CS406 Course Project

Shor's and other interesting Quantum Algorithms

Dhananjay Raman
210050044

Tanay Vineet Tayal
210050155

3 May 2024

Contents

1 Problem Description

Many encryption systems, like the RSA scheme, which is one of the cornerstones of modern cryptography, rely on the difficulty of prime factorization of large numbers into their prime components [1]. The security of these systems is based on the assumption that no efficient classical algorithm exists for factorization. The best classical algorithms for factoring large numbers grow sub-exponentially with the size of the numbers, making it computationally impractical for very large numbers [2]. This computational challenge is crucial in maintaining the security of modern cryptography.
Quantum computers are different from classical computers, utilizing quantum mechanical principles like superposition and entanglement. Quantum computers have the potential to solve certain problems much more efficiently (even exponential speedup is possible). **Shor's Algorithm** [3] poses a theoretical threat to the foundation of current cryptographic practices by promising an efficient method for factoring large numbers.

2 Shor's Algorithm

2.1 Overview

Shor's Algorithm is a quantum algorithm for integer factorization. It is a polynomial-time quantum algorithm that can factorize an integer N into its prime factors in $O((\log N)^3)$ time. The algorithm consists of two main parts:

1. **Classical Reduction**: Reduce the problem of factoring N to the problem of finding the period of a function.

2. **Quantum Subroutine**: Use a quantum subroutine to find the period of a function.

At the heart of Shor's Algorithm we have the problem of finding the period of a function. The specific function used in Shor's Algorithm is the **Modular Exponentiation Function**:

$$f(x) = a^x \mod N$$

where a is a randomly chosen integer between 1 and $N-1$ and N is the number to be factorized. The *order* or *period* of the function $f(x)$ is the smallest positive integer r such that $a^r \mod N = 1$. The period of the function $f(x)$ is crucial in the factorization of N. It has the following properties:

1. $f(x+r) = f(x) \quad \forall x \in \mathbb{Z}$

2. If r is even, then $a^r - 1$ can be factored as $(a^{r/2}-1)(a^{r/2}+1)$, and since N divides $a^r - 1$, N shares common factors with $(a^{r/2}-1)$ and $(a^{r/2}+1)$.

The algorithm is probabilistic and requires a quantum computer to perform the quantum subroutine. The algorithm has been proven to run in polynomial time on a quantum computer, but the practical implementation of the algorithm is still a challenge due to the error rates in quantum computers.
Note: For demonstration (and implementation) purposes, we describe the quantum circuit for a small number $N = 15$.

2.2 Classical Reduction

The classical reduction part of Shor's Algorithm involves reducing the problem of factoring N to the problem of finding the period of a function. The steps involved in the classical reduction are as follows:

1. Choose a random integer a between 1 and $N-1$.

2. Compute the greatest common divisor of a and N. If $1 < \gcd(a, N) < N$, then the greatest common divisor is a non-trivial factor of N.

3. If the greatest common divisor is a non-trivial factor of N, then the factorization is complete, and the algorithm returns the factors.

4. If the greatest common divisor is 1, then proceed to the quantum subroutine to find the period of the function $f(x) = a^x \mod N$.

5. If the period of the function is even, then compute $g = \gcd(a^{r/2} + 1, N)$. If g is non-trivial, then the algorithm returns the factors g and N/g.

6. If the period of the function is odd or g is trivial, then repeat the process with a different random integer a.

The pseudocode for the classical reduction part of Shor's Algorithm is as follows:

```
def shor(N):
    while True:
        a = random.randint(1, N-1)
        g = gcd(a, N)
        if 1 < g < N:
            return g, N//g
        if g == N:
            continue
        r = find_period(a, N)
        if r % 2 == 0:
            g = gcd(a**(r//2)+1, N)
            if 1 < g < N:
                return g, N//g
```

Listing 1: Core Algorithm

2.3 Quantum Subroutine

The quantum subroutine part of Shor's Algorithm involves finding the period of the function $f(x) = a^x \mod N$. The quantum subroutine uses the Quantum Fourier Transform (QFT) to find the period of the function.

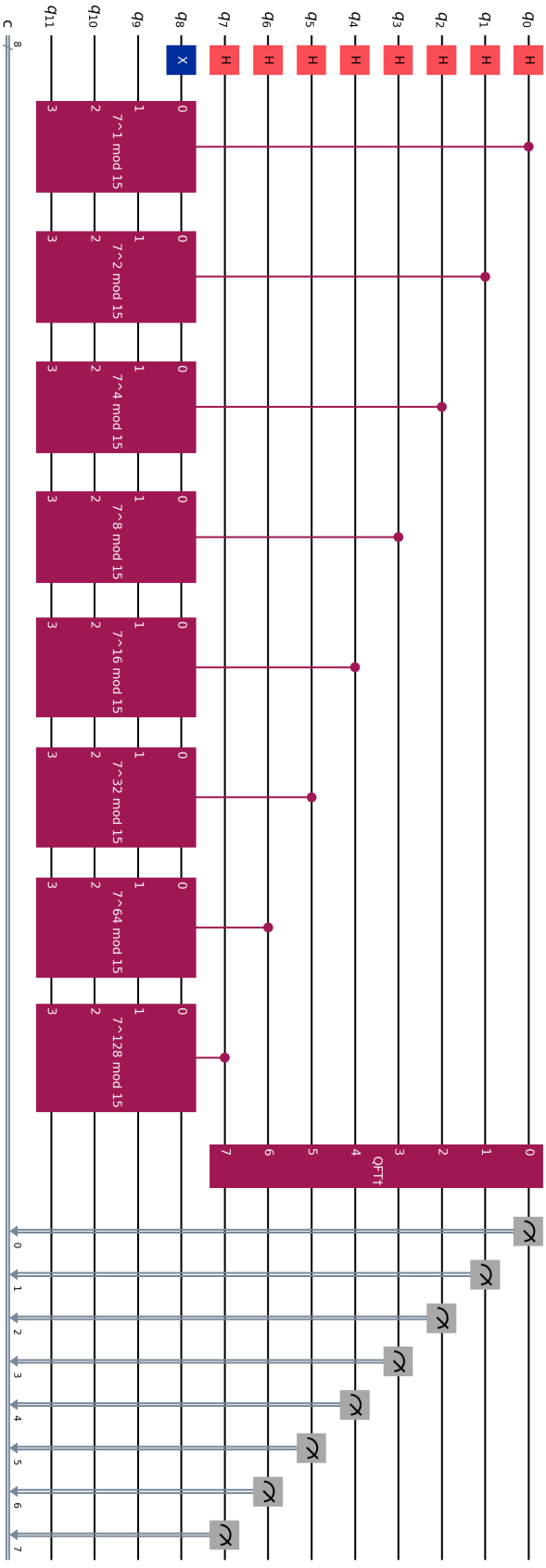

Figure 1: Quantum Circuit for Shor's Algorithm (for $N = 15$) in Qiskit [4].

The main steps involved in the quantum subroutine are as follows:

1. **Register Initialization**: We initialize three registers:
   (a) A register of size l qubits (`q0 - q7`) which will serve as the input to the Quantum Fourier Transform. These qubits are prepared in the superposition of all possible states using Hadamard (H) gates. Here l should be chosen such that $2^l > N^2$, so that the period can be used to accurately determine the factors of N.
   (b) A register of size n qubits (`q8 - q11`) which will be used to perform the modular exponentiation. Here, $n = \log_2 N$ is the number of bits required to represent N.
   (c) A third classical register (`c`) to store the results of the measurements.

2. **Modular Exponentiation**: The sequence of blocks labeled with $f(x) = 7^{2^i} \mod 15$ (where i ranges from 0 to 7) represent the modular exponentiation operation. The modular exponentiation is performed by applying controlled-U gates, where U is the unitary operator for the function $f(x) = a^x \mod N$. The unitary operator U is defined as:
   $$U|y\rangle|0\rangle = |y\rangle|a^y \mod N\rangle$$
   This operation is controlled by the qubits in the first register and the result is stored in the second register, and creates a highly entangled state where the second register contains the values of $a^x \mod N$ for all x, x being the state represented by the qubits `q0 - q7` in the first register.

3. **Quantum Fourier Transform**: The Quantum Fourier Transform is applied to the first register to find the period of the function. The Quantum Fourier Transform is a quantum analogue of the Discrete Fourier Transform and is used to find the period of a function in superposition. The Quantum Fourier Transform is implemented using a series of Hadamard gates and controlled-phase gates.

4. **Measurement**: The first register is measured, i.e. the qubits `q0 - q7` are measured, and the result is stored in the classical register `c`.

5. **Period Extraction**: The final measurement (of register `c`) gives an 8-bit string. This 8-bit string is converted from binary and divided by $2^8 = 256$ to get the phase as a fraction s/r, where s is a random integer between 0 and $r - 1$ and r is the period/order of the function. Thus after rationalizing the fraction, the denominator r gives a "guess" for the period r of the function, and is returned to the classical reduction part of the algorithm.

2.4 Code Implementation

The code for the quantum subroutine of Shor's Algorithm for $N = 15$ is implemented using Qiskit, an open-source quantum computing framework [4]. All the code for the quantum subroutine is available in the Jupyter Notebook `shors.ipynb` in the repository. The code can be run on a quantum simulator or a real quantum computer using IBM Quantum Experience. Link for the repository: `https://github.com/Seraphsnow/CryptoProject-`

2.5 Caveats

1. **Probabilistic**: The algorithm may require multiple runs to find the correct factors.

2. **Requirement of a large Quantum Computer**: The algorithm requires a large number of qubits to factor numbers of a practical size (such as 2048-bit numbers used in RSA). The current state of quantum computers is not sufficient to factor such large numbers.

3. **Error Rates and Quantum Decoherence**: Quantum Computers are prone to errors due to noise and decoherence (loss of quantum information from interaction of system with environment), which can affect the fidelity (amount of information preserved) of the quantum computation. The quantum gates must operate with very high precision, and maintaining coherence for the required number of qubits over the time is a significant challenge.

2.6 Variants

1. **Shor's Algorithm for the Discrete Log Problem**: Shor's Algorithm can also be used to solve the discrete logarithm problem [5], which is the basis of many cryptographic schemes like the Diffie-Hellman key exchange and the Digital Signature Algorithm. The algorithm works by finding the period of a function similar to the modular exponentiation function used in integer factorization:

$$f(x) = g^x \mod p$$

   where g is a generator of the group and p is a prime number. The period of the function is used to find the discrete logarithm of g with respect to p. Again, we reduce the problem to finding the period of the function and use the Quantum Fourier Transform/Phase Estimation to find the period.

2. **Generalization to the Hidden Subgroup Problem**: Shor's Algorithm can be generalized to solve the Hidden Subgroup Problem, which is a generalization of problems like the discrete logarithm problem and the integer factorization problem. The Hidden Subgroup Problem is a computational problem that arises in the study of quantum algorithms and has applications in cryptography and number theory.

   Given a group G and a function $f : G \to X$, the goal is to find the hidden subgroup H of G such that f is constant on the cosets of H. For any finite *abelian* group G, the Hidden Subgroup Problem can be solved efficiently using quantum algorithms based on Shor's Algorithm [6].

3 Other Quantum Algorithms

This section describes some other interesting quantum algorithms relevant to cryptography, mentioned here to generate interest in the reader.

3.1 Grover's Algorithm

Grover's Algorithm is a quantum algorithm for searching an unsorted database [7]. The algorithm provides a **quadratic speedup** over the best classical algorithms for the same problem. The algorithm has applications in searching, optimization, and cryptography. The algorithm works by iteratively applying a sequence of quantum operations to amplify the probability of finding the correct solution. The algorithm has been proven to run in $O(\sqrt{N})$ time on a quantum computer, where N is the size of the database.
Grover's Algorithm can be used to solve the problem of finding the pre-image of a hash function in $O(\sqrt{N})$ time, where N is the size of the hash function's output. This has implications for cryptographic hash functions and digital signatures, as it effectively reduces the security of n-bit hash functions to $n/2$ bits in the quantum setting.

3.2 Quantum Key Distribution (BB84 Protocol)

Quantum Key Distribution (QKD) is a secure communication method that uses quantum mechanics to secure a communication channel. The BB84 Protocol is a quantum key distribution protocol that allows two parties to securely establish a shared secret key [8]. The security of the protocol is based on the principles of quantum mechanics, such as the no-cloning theorem and the uncertainty principle. The protocol uses the properties of quantum states to detect eavesdropping and ensure the security of the communication channel.
The BB84 Protocol is used in quantum cryptography to secure communication channels and is the basis for many quantum key distribution systems. The protocol has been implemented in various quantum communication systems and has been shown to be secure against various types of attacks.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21, 2 (Feb. 1978), 120-126. https://doi.org/10.1145/359340.359342

[2] Buhler, J.P., Lenstra, H.W., Pomerance, C. (1993). Factoring integers with the number field sieve. In: Lenstra, A.K., Lenstra, H.W. (eds) The development of the number field sieve. Lecture Notes in Mathematics, vol 1554. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0091539

[3] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 1994, pp. 124-134, doi: 10.1109/SFCS.1994.365700.

[4] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, ... Christa Zoufal.

(2019). Qiskit: An Open-source Framework for Quantum Computing (0.7.2). Zenodo. https://doi.org/10.5281/zenodo.2562111

[5] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Comput. 26, 5 (Oct. 1997), 1484–1509. https://doi.org/10.1137/S0097539795293172

[6] Michael A. Nielsen and Isaac L. Chuang. 2010. Quantum Computation and Quantum Information: 10th Anniversary Edition (10th ed.). Cambridge University Press, New York, NY, USA.

[7] L. K. Grover, "A Fast quantum mechanical algorithm for database search," [arXiv:quant-ph/9605043 [quant-ph]].

[8] Bennett, Charles & Brassard, Gilles. (1984). WITHDRAWN: Quantum cryptography: Public key distribution and coin tossing. Theoretical Computer Science - TCS. 560. 175-179. 10.1016/j.tcs.2011.08.039.