A Review of Shor's Algorithm

Brady Li, Northeastern University, Boston, Massachusetts, USA

August 1st 2025

Abstract - Shor's algorithm stands out as a disruptive quantum algorithm demonstrating a significant exponential speedup over classical approaches to a well-known problem: integer factorization. This paper reviews the conceptual foundations, mathematical structures, and quantum methodologies that make Shor's algorithm possible, particularly its reliance on period finding and the Quantum Fourier Transform (QFT). We also briefly explore the broader implications of the algorithm, its current limitations, and its potential to undermine modern cryptographic systems.

***Keywords*: ** Shor's Algorithm; Quantum Computing; QFT; Integer Factorization; Period Finding

1 Introduction

Quantum computing radically changes how we approach algorithm design. Classical algorithms, such as Dijkstra's shortest path or mergesort, are deterministic and step-by-step. In contrast, quantum algorithms harness superposition, interference, and entanglement to operate on exponentially many states in parallel. These unique properties give rise to fundamentally different strategies for solving computational problems.

One of the most famous quantum algorithms is Shor's algorithm, developed by Peter Shor in 1994 with inspiration from Simon's quantum algorithm (1994), which involved finding a hidden binary string given a function with a specific periodicity. Shor's algorithm is also rooted in period finding and is designed to factor large integers in polynomial time, a task believed to require super-polynomial time classically. For those unfamiliar with cryptography, factoring large integers may appear simply as an interesting math problem, but in reality, it holds serious implications for our digital world.

This result showed, for the first time, that quantum computers could potentially break modern public-key cryptosystems like RSA, which rely on the difficulty of factoring large integers. This may sound alarming, prompting the question: why hasn't our digital world already collapsed? It is largely due to hardware limitations. Shor's algorithm does work, but only on relatively small numbers (like factoring 15 or 21). To factor something like a 2048-bit RSA key, you'd need a fault-tolerant quantum computer with millions of qubits; we're nowhere near that yet. Shor's algorithm is more like a sword in a stone right now, magical in theory, but waiting for the right machine to wield it. The moment we get a sufficiently powerful quantum computer, it will be a new age for cybersecurity.

2 Background

Shor's algorithm is based on a deep interplay between number theory, modular arithmetic, and quantum mechanics. This paper will review each step and its core components: period finding and the Quantum Fourier Transform (QFT) in the following sections, and assume a general understanding of quantum computing (quantum states, logic gates, and matrices).

2.1 Reducing Factoring to Period Finding

Shor's algorithm addresses the problem of factoring a large integer N. Classically, this problem is notoriously difficult: the best-known algorithms, such as the General Number Field Sieve, run in sub-exponential time. Shor's algorithm, by contrast, solves it in polynomial time by reframing factoring as a problem of finding the period of a modular exponentiation function, which quantum computers can solve exponentially faster.

Let us say we wish to factor a large composite number N. We define a function

$$f(x) = a^x \bmod N,$$

where a is a randomly chosen integer such that $1 < a < N$ and $\gcd(a, N) = 1$. This function is periodic, and the order r of $a \bmod N$ is defined as the smallest positive integer such that

$$a^r \equiv 1 \pmod{N}.$$

Why does finding r help us factor N? Consider the case where r is even and

$$a^{r/2} \not\equiv -1 \pmod{N}.$$

Then we can write:

$$a^r - 1 = \left(a^{r/2} - 1\right)\left(a^{r/2} + 1\right) \equiv 0 \pmod{N}.$$

Thus, N divides the product of two nontrivial numbers. Computing:

$$\gcd\left(a^{r/2} \pm 1, N\right)$$

reveals a nontrivial factor of N. These greatest common divisors are easy to compute classically via the Euclidean algorithm. Hence, the quantum part of Shor's algorithm is entirely focused on determining r, the order of $a \bmod N$, for a suitable a. Once r is known, the rest is efficient classical arithmetic.

This idea is rooted in the structure of modular arithmetic: for any two coprime integers A and B, the powers of A modulo B will eventually cycle. That is, repeated multiplication of A by itself will eventually produce a result congruent to 1 $\pmod{B}$. So,

$$A^x \equiv kB + 1,$$

for some x and k. In other words, the multiplicative group $\mathbb{Z}_N^\times$ is finite and hence all elements have finite order. This periodicity is the core of Shor's algorithm.

Reruns and Why They're Not a Problem

In the cases where the period r is odd, or if $a^{r/2} \equiv -1 \pmod{N}$, we do not obtain any factors. But that is okay! For most choices of a, these cases are rare. More formally, there is at least a 50% chance that r is even and yields a nontrivial square root of $1 \pmod{N}$. So if it fails, we pick another value of a and try again.

Furthermore, the quantum part of the algorithm is efficient; we can repeat it $O(\log N)$ times and still remain within polynomial time. The success probability compounds rapidly, making reruns and failed cases a trivial part of the process.

2.2 Quantum Fourier Transform

The Quantum Fourier Transform (QFT) is the quantum analog of the classical Discrete Fourier Transform (DFT), and it serves as the cornerstone for how a quantum computer extracts the hidden period r from the function $f(x) = a^x \bmod N$. Simply put, it reveals the "rhythm" behind quantum interference, enabling quantum computers to solve certain problems exponentially faster than classical ones. It works by transforming the periodic structure encoded in quantum states into a measurable frequency domain, allowing efficient extraction of hidden patterns like the period of a modular function.

The QFT uses a sequence of Hadamard gates and controlled phase rotations, applied in a specific order. These introduce interference based on the relative positions of the qubits, encoding phase relationships. The result is a superposition where the probability amplitudes spike at values that are multiples of the inverse period, making them likely to appear when measured. It essentially reshuffles phase information into a new configuration where the hidden structure (like the period of a function) shows up as peaks in the quantum interference pattern.

Mathematical Definition

For a quantum state over n qubits, we define $N = 2^n$. The QFT maps a computational basis state $|x\rangle$ to:

$$\text{QFT}\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N}\,|y\rangle$$

This transformation is represented by an $N \times N$ unitary matrix U_{FT} with complex entries:

$$U_{\text{FT}}[x, y] = \frac{1}{\sqrt{N}} e^{2\pi i x y/N}$$

This is the DFT matrix, and its action transforms the amplitude vector of a quantum state into its frequency components.

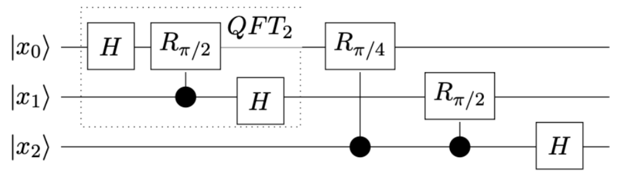
$$U_{FT} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix} \quad \text{where } \omega = e^{2\pi i/N}$$

While the definition of QFT depictes a $N \times N$ matrix, it is not implemented directly. Instead, we employ a circuit decomposition using Hadamard gates and controlled phase rotations R_k, where R_k is a single-qubit phase rotation, applied conditionally based on the control qubit's state.

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}$$

QFT Circuit

The QFT circuit for n qubits consists of the following steps and depicted here:



- **Apply Hadamard Gate:** This creates a superposition, initiating the phase encoding process.
- **Apply Controlled Phase Shifts:** For each subsequent qubit k from 2 to n, apply a controlled-R_k gate between the first and k-th qubit.
- **Repeat for Each Qubit:** Move to the next qubit, and similarly apply a Hadamard gate and controlled phase shifts. This creates an interference pattern across all qubits where the amplitudes encode the periodicity of the input function.
- **Reverse Qubit Order:** At the end of the circuit, apply a series of SWAP gates to reverse the order of the qubits, allowing for measurement.

Relation to Shor's Algorithm

After querying $f(x) = a^x \bmod N$ in superposition and measuring the output register, the input register collapses into a periodic superposition:

$$|\psi\rangle = \frac{1}{\sqrt{w}} \sum_{j=0}^{w-1} |x_0 + jr\rangle$$

where r is the period and $w \approx 2^n/r$ is the number of full periods in the input domain.

This state is periodic with period r. Applying the QFT to this state acts as a spectral analyzer. Since QFT is a linear operation, it leverages quantum interference: the amplitudes interfere constructively at frequencies $y \approx kN/r$ and destructively elsewhere.

The result of the QFT is a quantum state peaked at values y such that

$$y \approx \frac{kN}{r} \quad \text{and} \quad \frac{y}{2^n} \approx \frac{s}{r}, \quad \text{where } k, s \in \mathbb{Z}.$$

for some integers s, r:

$$\text{QFT}\,|\psi\rangle \approx \sum_{k=0}^{r-1} c_k \left|\left\lfloor \frac{kN}{r} \right\rceil\right\rangle$$

Measuring this state yields a value y, and with high probability:

$$\frac{y}{2^n} \approx \frac{s}{r}$$

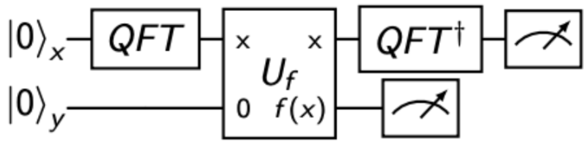
for some integer s. Using classical post-processing via the continued fraction expansion of $y/2^n$, we can deduce a candidate for r.

Efficiency

Unlike the classical Fast Fourier Transform, which runs in $O(N \log N)$ time where $N = 2^n$, the QFT completes the transform in just $O(n^2)$ time. This exponential speedup makes period finding, and therefor factoring, a tractable problem on a quantum computer. Without the QFT, we would be left with brute-force period detection, a process that scales poorly. Instead, the QFT utilizes quantum parallelism and interference to collapse the search space onto a small number of high-probability candidates for the period.

3 Implementation of Shor's Algorithm

Now that we've established the critical subroutines, we can describe the full implementation of Shor's algorithm. The following steps combine classical and quantum components to efficiently factor an integer N. The circuit can be generalized as such:



Step 1: Classical Preprocessing

Choose a random integer $a < N$. If $\gcd(a, N) \neq 1$, we have already found a nontrivial factor of N. Otherwise, define the function:

$$f(x) = a^x \bmod N.$$

Step 2: Quantum Period Finding

As outlined in Section 2.2, we now use a quantum computer to find the period r of $f(x)$. The process involves constructing the following quantum circuit:

1. **Initialize Registers:** Set both quantum registers to the state $|0\rangle\,|0\rangle$.
2. **Apply Hadamard Transform:** Apply Hadamard gates to the top register to create a uniform superposition.:

$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle\,|0\rangle\,,$$

   where $M = 2^m$ and m is the number of qubits used in the top input register.
3. **Apply Modular Exponentiation:** Apply a unitary transformation:

$$|x\rangle\,|0\rangle \longmapsto |x\rangle\,|f(x)\rangle = |x\rangle\,|a^x \bmod N\rangle\,.$$

   This step entangles the input and output registers.

4. **Measure the Output Register:** Measure the second (bottom) register. The outcome is a value $f(x_0)$, and the input register collapses into a superposition of all x satisfying $f(x) = f(x_0)$. The state becomes:

$$\frac{1}{\sqrt{w}} \sum_{j=0}^{w-1} |x_0 + jr\rangle ,$$

where $w \approx M/r$ and r is the period.

5. **Apply Inverse QFT:** Perform the inverse Quantum Fourier Transform on the top register. The periodic state transforms into a superposition that is sharply peaked at integer multiples of M/r, where $s \in \mathbb{Z}$ indexes the peaks:

$$\text{QFT}^{-1} \left(\frac{1}{\sqrt{w}} \sum_{j=0}^{w-1} |x_0 + jr\rangle \right) \approx \sum_{s=0}^{r-1} c_s |y_s\rangle , \quad \text{where } y_s \approx \frac{sM}{r}.$$

6. **Measure the Top Register:** Measuring this register yields an integer y such that

$$\frac{y}{M} \approx \frac{s}{r},$$

where s is an unknown integer. This gives us a good approximation of the rational number s/r.

Step 3: Classical Post-Processing

Use the continued fraction expansion of y/M to recover a candidate value for r. If r is even and $a^{r/2} \not\equiv -1 \pmod{N}$, then using Euclid's, which is trivial, compute:

$$\gcd \left(a^{r/2} \pm 1, N \right) ,$$

which will yield a nontrivial factor of N. If this fails, repeat the process with a new choice of a.

Example: Factoring 15

Let $N = 15$ and choose $a = 2$. Then $f(x) = 2^x \bmod 15$. The powers of 2 modulo 15 are:

$$2^1 = 2,\ 2^2 \quad = 4,\ 2^3 = 8,\ 2^4 \quad = 16 \equiv 1 \pmod{15}.$$

Thus, the period is $r = 4$. Then:

$$2^{r/2} = 2^2 = 4,$$

and the greatest common divisors are:

$$\gcd(4-1, 15) = \gcd(3, 15) = 3, \quad \gcd(4+1, 15) = \gcd(5, 15) = 5.$$

So, we have successfully factored 15 into 3 and 5.

4 Going Forward

While Shor's algorithm has shown to be theoretically sound, its full potential remains out of reach due to current hardware limitations. The algorithm's power lies in its exponential speedup for integer factorization, a problem at the heart of cryptography. However, unlocking that power requires a quantum computer of immense scale and reliability.

In order to factor a 2048-bit RSA key, we would need a quantum computer with millions of physical qubits with proper error correction. Today's quantum devices typically operate with fewer than 100 high-quality qubits, short coherence times, and non-negligible gate error rates. The procedures in the algorithm such as modular exponentiation and QFT require deep quantum circuits, making them highly fragile in noisy environments.

Despite this, real-world demonstrations continue to validate Shor's design. IBM's 2001 NMR experiment successfully factored 15 and newer platforms, such as superconducting qubits and trapped ions, have improved implementation fidelity. In the field, researchers have also worked towards optimized variants, including approximate QFT, semi-classical Fourier transforms, and modular arithmetic techniques to reduce resource overhead.

Most notably, Shor's algorithm has started the urgent global effort to develop post-quantum cryptography. Even if we are years away from cracking large keys, the mere existence of Shor's algorithm has exposed the vulnerability of widely used encryption schemes like RSA and ECC. Governments and Institutions like banks and technology companies are already preparing for a future where encrypted data, harvested today, may be decrypted when sufficiently advanced quantum hardware becomes available. It is both a display of quantum computational power and a foreseeable end to classical cryptography. As quantum hardware advances, Shor's algorithm will transition from theory to practical threat. When that day comes, it will not only be a triumph of computer science and engineering, but a fundamental turning point in how we secure digital information.

5 Conclusion

Shor's algorithm represents an elegant blend of structure and abstract mathematics. Rather than brute-forcing a solution to factoring, it reframes the problem entirely to period finding and utilizes the quantum Fourier transform to find it. The algorithm allows quantum mechanics to do what it does best: amplify the signal, suppress the noise, and let the answer emerge through interference. It uses the periodicity hidden in modular exponentiation as a bridge between classical number theory and quantum mechanics. The genius of the algorithm lies in how naturally each component supports the next: modular arithmetic encodes a structure, quantum superposition preserves it across many inputs, and the quantum Fourier transform exposes it through interference.

Shor's algorithm is less a collection of clever tricks and more a unified strategy and blend of disciplines that captures the very essence of quantum computation. It provides a perspective for how quantum algorithms might continue to emerge when we learn to see problems through the lens of quantum computing.

References and Acknowledgments

Special thanks to Andrew Steane and the Exeter College Summer Program (ECSP).

This paper is a final paper for the Quantum Computer Science course at ECSP. Due to constraints imposed by page limits and the scope of the course, this paper does not explore the latest advancements in quantum algorithm research. The sources cited herein reflect foundational and accessible treatments of the topic, selected to align with the educational objectives of this work.

1. Qiskit by IBM Quantum, *Shor's Algorithm – How Quantum Computers Break Encryption*, YouTube, 2019. Available at: `https://www.youtube.com/watch?v=lvTqbM5Dq4Q&t=870s`. Accessed: August 2, 2025.
2. Wikipedia contributors, *Shor's Algorithm*, Wikipedia. Available at: `https://en.wikipedia.org/wiki/Shor%27s_algorithm`. Accessed: August 2, 2025.

3. University of California, Berkeley, *BerkeleyX CS191x: Quantum Mechanics and Quantum Computation, Chapter 5 – QFT, Period Finding & Shor's Algorithm*, edX. PDF available at: `https://courses.edx.org/c4x/BerkeleyX/CS191x/asset/chap5.pdf`.

4. Andrew Steane, *ECSP Quantum Computer Science Lectures*, University of Oxford. Distributed as institutional lecture notes.