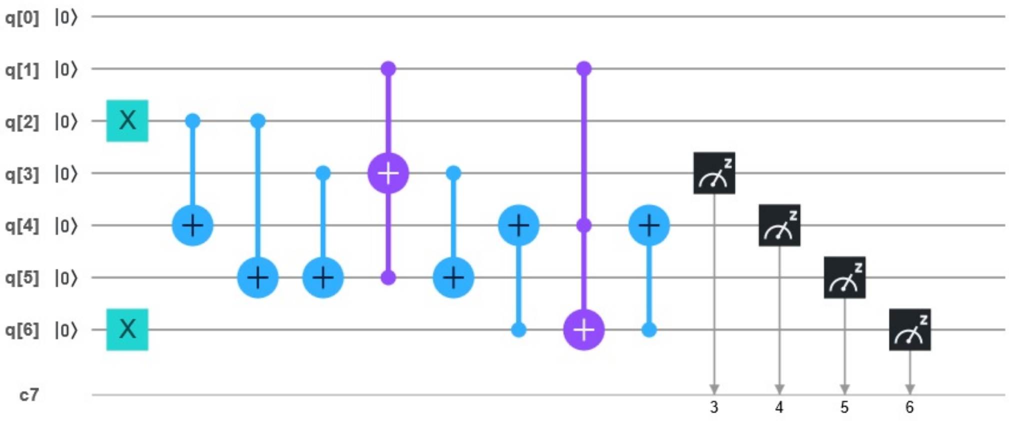

Figure 10 x=1

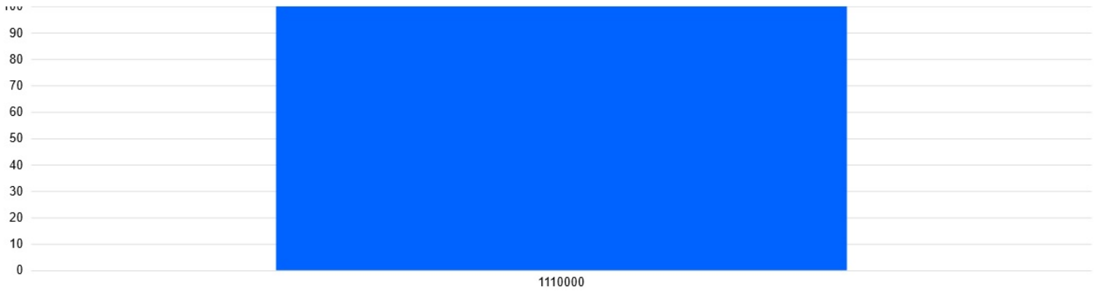

Figure 11 f(x) = 7

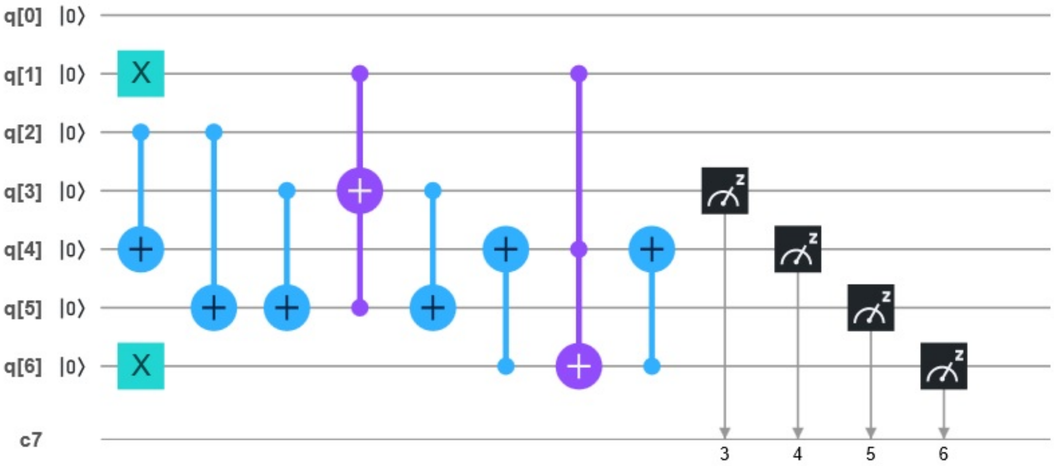

Figure 12 x = 2

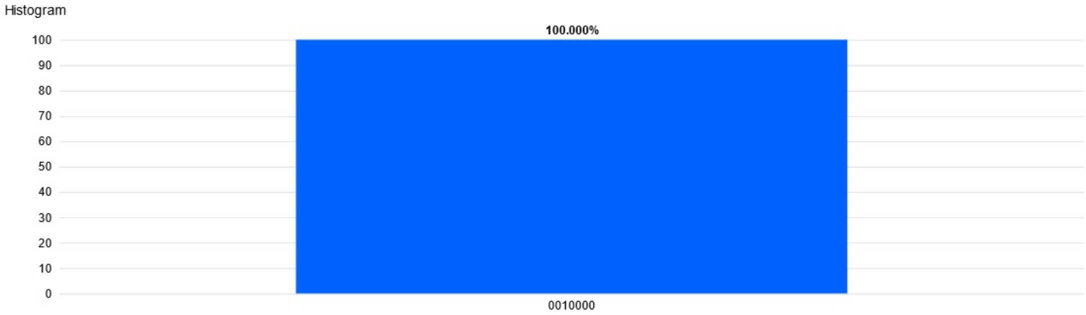

Figure 13 f(x) = 4

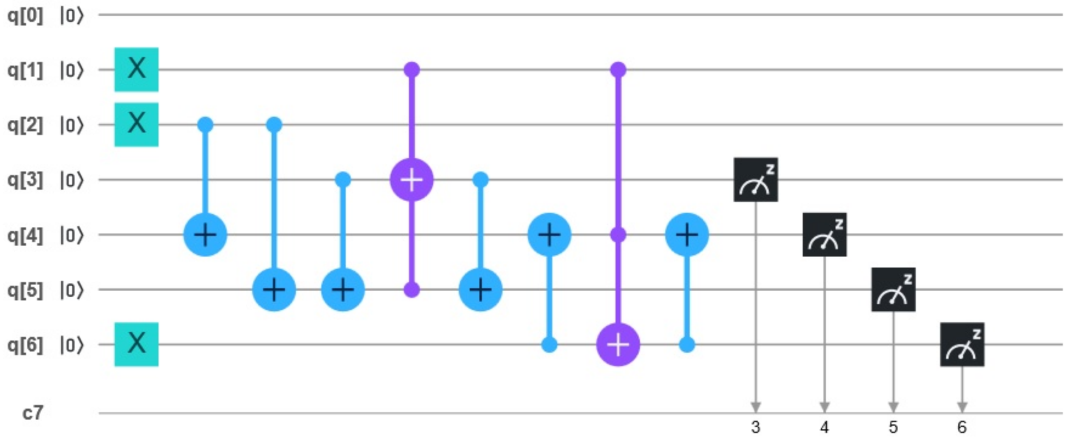

Figure 14 x = 3

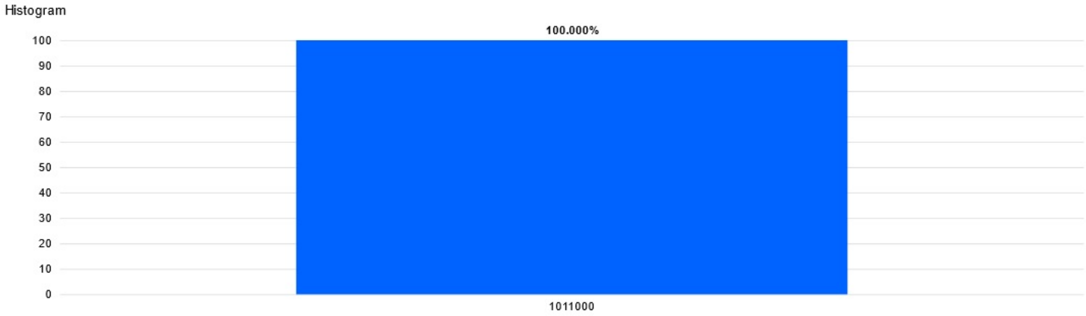

Figure 15 f(x) = 13

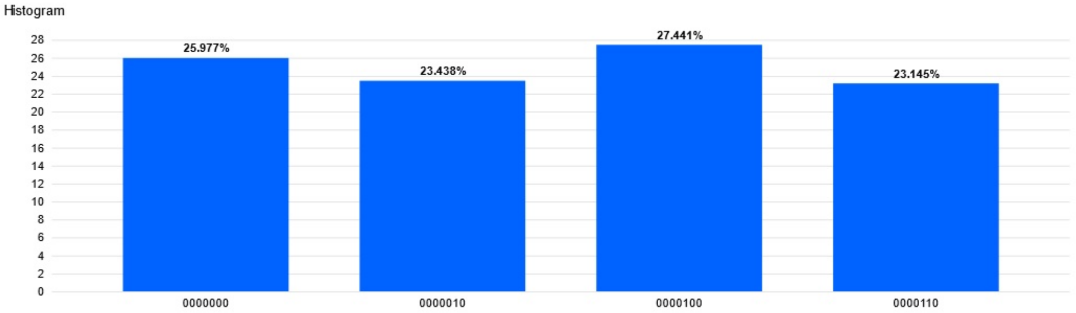

Figure 16 The period of y is 2

As is shown in Figure 16, $T_y = 2$, $Q = 8$, then $p = \frac{Q}{T_y} = 4$. $\gcd(7^{\frac{4}{2}} - 1, 15) = 3$, and $\gcd(7^{\frac{4}{2}} + 1, 15) = 5$. So 15=3 × 5.

Indeed, Building the $\mathrm{U_f}$ gate is one of the difficulties of realizing Shor's algorithm for different large integer N. There is no general $\mathrm{U_f}$ gate that can be applied universally, which means that the circuits above are limited to the case a = 7 and N = 15.

5.2 Case 2: N=15, a=11

In Figure , register A includes q[0-1], and register B includes q[2 - 4].

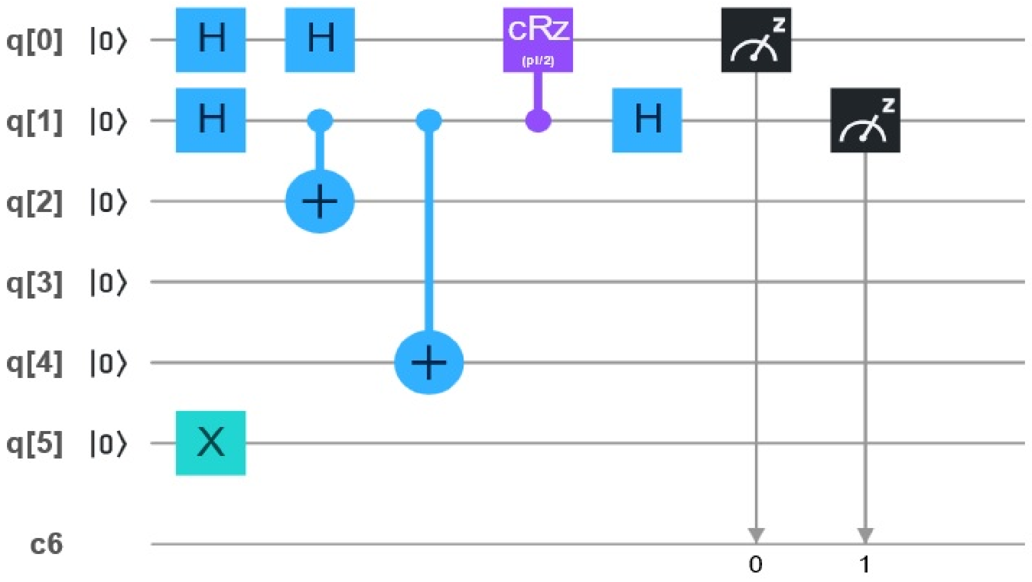

Figure 17 N=15, a=11

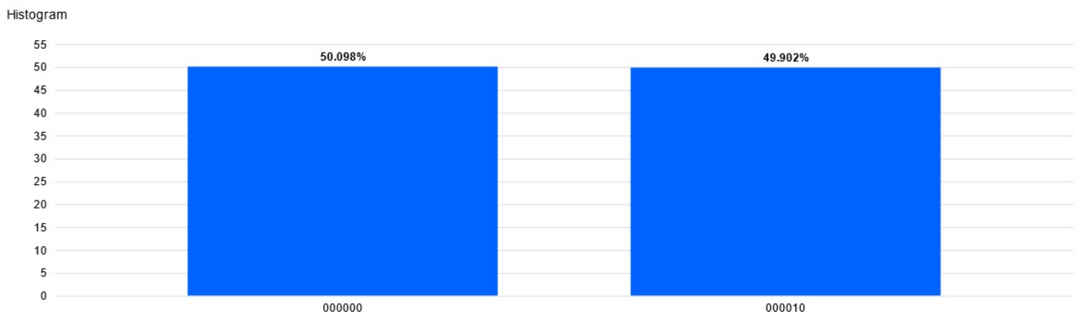

Figure 18 Output of a = 11, N=15

As is shown in Figure 18, $T_y = 2$, $Q = 8$, p=$\frac{\mathrm{Q}}{\mathrm{T_y}} = 4$, $\mathrm{T_y} = 2, Q = 8. p = \frac{\mathrm{Q}}{\mathrm{T_y}} = 4. \mathrm{gcd}(7^2 - 1,15) = 3$, and $\mathrm{gcd}(7^2 + 1,15) = 5$. So 15=3 $\times$ 5.

6. Conclusion

Based on quantum Fourier transform and modular exponentiation, Shor’s algorithm operated on a quantum computer is exceedingly efficient in factoring large integers than the classical algorithm. Despite the intricacy and annoyance of adjusting modular exponentiation gate (U_f) whenever a new set of (a, N) is fed in, Shor’s algorithm is feasible on quantum computers, at least for the case (a, N) = (7, 15) and case (a, N) = (11, 15), which are verified on IBM Q. It is promising that a general method of designing the modular exponentiation could be invented so that the circuit can be modified according to any given set of (a, N). If so, the classical method of encryption will be in jeopardy.

References

[1] G. Benenti, G. Casati and G. Strini, Principles of quantum computation and information

(Reprinted. ed.), World Scientific, 2004.

[2] M. Ettinger and P. Høyer, A quantum observable for the graph isomorphism problem, arXiv:quant-ph/9901029.

[3] D. Beckman, A. N. Chari, S. Devabhaktuni and J. Preskill, "Efficient Networks for Quantum Factoring," Physical Review A, p. 54 (2): 1034–1063, 1996.

[4] "Wolfram Mathworld," 23 October 2015. [Online]. Available: https://mathworld.wolfram.com/NumberFieldSieve.html.

[5] S. &. P. S. Burnett, The RSA security's official guide to cryptography, McGraw-Hill, Inc., 2001.

[6] N. D. Mermin, Quantum computer science: an introduction, Cambridge University Press, 2007.

[7] Trenar3, 14 February 2018. [Online]. Available: https://en.wikipedia.org/wiki/File:Q_fourier_nqubits.png.

[8] IBM, [Online]. Available: https://www.ibm.com/quantum-computing/.

[9] S. J. Lomonaco, Shor's quantum factoring algorithm. In Proceedings of Symposia in Applied Mathematics (Vol. 58, pp. 161-180), 2002.

[10] V. M. &. M. W. J. Kendon, "Entanglement and its role in Shor's algorithm," arXiv preprint quant-ph/0412140, 2004.

[11] L. M. S. M. B. G. Y. C. S. S. M. H. &. C. I. L. Vandersypen, "Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance," Nature, 414(6866), pp. 883--887, 2001.

[12] B. 2k14, 29 July 2014. [Online]. Available: https://commons.wikimedia.org/wiki/File:Shor%27s_algorithm.svg.