CS 599 P1: Introduction to Quantum Computation — Boston University, Fall 2025
Instructor: Alexander Poremba — **Scribe:** Steve Choi

LECTURE #12: SHOR'S ALGORITHM

1 Introduction

In the previous lectures, we explored the Quantum Fourier Transform (QFT) and Quantum Phase Estimation (QPE), two important algorithmic tools in quantum computing. Using these two building blocks, we will now study Shor's algorithm (proposed by Peter Shor [Sho97]), one of the most groundbreaking quantum algorithms that demonstrated an exponential speedup over the best known classical methods for factoring, which threatens most of the public-key cryptography today. An important example is the popular RSA encryption scheme, which can be broken in polynomial-time using Shor's algorithm.

2 The RSA Encryption Scheme

The RSA protocol, introduced by Rivest, Shamir, and Adleman, is a secure public-key encryption scheme whose security rests on the hardness of *factoring*: given a large number $N \in \mathbb{N}$, where

$$N = p \cdot q$$

and p, q are prime numbers, it is computationally infeasible to find the two prime factors of N.

For readers not familiar with cryptography, we first define what a public-key encryption scheme is.

Public-Key Encryption. A public-key encryption scheme is a method that allows two parties, traditionally called *Alice* and *Bob*, to communicate securely over an open (public) channel. Alice generates a pair of keys:

- a *public key* pk (to be published on the internet), as well as a
- a *secret key* sk (to be kept private).

Anyone (such as Bob) can use pk to encrypt a message and send it to Alice, but only Alice—using her secret key sk—can decrypt it and recover the original message. Formally, a public-key encryption scheme is a triplet of efficient (probabilistic) algorithms:

- (Key generation:) on input a security parameter, it outputs a public key pk and a secret key sk.
- (Encryption:) Given the public key pk and a plaintext message m, it outputs a ciphertext c.
- (Decryption:) Given the secret key sk and ciphertext c, it outputs the original plaintext message m.

We describe the RSA Algorithm as follows.

The RSA Algorithm

1. **Key Generation:**
   - Choose two large prime numbers p and q, and compute their product $N = p \cdot q$.
   - Compute Euler's totient $\varphi(N) = (p-1)(q-1)$.
   - Choose an integer e, the public exponent, such that $1 < e < \varphi(N)$ and $\gcd(e, \varphi(N)) = 1$.
   - Compute d, the private exponent, such that $e \cdot d \equiv 1 \pmod{\varphi(N)}$.
   - Let $pk = (N, e)$ and $sk = d$.
2. **Encryption:**
   - To encrypt a message m into ciphertext c, compute
   $$c = m^e \pmod{N}.$$
3. **Decryption:**
   - To decrypt a ciphertext c into message m, compute
   $$m = c^d \pmod{N}.$$

The RSA encryption algorithm is considered secure against classical computers, under the assumption that the best known classical algorithm for factoring takes roughly $2^{(\log N)^{1/3}}$ time—far longer than the age of the universe if N is a 2048-bit number. However, Peter Shor discovered that in the quantum setting, the prime factors p and q could be found in $\sim (\log N)^2$ time, a very reasonable amount!

3 Shor's Algorithm

Before we dive into the technical details of Shor's Algorithm, we present a high-level overview of the entire factoring procedure. Shor's algorithm combines classical number theory with quantum computation to efficiently factor a composite integer N. It consists of two main components: a *classical part* and a *quantum part*.

The key idea is to reduce the problem of integer factorization, which is believed to be hard for classical computers, into a problem of *period finding*. While the classical part performs this reduction and post-processes the result, the quantum part efficiently finds the period using the principles of quantum superposition and interference.

3.1 High-Level Description

Given an integer N that we wish to factor, Shor's algorithm proceeds as follows.

Shor's Algorithm (High-Level Steps)

1. **Random Selection (Classical Part).** Randomly choose an integer x such that $1 < x < N$.
   - If $\gcd(x, N) > 1$, then we have already found a nontrivial factor of N, and the algorithm terminates.
   - Otherwise, continue to the next step.
2. **Reduction to Order-Finding (Classical Part).** Define the *order* r of x modulo N as the smallest positive integer satisfying
   $$x^r \equiv 1 \pmod{N}.$$
   The factoring problem is now reduced to finding this period r. Classically, computing r is believed to be hard; however, it is precisely this subproblem that a quantum computer can solve efficiently.
3. **Period-Finding Subroutine (Quantum Part).** Invoke a quantum procedure that, given (x, N), efficiently finds the period r of the function
   $$f(j) = x^j \bmod N.$$
   This step leverages quantum parallelism and the Quantum Fourier Transform (QFT) to extract the periodic structure of f. We treat this as a *quantum black box* that outputs the correct period r with high probability.
4. **Post-Processing (Classical Part).** Once r is obtained, check whether it leads to nontrivial factors of N:
   - If r is odd, discard x and repeat from Step 1.
   - Compute $x^{r/2} \pmod{N}$. If $x^{r/2} \equiv -1 \pmod{N}$, the attempt fails; choose a new random x and repeat.
   - Otherwise, compute
     $$p = \gcd(x^{r/2} - 1, N), \qquad q = \gcd(x^{r/2} + 1, N).$$
     If p or q yields a nontrivial divisor of N, we have successfully factored N.
5. **Repeat if Necessary (Classical Part).** If no nontrivial factors are found (which occurs with small probability), repeat the entire procedure with a new random x.

3.2 Classical Part

We first state some number-theoretic components that we will use throughout the classical part.

Definition 3.1. *The set $\mathbb{Z}_N^*$, defined by*

$$\mathbb{Z}_N^* = \{x \in \{1, 2, \ldots, N-1\} \mid \gcd(x, N) = 1\},$$

is the set of integers modulo N that are coprime to N. A useful property is that elements in $\mathbb{Z}_N^*$ have a multiplicative inverse modulo N (e.g., if $x \in \mathbb{Z}_N^*$, then there exists $y \in \mathbb{Z}_N^*$ such that $xy \equiv 1 \pmod{N}$).

Definition 3.2. *The order of x in $\mathbb{Z}_N^*$, denoted by $\mathrm{ord}(x)$, is the smallest integer $r \in \mathbb{N}$ such that*

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

In the classical part, given an integer $N \in \mathbb{N}$, we will first randomly choose a number $x \in \{2, \ldots, N-1\}$ such that $\gcd(x, N) = 1$.

If we randomize over the choice of $x \in \{2, \ldots, N-1\}$, we will have the following property, with probability at least $\frac{1}{2}$.

Number-theoretic property (*)

- The order $\mathrm{ord}(x)$ of $x \in \{2, \ldots, N-1\}$ is even;
- $(x^{r/2} + 1)$ and $(x^{r/2} - 1)$ are not multiples of N, i.e., for $k \geq 0$,

$$(x^{r/2} + 1) \neq kN \quad \text{and} \quad (x^{r/2} - 1) \neq kN.$$

We first deal with the (easy) edge case, which happens with a tiny probability. If $\gcd(x, N) > 1$, then we have found a nontrivial factor of N, and thus we are done. Otherwise, if $\gcd(x, N) = 1$, then x and N do not share any factor apart from 1.

Consider the sequence

$$x^0 \pmod{N},\ x^1 \pmod{N},\ x^2 \pmod{N},\ \ldots,\ x^r \pmod{N}.$$

By the pigeonhole principle, this cannot continue indefinitely and we will eventually get $x^r \equiv 1 \pmod{N}$, where $r = \mathrm{ord}(x)$ in $\mathbb{Z}_N^*$. Observe that this sequence has a periodic behavior, where $x^0 \equiv x^r \pmod{N}$, $x^1 \equiv x^{r+1} \pmod{N}$, and so on. This periodic behavior is what we will utilize later in the quantum part to efficiently find the factors.

Assume the first property of Property (*) holds. Then we have that $x^{r/2} \pmod{N}$ exists, which implies

$$\begin{aligned}
& x^r \equiv 1 \pmod{N} && \\
\iff & (x^{r/2})^2 - 1 \equiv 0 \pmod{N} && (r \text{ is even}) \\
\iff & (x^{r/2} + 1)(x^{r/2} - 1) \equiv 0 \pmod{N}. && (\text{factor terms})
\end{aligned}$$

Equivalently, we have

$$N \mid (x^{r/2} + 1)(x^{r/2} - 1).$$

By property $(*)$ and since $x > 1$, we have $1 < x^{r/2} \pm 1 \pmod{N} < N$. Thus, by the assumption that N is a composite number, we know that there exist p and q such that $N = p \cdot q$. Therefore, any factor of N satisfies either

- $x^{r/2} + 1 \mid N$, or
- $x^{r/2} - 1 \mid N$.

This implies $\gcd(x^{r/2} + 1, N) > 1$ or $\gcd(x^{r/2} - 1, N) > 1$, which yields a true non-trivial factor.

Without loss of generality, let $p = \gcd(x^{r/2} + 1, N) > 1$. Then, we have $q = N/p$, and we are done, since we have factored N into its prime factors p and q.

Remark 1. *If property $(*)$ does not hold, we try again until property $(*)$ holds. This can be done in a constant expected number of repetitions.*

The above steps show that if we can somehow find the order r, we would be done. However, finding the order in the classical setting is computationally infeasible. The next (quantum) part will demonstrate how we can use quantum computing to our advantage to find r in a reasonable amount of time.

3.3 Quantum Part

We now demonstrate how we can formulate the problem of finding the order into a period-finding problem, which can be efficiently solved with a quantum computer. The key insight is to express the periodic structure of $x^j \pmod N$ in terms of the eigenvalues of a unitary operator, which allows us to use Quantum Phase Estimation (QPE) to find the period.

Recall from the previous lecture that QPE requires a unitary operator and one of its eigenstates. Our goal now is to show how we can construct such a unitary matrix.

We define the following unitary operator for n qubits such that $2^n \geq N$, over $\mathbb{C}^{2^n}$:

- $\mathcal{U}_x |y\rangle = |x \cdot y \bmod N\rangle$, for $0 \leq y < N$.
- $\mathcal{U}_x |y\rangle = |y\rangle$, for $N \leq y < 2^n$.

If the basis state y is at least N, we treat it as if we are applying the identity operator on $|y\rangle$, for $N \leq y < 2^n$, to ensure $\mathcal{U}_x$ is unitary.

We also need to find a way to efficiently implement a controlled-$\mathcal{U}_x^j$ operator to extract information about the phase we are trying to estimate.

For $j \in \{0, \ldots, 2^t - 1\}$, we can define

$$\mathcal{U}_x^j |y\rangle = |x^j \cdot y \bmod N\rangle ,$$

which can be computed in $\mathrm{poly}(n, t)$ time, since exponentiation by j can be done in $O(\log j)$ time via repeated squaring.

Algorithm: Repeated Squaring

Given integers a and b, we can recursively compute a^b in $O(\log b)$ time as follows:

1. **Input:** Integers a and b.
2. If $b = 1$, return a.
3. If b is even, recursively compute $a^{b/2}$, then return $(a^{b/2})^2$.
4. If b is odd, recursively compute $a^{\lfloor b/2 \rfloor}$, then return $a \cdot (a^{\lfloor b/2 \rfloor})^2$.

Let $r = \mathrm{ord}(x)$. For $0 \leq s < r$, define

$$|v_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega_r^{-sk} |x^k \bmod N\rangle ,$$

where $\omega_r = e^{2\pi i/r}$. This family of $|v_s\rangle$ states remains invariant under applying $\mathcal{U}_x$ up to a phase:

$$\begin{aligned}\mathcal{U}_x |v_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega_r^{-sk} |x^{k+1} \bmod N\rangle \\ &= \omega_r^s |v_s\rangle .\end{aligned}$$

Hence, $\omega_r^s = e^{2\pi i s/r}$ is the eigenvalue corresponding to $|v_s\rangle$.

However, we do not know r, so we cannot directly construct $|v_s\rangle$. Fortunately, $|1\rangle$ is a uniform superposition of all $|v_s\rangle$:

$$|1\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |v_s\rangle .$$

Therefore, for $j \in \{0, \ldots, 2^t - 1\}$, we have

$$\mathcal{U}_x^j |1\rangle = |x^j \bmod N\rangle .$$

Now, we can run the quantum part of Shor's algorithm—the (approximate) Quantum Phase Estimation algorithm—to obtain r:

Quantum Phase Estimation (with t-bits of precision)

1. Initialize $|0^t\rangle |1\rangle$.
2. Prepare a uniform superposition on register #1:
$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |1\rangle .$$
3. Apply controlled-$\mathcal{U}^j$ to get:
$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |x^j \bmod N\rangle \approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle |v_s\rangle .$$
4. Apply $\mathrm{QFT}_{2^t}^\dagger$ on register #1:
$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\tilde{s}/r\rangle |v_s\rangle .$$
5. Measure register #1 to obtain $\tilde{s}/r$, an estimate of s/r.
6. Run the continued fractions algorithm to recover r.

Once measured, we obtain an estimate

$$\tilde{\phi} = \frac{\tilde{s}}{2^t} \approx \frac{s}{r},$$

where $\tilde{s}$ is the measured integer outcome on the first register, represented as

$$\tilde{s}/r = 0.\underbrace{11001\ldots1}_{t\text{-bits}},$$

accurate up to t bits in its binary expansion. Intuitively, $\tilde{\phi}$ approximates the phase corresponding to an eigenvalue $e^{2\pi i s/r}$ of the unitary operator $\mathcal{U}_x$, and thus encodes information about the (unknown) period r.

To recover r from $\tilde{\phi}$, we use the *continued fractions algorithm*. The idea is to express $\tilde{\phi}$ as a rational number with a small denominator, under the assumption that $\tilde{\phi}$ is a close approximation to a rational of the form s/r.

Continued Fractions Algorithm

Given an estimate ϕ of s/r, satisfying

$$|\phi - s/r| \leq \frac{1}{2r^2},$$

the continued fractions algorithm returns the reduced fraction s/r in lowest terms.
The algorithm proceeds as follows:

1. Expand ϕ as a continued fraction:

$$\phi = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots + \cfrac{1}{a_k}}},$$

   where a_i are integers.

2. Compute the convergents $\frac{p_j}{q_j}$ for each truncation of this expansion.

3. For each candidate denominator q_j, test whether $x^{q_j} \equiv 1 \pmod{N}$.

4. The smallest q_j satisfying this condition is the order r.

The algorithm runs in time $\mathrm{polylog}(r)$.

Intuition and Accuracy. Since our quantum measurement has finite precision, $\tilde{\phi}$ is not exactly equal to s/r. However, if we choose t large enough (typically $t = 2n$ bits, where $n = \lceil \log_2 N \rceil$), the approximation satisfies

$$\left|\tilde{\phi} - \frac{s}{r}\right| \leq \frac{1}{2^{t+1}} \leq \frac{1}{2r^2},$$

ensuring that the continued fractions algorithm can uniquely identify the correct rational number s/r.

Note that the value of s is uniformly random in $\{0, 1, \ldots, r-1\}$, and since $\gcd(s, r)$ may not be 1, the first fraction we recover might correspond to a smaller denominator $r' = r/\gcd(s, r)$. In that case, we can test whether $x^{r'} \equiv 1 \pmod{N}$:

- If true, then r' is already the correct period.

- If false, we can repeat the algorithm with a new random x until we obtain an s coprime with r (which happens with probability $\varphi(r)/r > 1/\log\log r$).

From the Order to the Factors. Once we have found the correct order r, we return to the classical part. If r is even and $x^{r/2} \not\equiv -1 \pmod{N}$, then we compute:

$$p = \gcd(x^{r/2} - 1, N), \qquad q = \gcd(x^{r/2} + 1, N),$$

yielding two nontrivial factors p and q of N. Each step here is efficiently computable on a classical computer.

Complexity. Overall, the quantum circuit for period finding requires $O((\log N)^2)$ operations, dominated by modular exponentiation and the Quantum Fourier Transform. Together with the efficient post-processing using continued fractions and $\gcd$ computations, Shor's algorithm factors N in polynomial time, certainly at most $O((\log N)^3)$, which is exponentially faster than the best known classical algorithms.

Summary. To summarize:

1. The quantum part efficiently estimates a phase $\tilde{\phi} \approx s/r$.
2. Classical post-processing with continued fractions recovers the true period r.
3. Using r, we compute nontrivial factors of N.

Hence, Shor's algorithm demonstrates that quantum computers can efficiently solve the integer factorization problem — a problem believed to be intractable for classical computation.

References

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. 1