An analytical study of Shor's Algorithm

Aradhita Sharma
Electrical Engineering Department
Arizona State University
Tempe, Arizona
ashar314@asu.edu

Dr. Christian Arenz
Electrical Engineering Department
Arizona State University
Tempe, Arizona
carenz1@asu.edu

***Abstract*—This paper describes the implementation of Shor's algorithm, which aims to find out prime factors of an integer using quantum computing. This algorithm can easily break public key cryptography provided a sufficiently large quantum computer for computation. A comparison of factor finding algorithm in classical computation and quantum computation is being made. Qiskit platform is used for the demonstration of Shor's algorithm.**

I. Introduction

Factorising an integer into its prime factors is easy enough if the given integer is small. However, as the integer increases upto 20 digits or more, it would take unreasonable amount of time to find the prime factors. Means, the bigger the integer, the longer it takes to find the prime factors. That is why prime numbers are at the heart of the most difficult problems in number theory. RSA (Rivest–Shamir–Adleman), the public-key cryptosystem is based on the assumption that factoring large integers with greater digits is computationally intractable. RSA makes use of a public key which is the product of two large prime numbers. Factoring this public key can crack the RSA encryption, but factoring becomes increasingly time consuming as public key value grows large.

Peter Shor (Professor of Applied Mathematics at MIT) came up with a quantum algorithm known as Shor's algorithm which calculates the prime factors of a large number more efficiently than a classical computer in 1994 [1]. No classical algorithm is known that can factor integers in polynomial time. However, Shor's algorithm shows that factoring integers is efficient on an ideal quantum computer, and it is possible to defeat RSA by factoring integers in polynomial time given a sufficiently large quantum computer.

Quantum computer is a machine which makes the use of quantum physics properties for performing computations. These quantum mechanics laws are used to solve problems which are too complex to be solved by classical computers, one such problem includes the problem of factor finding, which is solved using Shor's algorithm. Factorisation of integer (N) using Shor's algorithm results in the time complexity of the order O(log N), which is the exponential speedup (also known as quantum speedup) compared to the most effective classical algorithm for factorization known (which is GNFS) [11].

II. Mathematics Background

This section covers basic mathematics terminologies which are used to explain the procedure of Shor's algorithm [2].

A. Greatest Common Divisor (GCD)

It is also known as Highest Common Factor (HCF). It represents the largest number which divides any two given numbers, that is common factor of both given numbers. For example, GCD (18, 24) = 6. Euclidean theorem is used for finding GCD.

B. The Euclidean Algorithm

This algorithm is used to find the GCD of two numbers. Let the given two numbers be A, B. Find the largest among these two numbers, let that be A. Therefore A can be written as

$$A = q_i * B + r_i$$

where q_i represents quotient and r_i represents the remainder for i = 1 to k, where $k^(th)$ step is when the remainder r_i obtained is zero. It is an iterative process untill remainder comes to be zero. And the last non-zero remainder is the required GCD (greatest common divisor). [3]

C. Modulo Operator

Modulo operator is an arithmetic operator which returns the remainder of an integer division. For example, 17 mod 5 = 2

D. Constraints on N

There are some limitations in finding the factors of the number N by Shor's algorithm. A number which satisfies all these conditions can be factorized by this Algorithm.

- The number should not be a prime number as prime numbers do not have any non trivial factors.
- The number should be an odd number, since even number already has 2 as one of its factors.
- The number should not be of the form r^p where r and p are positive integers greater than 1.

III. Quantum Mechanics Background

This section covers basic quantum terminologies which are used to explain the procedure of Shor's algorithm [5].

A. Qubits

The basic building block of a quantums system is called a qubit (quantum particle). It is analogous to bits represented in classical computing. For a two level quantum system, qubit can exist in two quantum states represented by $|0\rangle \, and \, |1\rangle$

B. Superposition

This is a quantum property which states that n quantum particles can exist in more than "n" states simultaneously. Superposition of quantum states can be achieved by the use of hadamard gate such that there is 0.5 probability to measure state $|0\rangle$ and 0.5 probability to measure state $|1\rangle$

$$\underset{\text{initial state}}{|0\rangle} \xrightarrow{H} (|0\rangle + |1\rangle)/2^{1/2}$$

$$\underset{\text{initial state}}{|1\rangle} \xrightarrow{H} (|0\rangle - |1\rangle)/2^{1/2}$$

C. Measurement

A measurement is the manipulation of a quantum system to yield a numerical result based on the probability values. A quantum system in N quantum states will collapse to one state while destroy all the rest states once measurement is performed. As example, measurement of a quantum system consisting of hadamard gate will result in either $|0\rangle$ state of $|1\rangle$ state with 0.5 probability.

$$\underset{\text{initial state}}{|0\rangle} \xrightarrow{H} \overset{0/1}{\boxed{\nearrow}} \rightarrow |0\rangle \, or \, |1\rangle$$

D. Quantum Fourier Transform

QFT is analogous to discrete Fourier transform, just that it is the linear transformation applied to quantum bits [7]. $|X\rangle = \Sigma_{i=0}^{N-1} x_i \, |i\rangle$ and maps it to quantum state $\Sigma_{i=0}^{N-1} y_i \, |i\rangle$

$$|k\rangle = QFT : |x\rangle = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n e^{2\pi i \frac{jk}{N}}$$

for all $\quad k = 0, 1, .., N-1$

This state equation can be obtained by applying hadamard gates and controlled rotation gates to the qubits whose matrix representation is given as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{and} \qquad R_m = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^m} \end{pmatrix}$$

where, controlled rotation gate means the rotation of 2^{nd} qubit is controlled by the input value of 1^{st} qubit.

QFT circuit for 2 qubits is represented by hadamard gate and there is no rotation performed. QFT circuit for 3 qubits and 4 qubits are shown below in figure 1 and figure 2 respectively.

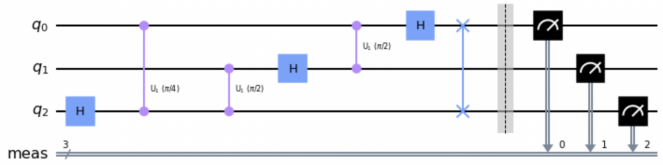

Fig. 1. QFT circuit for "3" qubits consisting of 3 hadamard gates for creating superposition of 3 qubits, and (1+2) controlled rotations.

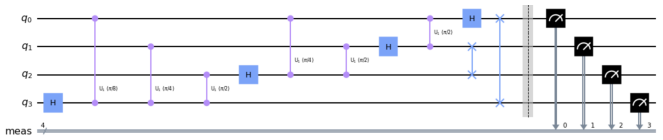

Fig. 2. QFT circuit for "4" qubits consisting of 4 hadamard gates for creating superposition of 4 qubits, and (1+2+3) controlled rotations.

Qubits are represented upside down for better convenience, which needs swapping at the end of qubit rotations which is performed with the help of swap gates. Figure 1 and Figure 2 includes the measurement operation performed in the quantum circuit, where the measured value is stored in classical registers. Classical Fourier transform (FFT) algorithm requires $NlogN = n2^n$ computational steps whereas $O(n^2$ gates are required for n-qubit QFT implementation evaluating it in polynomial time [14].

IV. Procedure

Finding factors of an integer N means finding another integer "p" between 1 and N that divides N. Shor's algorithm used to find the factor consists of two parts :

- Find the order of the integer using classical computation.
- Solve the order-finding problem using quantum computation.

These two parts are discussed as follows.

A. Convert factoring into order finding problem

The first step in Shor's algorithm is to reduce the factoring problem to order finding problem. Assume that N integer is not even. Steps to reduce this problem are :

1) Picking a random number A such that 1¡A¡N
2) Compute the greatest common divisor of A, N using Euclidean algorithm. Let K = GCD (A,N)
3) If K $\neq$ 1, K is the GCD of N, then skip the next step.
4) If K = 1, find "r" such that

$$f(x) = A^x mod N$$

such that $f(A) = f(A + r)$
Define a new variable Q = 1 and find $Q * AmodN$ and set value of Q = remainder obtained and repeat calculating $Q * AmodN$ till the remainder obtained is 1. Here, number of repetitions (period) is the value "r" such that $f(A) = f(A + r)$

5) If "r" obtained is odd, or $A^{r/2} = -1 mod N$ choose a different number A.
6) Otherwise, both $GCD(A^{r/2}+1, N)$ or $GCD(A^{r/2}-1, N)$ are nontrivial factors of N [3].

For example: For given N = 15, let A = 7 and r = 4.

$$GCD(7^{4/2}+1, 15) = GCD(50, 15) = 5$$
$$GCD(7^{4/2}-1, 15) = GCD(48, 15) = 3$$

such that, N is the product of two distinct primes (3,5),

$$N = 15 = 3x5$$

B. Solve the order finding problem

Quantum computing is used to find the order (period) value "r" of $f(x) = A^x mod N$ such that $f(A) = f(A+r)$ The order is the smallest number such that

$$A^r mod N = 1$$

then,

$$(A^r - 1) mod N = 0$$

which means N must divide $(A^r - 1)$ and this can be written as

$$A^r - 1 = (A^{r/2} - 1)(A^{r/2} + 1)$$

There is a high probability that greatest common divisor $GCD(A^{r/2}+1, N)$ or $GCD(A^{r/2}-1, N)$ is a factor of N.

Period (Order) finding algorithm relies on the superposition property of quantum states. Because of this property, to compute the period of a function f, function can be evaluated at all points simultaneously. Fourier transforms are used to find periodicity of signals, similarly, quantum Fourier transform (QFT) can be used to find period of a function. QFT transforms the superposition to another state which yeilds the period of function with highest probability as the output. Once measurement is performed, it yield only one of all possible values (maximum times the highest probability value) and destroys all the other states. Hence, period ("r") such that $f(A) = f(A+r)$ is obtained using quantum Fourier transform which results in exponential speedup.

V. Comparison of classical vs quantum algorithm

Shor's algorithm and any classical algorithm for factorisation have the first step to convert factoring problem into order finding problem in common. However, for finding the order (period), quantum approach is different from classical approach.

Classically, the algorithm will run "r" times to calculate $f(x) = A^x mod N$ for x = 0,1,2..m till the period "r" is obtained such that $A^r mod N = 1$. However, with the help of quantum computing, using quantum Fourier transform, period can be computed in a single step. In QFT, this x acts as quantum states which exist in superposition of 0 to $2^n - 1$ states, where "n" is the number of qubits, and the amplitudes associated with these quantum states (also known as wavefunctions) are in such a way that sum of amplitude square of all quantum states represents the probability and is equal to 1. When implementing the Shor's algorithm, the wavefunction associated with the period of function gives the highest probability.

Therefore, with the help of QFT, the period of $f(x) = A^x mod N$ can be extracted from O(1) measurements instead of $O(r) = O(2^n)$ in case of classical computing. Complexity of algorithms is described in terms of the size of the representation of the numbers (2^n for n qubits). n-qubit implementation of QFT requires $O(n^2)$ gates evaluating it in polynomial time, this QFT is implemented n times for having random A value, making total $O(n^3)$ steps. Hence, Shor's algorithm factor the integer in $O(n^3)$, polynomial time, which no classical algorithm can [15].

A. Comaprison : GNFS vs Shor's

GNFS (General number field sieve) is a classical approach to find factors of a large integer N. This approach came to existence in 1996, and successfully calculated the factors of 130 digit RSA challenge number. A research was performed to compare GNFS and Shor's algorithm for N (100 digits) as input to be [8] :
2881039827457895971881627053137530734638790825166127496066674320241571446494762386620442953820735453
Output factors:
p (45 digits) =
618162834186865969389336374155487198277265679
q (55 digits) =
4660648728983566373964395375209529291596595400646068307

GNFS consists of following steps :

- Selecting the polynomial.
- Field sieving.
- Process the relations.
- Solve matrices to find dependencies among relations.
- Square root to compute the final factorization.

Function	Average CPU Utilization (%)	Average core temperature (°C)	Running time (hours)
Polynomial	25	65	0.40
Sieving	100	82	1.33
Relations	100	82	0.02
Matrix	100	82	0.06
Square root	100	82	0.01

Fig. 3. Running time of GNFS python software algorithm [8]

From the tables above, it is seen that GNFS takes a total of more than 1 hour running time to calculate the factors, whereas

Clock Speed	Approximate Time
1 GHz	< 1 *second*
1 MHz	≈120 *seconds*
1 KHz	≈15 *days*
1 Hz	≈10 *years*

Fig. 4. Approximate Running time of Shor's algorithm in factoring 375 bit integer [8], [9]

it is approximated that shor's algorithm can factor in less than 1 second time if given clock speed is 1GHz.

VI. QISKit Simulation Results

QISKit is a python library which is provided by IBM which works as a quantum simulator.For comparing classical approach and quantum approach, python programming language is used. Inverse QFT circuit acts as phase estimation algorithm, for finding the period of the function. Controlled multiplication by A mod N is constructed as a quantum circuit and quantum phase is estimated using this controlled multiplication and inverse QFT circuit. Here, "Aer simulator" provided by QISKit is used as backend (which represents quantum simulator and show results similar to the results obtained on real quantum device). This quantum circuit measurement results in quantum state with maximum probability which corresponds to the phase of the function. With the help of this phase information, period of function is calculated, which gives the prime factors of the function (number).

For this study, shor's algorithm is performed for calculating the factors of number N = 15. Result obtained is shown in Figure 6, where it is seen that classical computation takes time in the order of nanoseconds, whereas quantum computation takes time of 2.1 seconds. Clearly Shor's algorithm is taking more time, but here, N value is very less. Shor's algorithm will be taking same amount of time for N value consisting of more than 100 digits, whereas the computation time for classical method will increase drastically.

A tabular comparison has been made to observe the results of Shor's algorithm for different values of "A"

S.No.	Value of A	period "r"	Execution time (s)
1	4	2	1.171
2	7	2	2.107
3	8	4	2.815
4	11	2	2.157
5	13	4	2.181

From the table, approximate same execution time is observed for all possible values of A to find period (r) and then find the factors of N = 15. Hence, any random value of A will lead

```
Classical Computation
factors obtained [3, 5]
execution time 0.00049591064453125

Quantum Computation
Random A =  7

Attempt 1:
Result: r = 4
*** factor found: 3 ***
*** factor found: 5 ***
quantum execution time 2.1074957847595215
```

Fig. 5. Classical and Quantum computation results obtained for N = 15

the same result and will take same amount of time to calculate the prime factors.

VII. Conclusion

Total time complexity for finding factors using fastest classical algorithm (GNFS) is $O(e^{\sqrt{\frac{64}{9}}n^{1/3}(logn)^{2/3}})$ [16] which takes exponential time to find the factors of number N (represented by 2^n) whereas Shor's algorithm takes polynomial time $O(n^3)$ which is now improved to $O(n^2 logn)$ to find the factors [15].

Generally RSA keys are 1024-4096 bits long, so considering a 1024 bits (n qubits) long number (about 308 digits long [12]), classical approach of finding the prime factors of such a large number will take

$$O(e^{\sqrt{\frac{64}{9}}n^{1/3}(logn)^{2/3}}) = O(e^{\sqrt{\frac{64}{9}}1024^{1/3}(log1024)^{2/3}})$$
$$\mathrm{O}(e^{2.67*10.07*(10)^{2/3}}) = O(e^{2.67*10.07*4.64})$$
$$\mathrm{O}(e^{124.7}) = O(e^{124.7}) = O(3.45 * 10^{37})$$

operations which is very large, whereas on the contrary, quantum approach with Shor's algorithm will take

$$O(n^3) = O(1024^3) = 1073741824$$

operations, which is very less compared to the classical computing method. Hence, for the operation which could take years to compute with classical approach, quantum computing can compute within a day. It is seen that Shor's algorithm can factorise a given number in quantum polynomial time, this could break the RSA cryptosystem, most reliable technique of modern cryptography. Because of this possibility, Shor's algorithm pose a potential threat to most modern encryption techniques. However, this opens up a whole new domain of encryption techniques with the help of quantum computing, known as quantum cryptography, which can help in building more strong and unbreakable encryption schemes [10].

VIII. Code

Here is a link to the python code written in google colab platform, showing the simulation results of Shor's algorithm and classical algorithm to find out the prime factors of a given number.
Shor's algorithm implementation on Google Colab link

References

[1] Monz T, Nigg D, Martinez EA, Brandl MF, Schindler P, Rines R, Wang SX, Chuang IL, Blatt R. Realization of a scalable Shor algorithm. Science. 2016 Mar 4;351(6277):1068-70. doi: 10.1126/science.aad9480. PMID: 26941315.

[2] Medium.com, Shors factoring algorithm, https://kaustubhrakhade.medium.com/shors-factoring-algorithm-94a0796a13b1

[3] Medium.com, The euclidean algorithm, https://medium.com/i-math/the-euclidean-algorithm-631d7ddf2382

[4] Wikipedia.org, Shor's Algorithm, https://en.wikipedia.org/wiki/Shor%27s_algorithm

[5] Nielsen, M.A. Chuang, I.L., 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press.

[6] Qiskit.org, Quantum Fourier transform https://qiskit.org/textbook/ch-algorithms/quantum-fourier-transform.html

[7] Wikipedia.org, Quantum Fourier transform https://en.wikipedia.org/wiki/Quantum_Fourier_transform

[8] S. M. Hamdi, S. T. Zuhori, F. Mahmud and B. Pal, "A Compare between Shor's quantum factoring algorithm and General Number Field Sieve," 2014 International Conference on Electrical Engineering and Information Communication Technology, 2014, pp. 1-6, doi: 10.1109/ICEEICT.2014.6919115.

[9] R. Meter, K. Itoh, T. Ladd, "Architecture Dependent ExecutionTime of Shor's Algorithm", arXiv:quant-ph/0507023v2, May 25, 2006.

[10] caltech.edu, Quantum science explained quantum cryptography https://scienceexchange.caltech.edu/topics/quantum-science-explained/quantum-cryptography

[11] Huawei HiQ, Using Shor's Algorithm to Achieve Factor Decomposition, https://hiqsimulator.readthedocs.io/en/latest/examples/examples.ShorAlgorithm.html#:~:text=The%20time%20complexity%20of%20Shor's,effective%20classical%20factorization%20algorithm%20known.

[12] blog.1password.com, Large even prime number discovered, https://blog.1password.com/large-even-prime-number-discovered/#:~:text=The%20prime%20numbers%20used%20in,(about%20616%20digit)%20products.

[13] Damian Musk. A Comparison of Quantum and Traditional Fourier Transform Computations. Authorea. November 23, 2020. DOI: 10.22541/au.160614804.47667838/v1

[14] Yehuda B. Band, Yshai Avishai, Quantum Fourier Transform in Quantum Mechanics with Applications to Nanotechnology and Information Science, 2013

[15] National Academies of Sciences, Engineering, and Medicine. 2019. Quantum Computing: Progress and Prospects. Washington, DC: The National Academies Press. https://doi.org/10.17226/25196.

[16] wikipedia.org, General number field sieve, https://en.wikipedia.org/wiki/General_number_field_sieve