Overview of the RSA cryptosystem

Joshua Pité and Yiying Zhong
Mentor: Honglin Zhu

MIT PRIMES Conference

May 19, 2024
Outline

1. History
2. Mathematical preliminaries
3. Outline of the RSA algorithm
4. Security of RSA
5. Implementation of RSA
History

Early cryptography: private key. (Caesar cipher)

Modern cryptography: public key.

The RSA cryptosystem, was named after Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described it in 1977.
Early cryptography: private key. (Caesar cipher)
History

- Early cryptography: private key. (Caesar cipher)
- Modern cryptography: public key.
History

- Early cryptography: private key. (Caesar cipher)
- Modern cryptography: public key.
- The RSA cryptosystem, was named after Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described it in 1977.
## Mathematical preliminaries

- Modular arithmetic
- Euler totient and Euler’s theorem
Modular arithmetic

Congruence modulo $m$

Integers $a$ and $b$ are congruent modulo $m$ if $m$ divides their difference $a - b$. We denote it as $a \equiv b \pmod{m}$

Greatest common divisor

The greatest common divisor of integers $a$ and $b$, denoted $\gcd(a, b)$, is the largest integer that divides both $a$ and $b$.

Multiplicative inverse

An integer $b$ is the multiplicative inverse of $a$ modulo $m$ if:

$$ab \equiv 1 \pmod{m}$$

This exists if and only if $\gcd(a, m) = 1$. 
Euler’s totient function

Euler’s totient function \( \phi(n) \)

Counts the number of integers up to \( n \) that are relatively prime to \( n \)

\[
\phi(n) = \#\{m \in \mathbb{N} : 1 \leq m < n \text{ and } \gcd(m, n) = 1\}
\]

Examples of \( \phi(n) \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi(n) )</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>

Properties of \( \phi(n) \)

- If \( p \) is a prime number: \( \phi(p) = p - 1 \).
- If \( a \) and \( b \) are coprime: \( \phi(ab) = \phi(a)\phi(b) \).
Theorem (Euler)

If $N$ and $m$ are coprime, then

$$m^{\phi(N)} \equiv 1 \pmod{N},$$

This theorem generalizes Fermat’s little theorem, providing a fundamental reduction method for large powers in modular arithmetic.

Special cases used in RSA

- $N = pq$ with $p$ and $q$ primes.
- $\phi(N) = \phi(pq) = \phi(p)\phi(q) = (p - 1)(q - 1)$.
- $m^{(p-1)(q-1)} \equiv 1 \pmod{N}$. 
### Outline of the RSA Algorithm

<table>
<thead>
<tr>
<th>Alice</th>
<th>Eve</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key Creation</strong></td>
<td></td>
<td>Choose large primes $p$, $q$, and compute $N = p \cdot q$. Choose $e$, with $\gcd(e, (p - 1)(q - 1)) = 1$.</td>
</tr>
<tr>
<td>Create plaintext $m$.</td>
<td>Insecure ciphertext $c$.</td>
<td></td>
</tr>
<tr>
<td>Use known key $(N, e)$ to compute $c \equiv m^e \pmod{N}$.</td>
<td></td>
<td>Compute $d$ satisfying $ed \equiv 1 \pmod{(p - 1)(q - 1)}$.</td>
</tr>
<tr>
<td>Send ciphertext $c$ to Bob.</td>
<td></td>
<td>Compute $c^d \pmod{N}$:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$c^d \equiv m^{de}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\equiv m^{k(p-1)(q-1)+1}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\equiv m \pmod{N}$.</td>
</tr>
</tbody>
</table>
Security of RSA

- Security foundation
- Common uses of RSA
- Considerations for quantum computing
Basic security foundation of RSA

Problem 1: integer factorization

Given an integer $N$ promised to be a product of two large primes $p$ and $q$, find $p$ and $q$.

No known efficient (polynomial time) algorithm with classical computers.

Hard to obtain the decryption exponent $d$ from published public key $N$ alone.

Problem 2: RSA

Given $e$, $c$ and $N$, also with this equation known, find the value of $x$.

$$x^e \equiv c \pmod{N}$$

The security of the RSA relied on the assumption that it is hard to compute the $e$th roots modulo $N$. 

J. Pité and Y. Zhong
The RSA cryptosystem
Basic security foundation of RSA

Problem 1: integer factorization

Given an integer \( N \) promised to be a product of two large primes \( p \) and \( q \), find \( p \) and \( q \).
Basic security foundation of RSA

Problem 1: integer factorization

Given an integer $N$ promised to be a product of two large primes $p$ and $q$, find $p$ and $q$.

- No known efficient (polynomial time) algorithm with classical computers.
- Hard to obtain the decryption exponent $d$ from published public key $N$ alone.
Basic security foundation of RSA

Problem 1: integer factorization

Given an integer $N$ promised to be a product of two large primes $p$ and $q$, find $p$ and $q$.

- No known efficient (polynomial time) algorithm with classical computers.
- Hard to obtain the decryption exponent $d$ from published public key $N$ alone.

Problem 2: RSA

Given $e$, $c$ and $N$, also with this equation known, find the value of $x$.

$$x^e \equiv c \pmod{N},$$

- The security of the RSA relied on the assumption that it is hard to compute the $e$ th roots modulo $N$. 
Basic security foundation of RSA

Theorem

If the Problem 1 (integer factorization) is solved, Problem 2 (RSA) can also be solved.
Basic security foundation of RSA

Theorem

If the Problem 1 (integer factorization) is solved, Problem 2 (RSA) can also be solved.

- It is suspected, but not proved, that Problem 2 may be easier than Problem 1. (Boneh and Venkatesan)
- Thus, breaking RSA may be easier than solving integer factorization.
Common uses of RSA

- RSA is considered very secure and has been widely used, such as in data transmission, digital signature and private key exchange.

Advantages and limitations
Common uses of RSA

- RSA is considered very secure and has been widely used, such as in data transmission, digital signature and private key exchange.

Advantages and limitations

- **High security:** Provides strong security through the use of large keys and complex mathematical operations.

- **Computational intensity:** High computational demand because of the high-digit prime numbers, and the complex operations.
Considerations for quantum computing

Impact of quantum computing on RSA

Quantum computing could produce more efficient algorithms that break RSA. For example, Shor's algorithm is a quantum algorithm that solves integer factorization efficiently. For now, we cannot build sufficiently sophisticated quantum computers that execute these complex algorithms.
Considerations for quantum computing

Impact of quantum computing on RSA

- Quantum computing could produce more efficient algorithms that break RSA.
Considerations for quantum computing

Impact of quantum computing on RSA

- Quantum computing could produce more efficient algorithms that break RSA.
- For example, Shor’s algorithm is a quantum algorithm that solves integer factorization efficiently.
Considerations for quantum computing

Impact of quantum computing on RSA

- Quantum computing could produce more efficient algorithms that break RSA.
- For example, Shor’s algorithm is a quantum algorithm that solves integer factorization efficiently.
- For now, we cannot build sufficiently sophisticated quantum computers that execute these complex algorithms.
Implementation of RSA

- Finding prime numbers
- RSA demonstration
Finding prime numbers for RSA

Selecting primes $p$ and $q$

The security of RSA relies heavily on the choice of the two large prime numbers $p$ and $q$. These primes should be:

- Large enough to avoid trivial factorization;
- Randomly selected;
- Not too close to each other to prevent Fermat's factorization attack.
Finding prime numbers for RSA

Selecting primes $p$ and $q$

The security of RSA relies heavily on the choice of the two large prime numbers $p$ and $q$. These primes should be:

- Large enough to avoid trivial factorization;
Selecting primes \( p \) and \( q \)

The security of RSA relies heavily on the choice of the two large prime numbers \( p \) and \( q \). These primes should be:

- Large enough to avoid trivial factorization;
- Randomly selected;
Finding prime numbers for RSA

Selecting primes $p$ and $q$

The security of RSA relies heavily on the choice of the two large prime numbers $p$ and $q$. These primes should be:

- Large enough to avoid trivial factorization;
- Randomly selected;
- Not too close to each other to prevent Fermat’s factorization attack.
Primality testing is crucial for verifying if a randomly generated number is prime.
Finding prime numbers for RSA

Primality testing

Primality testing is crucial for verifying if a randomly generated number is prime.

- **Probabilistic tests**, like Miller-Rabin test, provide a high degree of certainty.
Finding prime numbers for RSA

Primality testing

Primality testing is crucial for verifying if a randomly generated number is prime.

- **Probabilistic tests**, like Miller-Rabin test, provide a high degree of certainty.
- **Deterministic tests**, like AKS, are used for conclusive results but are less efficient.
RSA demonstration

Python implementation example