RSA and Shor's Algorithm

CS-4390/5390: Quantum Information Science (QIS)

Mohammad Saidur Rahman, Ph.D.

Department of Computer Science
The University of Texas at El Paso (UTEP)

Course Website: https://quantum.rahmanmsaidur.com/

April 10, 2025

Announcements

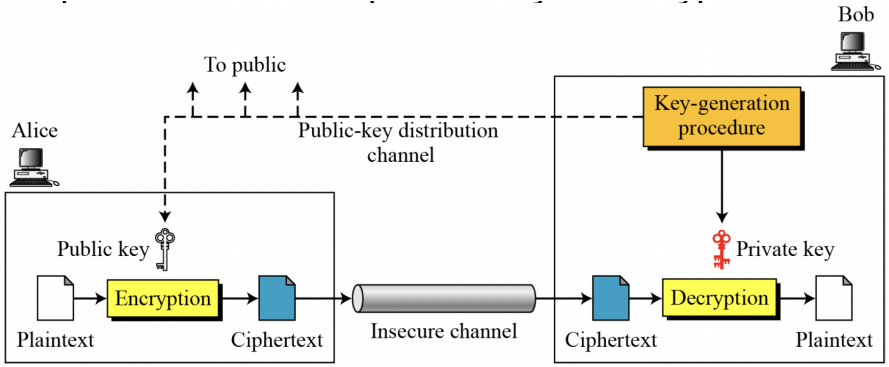
- Project Proposal Presentation (Group):
  - April 15, 2025
  - 15 mins + 5 mins
- Quiz 2 $\rightarrow$ April 17, 2025
- Comprehensive Quiz $\rightarrow$ May 01, 2025
- Lab 5: QML
- Final Project Presentation (Group):
  - 30 mins + 5 mins
  - Two Groups, May 06, 2025
  - Two Groups, May 08, 2025
  - Report + Slides need to be submitted as well.

Public Key Cryptography

- **Public Key Cryptography** = Asymmetric Cryptography
- Uses **two keys** that come in pairs:
  - **Public Key** (shared openly)
  - **Private Key** (kept secret)
- Keys are mathematically linked but **different**.
- **Usage**:
  - One key encrypts the message.
  - The other key decrypts the message.

How Public Key Encryption Works

- **Sender** encrypts the message using the **recipient's public key**.
- **Recipient** decrypts the message using their **private key**.



Public Key Cryptography

Strengths:

- **Better Scalability**: Number of keys grows linearly with users.
- **Supports Non-Repudiation**: Digital signatures prove authorship.
- **Simplified Key Distribution**: Public keys are openly shareable.

Weakness:

- **Slow**: Computation is heavier than symmetric cryptography.

Security of Public Key Cryptography

- Based on **trapdoor one-way function**.
- **One-way function**:
  - Easy to compute: $Y = f(X)$
  - Infeasible to reverse: find X from Y such that $X = f^{-1}(Y)$
- **Trapdoor one-way function**:
  - $Y = f_k(X)$: easy if k and X are known
  - $X = f_k^{-1}(Y)$: easy if k and Y are known
  - $X = f_k^{-1}(Y)$: infeasible if only Y is known but k is unknown
- Example: **Integer Factorization Problem**:
  - Given primes p and q, compute $N = p \times q$ easily.
  - Given N, finding p and q is hard.

Designing a public-key algorithm is to find an appropriate trapdoor one-way function.

Requirements for Public Key Cryptography

- It is computationally easy to generate a pair of public key and private key.
- It is computationally easy to generate a ciphertext using the public key.
- It is computationally easy to decrypt the ciphertext using the private key.
- It is computationally infeasible to determine the private key from the public key.
- It is computationally infeasible to recover the message from the ciphertext and the public key.

Uses of Public Key Cryptography

1. **Encryption/Decryption**:
   - Suppose we encrypt message, M, with Bob's public key, PK
   - Only Bob's private key, SK ,can decrypt to find M
2. **Digital Signature**:
   - Bob signs by "encrypting" with his private key.
   - Anyone can verify the signature by "decrypting" with Bob's public key.
   - Only the private key holder (Bob) could have created the valid signature.
   - It is like a handwritten signature.
3. **Key Exchange**:
   - Two parties cooperate to securely exchange a session key.
   - Symmetric key cryptosystems are used later with the session key.

RSA

- Invented by Rivest, Shamir, and Adleman (RSA)[1] in 1977
- An equivalent system was developed secretly in 1973 at Government Communications Headquarters (GCHQ), the British signals intelligence agency, by the English mathematician Clifford Cocks
- Depends on difficulty of integer factorization problem – product of **VERY LARGE** prime numbers
- **Generation of RSA public-private key pair**:
  - Let p and q be two large prime numbers.
  - Let $N = p \times q$ be the modulus.
  - Choose e relatively prime to $\varphi(N)$, where $\varphi(N) = (p-1)(q-1)$.
  - Find d such that $e \cdot d \equiv 1 \mod \varphi(N)$.
  - **Public key** is (N, e)
  - **Private key** is d *(Note: p and q are also secrets!)*

[1] Rivest, Ronald L., Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21, no. 2 (1978): 120-126.

RSA Key Generation

- Select two large random prime numbers:

$$p \quad \text{and} \quad q$$

- Compute the modulus:

$$N = p \times q$$

- N is used in both public and private keys.

RSA Key Generation

- Compute Euler's totient function:

$$\varphi(N) = (p-1)(q-1)$$

- $\varphi(N)$ is the number of integers less than N that are coprime to N.

RSA Key Generation

- Choose a public exponent e such that:

$$1 < e < \varphi(N) \quad \text{and} \quad \gcd(e, \varphi(N)) = 1$$

- e and $\varphi(N)$ must be coprime.

RSA Key Generation

- Compute the private exponent d such that:

$$e \times d \equiv 1 \mod \varphi(N)$$

- d is the modular inverse of e modulo $\varphi(N)$.
- **Public Key**: (N, e)
- **Private Key**: (N, d)
- *Important:* p and q must be kept secret!

RSA: Encryption and Decryption

- **To encrypt** a message M:

$$C = M^e \mod N$$

  where C is the ciphertext.
- **To decrypt** the ciphertext C:

$$M = C^d \mod N$$

  where M is the recovered plaintext.
- N: Modulus, e: Encryption exponent, d: Decryption exponent.

RSA: Security Considerations

- Recall that e and N are **public**.
- If an attacker can factor N, they can compute $\varphi(N)$ and find d since:

$$e \times d \equiv 1 \mod \varphi(N)$$

- **Factoring the modulus** N breaks RSA.
- **RSA Problem**: Is factoring the *only* way to break RSA?

Public-Key Cryptanalysis

- **Brute-force attack**
  - Try all possible keys.
- **Derivation of private key from public key**
  - Try to find the relationship between the public key and the private key.
  - Attempt to compute the private key from the public key.
- **Probable-message attack**
  - The public key is known.
  - Encrypt all possible messages.
  - Try to find a match between the ciphertext and one of the encrypted messages.

Does RSA Really Work?

- Given the ciphertext:

$$C = M^e \mod N$$

we want to prove that decryption recovers the message:

$$M = C^d \mod N = M^{ed} \mod N$$

- **We will use Euler's Theorem**:
  - If M is relatively prime[2] to N, then:

$$M^{\varphi(N)} \equiv 1 \mod N$$

- This theorem is the key reason why RSA works!

[2] The two numbers M and N do not share any common factors except 1. $gcd(M, N) = 1$

RSA Proof

- We know that:
$$e \times d \equiv 1 \mod (p-1)(q-1)$$
- By definition of congruence, this means:
$$e \times d = k(p-1)(q-1) + 1$$
where k is an integer.
- Recall that:
$$\varphi(N) = (p-1)(q-1)$$
- Thus:
$$e \times d = k\varphi(N) + 1$$
- This relationship is crucial for the next step.

RSA Proof

- Start from:
$$M^{ed} = M^{k\varphi(N)+1}$$
- Using properties of exponents:
$$M^{k\varphi(N)+1} = M^{k\varphi(N)} \times M$$
- By Euler's theorem:
$$M^{\varphi(N)} \equiv 1 \mod N \quad \Rightarrow \quad (M^{\varphi(N)})^k \equiv 1^k \equiv 1 \mod N$$
- Thus:
$$M^{ed} \equiv M \times 1 = M \mod N$$

RSA Proof

- Therefore, RSA decryption correctly recovers the original message:

$$C^d \mod N = M$$

- **Note:** RSA also works if M is not relatively prime to N, but the proof becomes more complicated.
- RSA relies on the special relationship:

$$e \times d \equiv 1 \mod \varphi(N)$$

and the structure of modular arithmetic!

RSA Example: Key Generation

- Choose two prime numbers:
$$p = 5, \quad q = 11$$
- Compute modulus:
$$N = p \times q = 5 \times 11 = 55$$
- Compute Euler's totient:
$$\varphi(N) = (p-1)(q-1) = 4 \times 10 = 40$$

RSA Example: Public and Private Keys

- Choose public exponent:

$$e = 3 \quad (\text{since } \gcd(3, 40) = 1)$$

- Compute private exponent d:

$$e \times d \equiv 1 \mod 40$$

- Solve:

$$3 \times 27 = 81 \quad \Rightarrow \quad 81 \mod 40 = 1$$

- Thus:

$$d = 27$$

- **Public Key**: $(N = 55, e = 3)$ **Private Key**: $d = 27$

RSA Example: Encryption

- Suppose message:
$$M = 7$$
- Encrypt:
$$C = M^e \mod N = 7^3 \mod 55$$
- Calculate:
$$7^3 = 343, \quad 343 \mod 55 = 13$$
- Thus, ciphertext:
$$C = 13$$

RSA Example: Decryption

- Decrypt ciphertext:

$$M = C^d \mod N = 13^{27} \mod 55$$

- **Step 1: Precompute small powers of 13 modulo 55**
  - $13^2 \mod 55 = (13 \times 13) \mod 55 = 169 \mod 55 = 4$
  - $13^4 \mod 55 = (4 \times 4) \mod 55 = 16$
  - $13^8 \mod 55 = (16 \times 16) \mod 55 = 256 \mod 55 = 36$
  - $13^{16} \mod 55 = (36 \times 36) \mod 55 = 1296 \mod 55 = 31$
- **Goal**: Use these to build 13^{27} easily without computing 13^{27} directly!

RSA Example: Decryption

- **Step 2: Break 27 into binary sum of powers of 2**

$$27 = 16 + 8 + 2 + 1$$

- Thus:

$$13^{27} = 13^{16} \times 13^{8} \times 13^{2} \times 13^{1} \mod 55$$

- **Step 3: Multiply step-by-step under modulo 55**
  - $31 \times 36 \mod 55 = 1116 \mod 55 = 16$
  - $16 \times 4 \mod 55 = 64 \mod 55 = 9$
  - $9 \times 13 \mod 55 = 117 \mod 55 = 7$
- Thus, recovered message:

$$M = 7$$

- We get back the original message 7 after decryption!

Factoring N: Naïve and Advanced Algorithms

- The **naïve algorithm** (trial division) tests all divisors up to $\sqrt{N}$.
  - Every divisor larger than $\sqrt{N}$ must have a corresponding smaller divisor.
- **Worst-case time complexity**: exponential in $n = \log(N)$ (the number of digits needed to specify N).
- **Faster algorithms discovered by number theorists**:
  - **Quadratic Field Sieve** (1981): runs in approximately $2^{O(\sqrt{n})}$.
  - **Number Field Sieve**: fastest known, runs in roughly $2^{O(n^{1/3})}$.
  - The correctness of the Number Field Sieve depends on an unproven conjecture.

Breaking RSA?

Breaking a 2048-bit RSA key would take 1 billion years with a classical computer.

Breaking RSA?

Breaking a 2048-bit RSA key would take 1 billion years with a classical computer.

How long would it take for a quantum computer?

Breaking RSA?

A quantum computer could do it in 100 seconds.[3]

[3] https://www.technologyreview.com/2019/05/30/65724/how-a-quantum-computer-could-break-2048-bit-rsa-encryption-in-8-hours/

Shor's Algorithm

Problem?

Say we are given a number N that is the product of two prime numbers p and q. The goal is to factor N, i.e., to find its factorsp and q.

Factoring with Classical Computers

The best-known classical algorithm for factoring large numbers is the **number field sieve**.
The exact working of the number field sieve is beyond the scope of this lecture, but its runtime for factoring an n-bit number grows approximately like $e^{n^{1/3}}$, which is considered **subexponential**.
This means:

- The runtime grows faster than any polynomial function of n, making factoring **inefficient** for classical computers.
- However, it grows slower than a true exponential function, due to the influence of natural logarithms.

As a result, factoring remains hard but not as hard as purely exponential problems.

Overview of Shor's Algorithm[4]

- Goal: Efficiently factor a large number $N = pq$ where p and q are prime.
- Shor's Algorithm consists of three key steps:
  1. Choose a random number a and compute $\gcd(a, N)$.
  2. Find the period r of $a^x \mod N$.
  3. Use r to factor N.
- Combines classical and quantum techniques.

[4] Shor, Peter W. "Algorithms for quantum computation: discrete logarithms and factoring." In Proceedings 35th annual symposium on foundations of computer science, pp. 124-134. Ieee, 1994.

Step 1: Pick a Random Number a

Choosing a for Coprimality

- Pick a random integer $1 < a < N$.
- Compute $\gcd(a, N)$.
- If $\gcd(a, N) \neq 1$, we are lucky! We have already found a nontrivial factor of N.
- Otherwise, proceed to Step 2.

Example: If $N = 15$ and we pick $a = 6$,

$$\gcd(6, 15) = 3$$

Thus, $p = 3$ and $q = 5$.

Step 2: Find the Period r

Finding the Period

- Find the smallest positive integer r such that:

$$a^r \equiv 1 \mod N$$

- Classically hard but efficient with quantum computers.
- Requirements:
  - r must be even.
  - $a^{r/2} \not\equiv -1 \mod N$.

Period r

What is Period r?

- Consider the function:
$$f(x) = a^x \mod N$$
- The **period** r is the smallest positive integer such that:
$$a^r \equiv 1 \mod N$$
- Powers of a modulo N repeat after r steps.
- Mathematically, r is the **order** of a in the group $\mathbb{Z}_N^*$.

Example: Finding the Period r

Example: Let $a = 2$, $N = 15$

- Compute:

x	$2^x \mod 15$
1	2
2	4
3	8
4	1

- Thus, $2^4 \equiv 1 \mod 15$
- Therefore, the period $r = 4$

Example: Finding the Period r for $a = 3$, $N = 17$

Powers of 3 Modulo 17

x	$3^x \mod 17$
1	3
2	9
3	10
4	13
5	5
6	15
7	11
8	16

x	$3^x \mod 17$
9	14
10	8
11	7
12	4
13	12
14	2
15	6
16	1

Thus, the period is:

$$\boxed{r = 16}$$

Why is Finding the Period Important?

- Once r is found:

$$a^r \equiv 1 \mod N \quad \Rightarrow \quad (a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \mod N$$

Note that,

$$a^r - 1 = \left(a^{r/2}\right)^2 - (1)^2 = \left(a^{r/2} - 1\right)\left(a^{r/2} + 1\right)$$

- If r is even and $a^{r/2} \not\equiv -1 \mod N$, we can compute:

$$\gcd(a^{r/2} - 1, N) \quad \text{or} \quad \gcd(a^{r/2} + 1, N)$$

- These yield nontrivial factors of N.

Thus, finding r is the key to factoring N efficiently!

Special Cases

- If r is odd, go back and pick a new a.
- If $a^{r/2} \equiv -1 \mod N$, pick a new a.
- With probability at least 50%, a good a can be chosen after a few tries.

Step 3: Use r to Factor N

Using the Period

- From Step 2, we know:
$$a^r \equiv 1 \mod N$$
- This implies:
$$a^r - 1 \equiv 0 \mod N$$
- Therefore, $a^r - 1$ is divisible by N:
$$a^r - 1 = kN$$
for some integer k.

Factoring $a^r - 1$

- Recall:
$$a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$$
- Thus:
$$(a^{r/2} - 1)(a^{r/2} + 1) = kpq$$
- At least one of $a^{r/2} - 1$ or $a^{r/2} + 1$ must share a nontrivial factor with N.

Finding the Factors

- Compute:

$$p = \gcd(a^{r/2} - 1, N)$$

$$q = \gcd(a^{r/2} + 1, N)$$

- These computations yield nontrivial factors p and q.

Thus, we have factored N.

Example: Factoring $N = 15$

- Pick $a = 2$.
- $\gcd(2, 15) = 1$, so continue to Step 2.
- Find the period of $2^x \mod 15$: $r = 4$.
- Compute:
$$a^{r/2} = 2^2 = 4$$
- Check:
$$a^{r/2} + 1 = 5 \quad (5 \mod 15 \neq 14)$$

Example Continued: Factoring $N = 15$

- Compute:

$$p = \gcd(4 - 1, 15) = \gcd(3, 15) = 3$$

$$q = \gcd(4 + 1, 15) = \gcd(5, 15) = 5$$

- Thus, $N = 3 \times 5$.

Successfully factored!

Bottleneck and Post-Quantum Cryptography

- **Bottleneck:** Finding the period r is classically hard.
- **Quantum Advantage:** Quantum computers can find periods efficiently.
- **Impact:** RSA and similar cryptosystems would be broken.
- **Post-Quantum Cryptography:** New cryptosystems resistant to quantum attacks are being developed.

Exercises

1. Use Shor's algorithm to factor $N = 35$.
   (a) Pick a value of a such that $\gcd(a, N) = 1$ so that we can continue with the remaining steps of the algorithm.
   (b) What is the period of $a^x \mod N$? Find the period classically. Ensure the period r is even and $a^{r/2} \not\equiv -1 \mod N$.
   (c) Calculate the factors $p = \gcd(a^{r/2} - 1, N)$ and $q = \gcd(a^{r/2} + 1, N)$.
2. Use Shor's algorithm to factor $N = 209$. Pick $a = 22$.
   (a) Show that $\gcd(a, N) \neq 1$.
   (b) What are the factors of N?

Question?