To: Professor Alex Matos Abiague

From: Aimen Al-Doais, Ayman Elfayoumi, and Adham Elmadawy

Date: Dec 11 2024

Subject: Implementation of Shor’s algorithm on a Quantum Computer

### What is Shor's Algorithm?

Shor’s algorithm is a quantum algorithm designed to find the prime factors of a number. It achieves this in polynomial time, whereas classical computers require exponential time for the same task when the number is large.

### The prime factorization problem and its complexity class when executed on a classical computer, and how can it be used for encryption?

Before understanding the significance of Shor’s algorithm and how it works, one must first understand the problem that the algorithm is designed to solve. The prime factorization problem describes the mathematical problem where one must compute the prime numbers that when multiplied together, result in the given number. This problem is incredibly tedious for classical computers to solve. Its tediousness is the reason why the problem forms the basis of the modern asymmetric cryptography used to keep sensitive information safe.

The most used asymmetric cryptography algorithm - Rivest, Shamir, and Adleman (RSA) uses a public and private key pair for someone to securely send an encrypted message to another person. The public key is used by the sender to encrypt the message so that it is unreadable. Only the recipient, who possesses the private key, can decrypt the message. RSA cryptography leverages the prime factorization problem and sets a mathematical relationship between a public and private key pair and a number *n*, such that the only way to break the encryption would be to find the prime factors of *n* (*n* = *p\*q*). For RSA encryption to work, *n* must be a composite number, which can only be factored into two prime numbers *p* and *q*. No other numbers besides *n* and 1 can multiply together to get *n*. Given *n*, the only way to find *p* and *q* is to repeatedly guess a number, divide n by that number, and see if the remainder is 0. *p* is only found when the remainder is 0, and *q* can be immediately found once *p* is found. For RSA to be helpful and resistant to brute-force attacks, *n* must be a very large number, recommended to be in the order of 2^2048 (Barker, Dang 2015). This ensures that the number of guesses required to find *p* is impractical for a conventional computer to do in a reasonable amount of time. To date, the most efficient known algorithm to find the prime factors of *n* using a conventional computer, is the General Number Field Sieve (GNFS) algorithm. Despite being the fastest known algorithm, GNFS is in the complexity class of the following for large numbers *n*:

O()

Plugging in 2^2048 for *n*, this means that it would still take around 10^43.5 operations to find *p* and *q*. This means that without unfathomably good luck, it is essentially impossible to break RSA encryption using the fastest conventional computer and algorithm.

### The mathematics behind Shor’s algorithm.

1. **Classical Setup: Converting Factoring Into a Period-Finding Problem**
   * Begin by picking a number **a** such that 1 < **a** < N and gcd(a, N) = 1
   * If gcd(a, N) > 1, you’ve already found a **non-trivial factor** of NNN, solving the problem. Otherwise, proceed.
   * The modular arithmetic function:
     1. f(x) = x^a mod N
   * Where
     1. f(x + r) ≡ f(x)
   * Which implies
     1. x^r ≡ 1 mod N
2. **Using the Quantum Fourier Transform to Find the Period r**
   * This is the only step that requires a **quantum computer**.
   * Prepare a quantum state encoding the function f(x) in a superposition of all possible inputs x.
   * Apply the Inverse Quantum Fourier Transform (QFT) to this state. The QFT uses Hadamard gates for superposition and quantum interference to isolate the period r with high probability.
   * Measure the resulting state, which provides a multiple of 1/r. Then we can use classical post-processing (e.g., continued fractions) to deduce r.
3. **Classical Post-Processing: Finding the Factors Using r**
   * If r is odd or if x^r/2 ≡ -1 mod N, return to step 1 and use a different a
   * Otherwise,
     1. x^r ≡ 1 mod N => (x^(r/2) + 1)(x^(r/2) - 1) ≡ 0 mod N
   * Then use GCD to find factors
     1. gcd(x^(r/2) - 1, N) and or gcd(x^(r/2) + 1, N)
   * As long as (x^(r/2) + 1) or (x^(r/2) - 1) are not an integer multiple of N, then one of them should have a non trivial factor in common of N

**How Are Quantum Computing/Mechanics Used in Shor's Problem?**

Quantum computing is utilized during the Quantum Fourier Transform (QFT) stage of the algorithm to find the period of a modular arithmetic function. This step leverages quantum mechanics, specifically superposition (creating a state with many possibilities simultaneously) and interference (amplifying correct results while canceling incorrect ones).

**The advantage of the quantum algorithm for solving the prime factorization problem.**

The biggest advantage for solving the prime factorization problem is that a quantum computer is able to solve the problem in polynomial time (O[N^3]) compared to a classical computer which takes an exponential time (O[exp (L1/3(log L)2/3)]).

### What is the relevance of Shor’s algorithm?

1. **Quantum Advantage:**  
   Shor’s algorithm demonstrates how quantum computers can solve certain problems—such as integer factorization—exponentially faster than classical computers. This highlights the transformative potential of quantum computing.
2. **Impact on Data Security**:  
   Modern cryptographic systems, like RSA, rely on the computational difficulty of factoring large integers. Shor’s algorithm threatens these systems, making it vital to develop post-quantum cryptography resistant to quantum attacks.

### The Order-finding Algorithm.

Given:

N is the number to be factored

A is a randomly chosen integer

Goal:

Find R, smallest positive integer such that A^R mod N = 1

Steps:

1. Prepare a quantum register that will cover all possible values from 0 to N-1

All qubits are put into superposition.

This holds all possible x values for the equation f(x) = a^x mod N

1. Prepare another quantum register initialized to 0

This will hold the result f(x)

1. Then store a^x mod N into the second register using the first register as x
2. Measure the second register.

This will get some value Y

This will also create a quantum entanglement between the two registers.

Because of this, the first register contains a periodic superposition of x values that yield f(x) = Y and also are multiples of R apart.

1. The Quantum Fourier Transform (QFT) is applied to the first register,

After this, measuring the first register yields R

### The Phase Estimation Algorithm.

It is used in Order finding algorithm

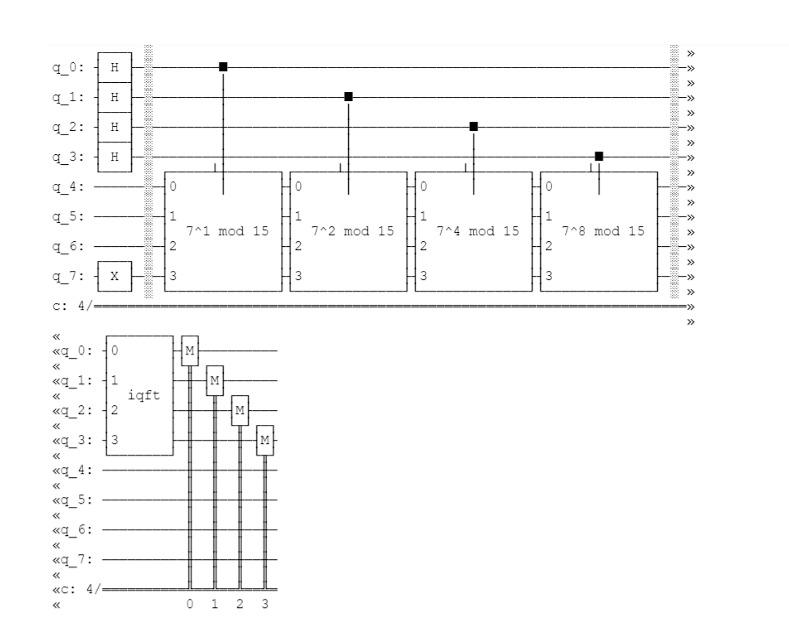
Order finding algorithm is essentially the phase estimation algorithm used on the equation:

A^R mod N = 1

The Phase estimation algorithm in general however can be used in various equations.

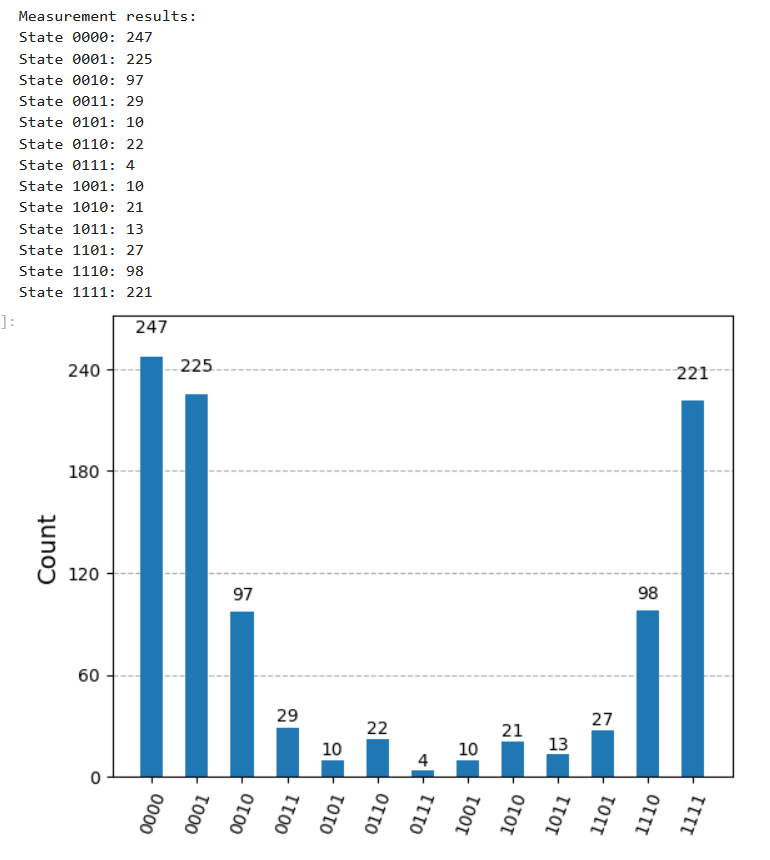
### Coding Results

Simplified Circuit (Q-munity)

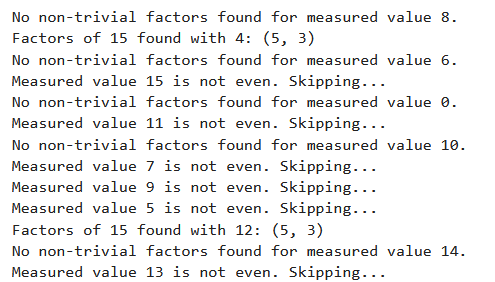


**Aer Simulator Results:**

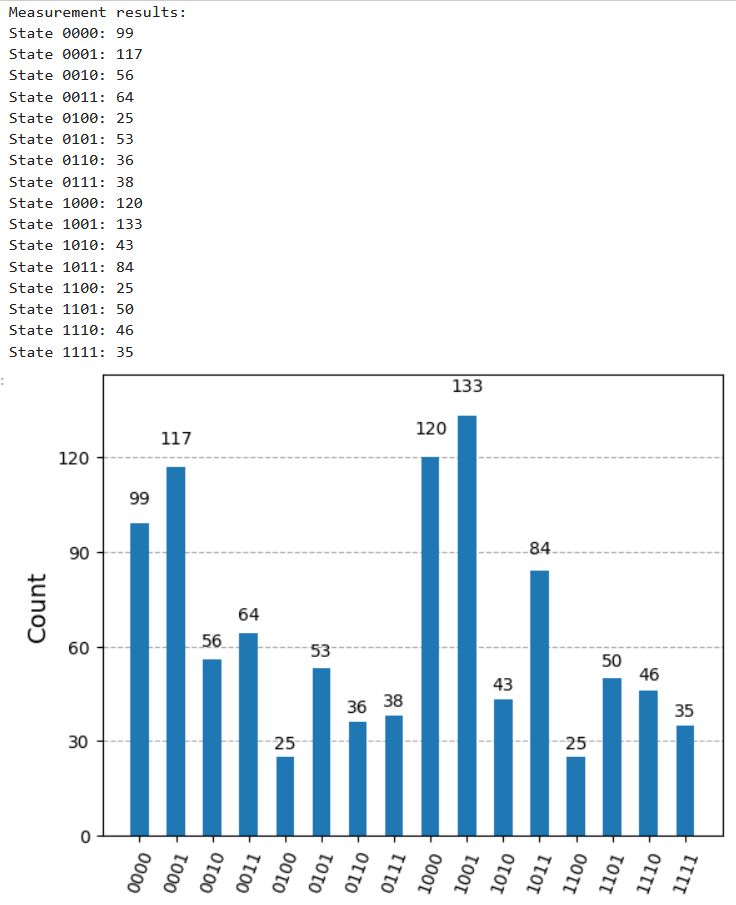
The output of the Inverse QFT giving us r:

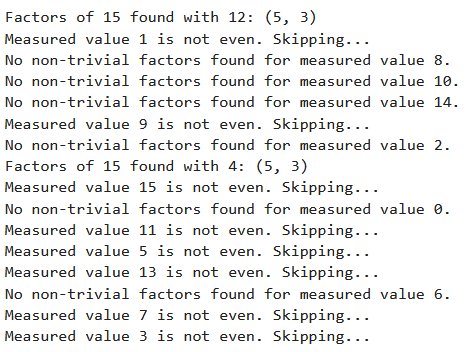


Testing gcd(x^(r/2) - 1, N) and gcd(x^(r/2) + 1, N) for the factors:



**IBM Computation**





Looking at the results for the Aer simulator run, we can find the factors by testing the output states as r in our post-processing algorithm. In our algorithm, we scan through and test all output states, but if we were to test only those representing an even number r (ends in 0 when reversed) and test in order of highest probability, we would first look at states |0000> and |1000>. State |0000> gives us GCD(r/2 - 1) = 0 & 5. In fact, 5 is a factor of 15 and dividing 15 by 5 gives us the other factor of 3. Running the same circuit through an IBM Q-computer, we get more scattered results, likely due to the more frequent errors with real qubits. However, the top two even results are still |1000> and |0000>, giving the same results in one less post-processing step due to |1000> having higher counts than |0000> in our IBM run.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### Contributions

Aimen Al-Doais

* Report
  + The prime factorization problem and its complexity class when executed on a classical computer, and how can it be used for encryption?
  + Code Results
* Code
  + Ran the IBM Simulation.
  + Interpretation of code output.
* Presentation
  + Created the entire presentation.
  + Presenting slides 1, 2-4, 14, and 15.

Ayman Elfayoumi

* Report
  + The Order-finding Algorithm.
  + The Phase Estimation Algorithm.
* Code
  + The initialization of the code
  + The entire circuit.
* Presentation
  + Presenting slides 1, 10-12, and 15.

Adham Elmadawy

* Report
  + What is Shor's Algorithm?
  + Mathematics Behind Shor's Algorithm.
  + Relevance of Shor's Algorithm?
  + Code Results.
* Code
  + Worked on getting the Aer Simulator to run properly using transpile.
  + The post processing step and finding the factors.
* Presentation
  + Presenting slides 1, 5-9, 13, and 15.

### References

0612 TV w/ NERDfirst. (2018, July 11). *Prime Factorization Algorithm!* YouTube. [Link](https://www.youtube.com/watch?v=kM_y15bYB8s)

Veritasium. (2023, March 20). *How Quantum Computers Break the Internet… Starting Now* YouTube. [Link](https://www.youtube.com/watch?v=-UrdExQW0cs&t=1266s&ab_channel=Veritasium)

Barker, E., & Dang, Q. (2015, January). *NIST Special Publication 800-57 Part 3 Revision 1*. NIST. [Link](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf)

Wikimedia Foundation. (2024, September 26). *General Number Field Sieve*. Wikipedia. [Link](https://en.wikipedia.org/wiki/General_number_field_sieve)

Wikimedia Foundation. (2024, November 29). *RSA (Cryptosystem)*. Wikipedia. [Link](https://en.wikipedia.org/wiki/RSA_(cryptosystem))

Qiskit Community Tutorials. *Shor's Algorithm*. GitHub. [Link](https://github.com/qiskit-community/qiskit-community-tutorials/blob/master/algorithms/shor_algorithm.ipynb)

Q-munity. *Shor’s Algorithm.* The Quantum Insider. [Link](https://qmunity.thequantuminsider.com/2024/06/10/shors-algorithm/)

GeeksforGeeks. *Shor's Factorization Algorithm*. [Link](https://www.geeksforgeeks.org/shors-factorization-algorithm/#)

OpenAI. (2024). *ChatGPT Language Model Output*. OpenAI.