Abstract Algebra

Ji Yong-Hyeon

Ji Yong-Hyeon
March 7, 2023

Contents

1 Public Key Cryptography in a Nutshell: Classification and Security Notions **1**
1.1 PKE 1

2 IFP-based Primitives **2**
2.1 Textbook RSA 2
2.1.1 The RSA Algorithm 2
2.1.2 Security 3
2.1.3 Conclusion 3
2.2 RSA-CRT 3
2.3 Fermat's Little Theorem 6
2.4 Euler's Theorem 8
2.5 Primality Test 9
2.6 EEA 10
2.7 CRT 10

Chapter 1

Public Key Cryptography in a Nutshell: Classification and Security Notions

1.1 PKE

Signature, Key Establishment
Onewayness Trapdoor, IND-CCA2, EUF-CMA

Chapter 2

IFP-based Primitives

2.1 Textbook RSA

RSA is a public-key cryptosystem that uses the mathematics of prime numbers to secure communication over the internet. It is widely used in various applications, such as digital signatures, secure email, and online banking.

The RSA cryptosystem is based on the following mathematical concepts:

- Modular arithmetic: - a mod b (remainder when a is divided by b) - Euler's totient function phi(n) (number of positive integers less than n that are coprime to n)
- Prime factorization: - finding the unique prime factors of a given integer
- The Chinese Remainder Theorem: - a theorem that provides a solution to a system of linear congruences with pairwise relatively prime moduli
- Fermat's Little Theorem: - a theorem that states that if p is a prime number and a is an integer not divisible by p, then $a^{(p-1)}$ is congruent to 1 modulo p.
- Euler's Theorem: - a generalization of Fermat's Little Theorem that states that if a and n are coprime, then $a^{(\phi(n))}$ is congruent to 1 modulo n.

2.1.1 The RSA Algorithm

Key Generation

To generate an RSA key pair, we follow these steps:

1. Choose two large prime numbers p and q.
2. Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
3. Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
4. Compute d such that $de \equiv 1 \pmod{\phi(n)}$.

The public key is (n, e), and the private key is d.

Encryption

To encrypt a message M, we use the public key (n, e) and compute:

$$C = M^e \pmod{n}$$

The ciphertext C is then sent to the recipient.

Decryption

To decrypt the ciphertext C, we use the private key d and compute:

$$M = C^d \pmod{n}$$

The plaintext M is then recovered.

2.1.2 Security

RSA is secure because it is based on the difficulty of factoring large integers. If an attacker can factor n into its prime factors p and q, then they can compute $\phi(n)$ and derive the private key. However, factoring large numbers is currently considered computationally infeasible, making RSA a secure choice for many applications.

2.1.3 Conclusion

RSA is a widely used public-key cryptosystem that uses prime numbers to secure communication over the internet. It is based on the difficulty of factoring large integers and is currently considered secure for many applications.

2.2 RSA-CRT

RSA-CRT is a modified version of the RSA public-key cryptosystem that leverages the Chinese Remainder Theorem to enhance the speed of the decryption process.

RSA-CRT works by first generating two large primes, p and q, and computing the modulus $n = pq$. Then, the public and private keys are generated using the usual RSA key generation algorithm.

When encrypting a message, the sender uses the recipient's public key to encrypt the message, as in standard RSA. However, when decrypting a message, the recipient first computes two intermediate values, m_1 and m_2, using the Chinese Remainder Theorem. These values are then combined to obtain the original message.

The Chinese Remainder Theorem states that given a system of linear congruences with pairwise relatively prime moduli, there exists a unique solution modulo the product of the moduli. In the case of RSA-CRT, the two moduli are p and q, which are both primes, and the system of congruences is:

$$m \equiv c^d \pmod{p} \; m \equiv c^d \pmod{q}$$

where c is the encrypted message, d is the recipient's private key, and m is the decrypted message.

The two intermediate values, m_1 and m_2, are computed as:

$$m_1 \equiv c^d \pmod{p} \; m_2 \equiv c^d \pmod{q}$$

These values can be computed efficiently using modular exponentiation, which is much faster than using the standard RSA decryption algorithm to compute m directly.

Finally, the original message can be recovered by combining m_1 and m_2 using the Chinese Remainder Theorem:

$$m \equiv m_1 + q((m_2 - m_1)q^{-1} \bmod p) \pmod{n}$$

where q^{-1} is the modular inverse of q modulo p. This formula allows the recipient to efficiently compute the original message without having to compute the expensive modular exponentiation step for the entire modulus n.

RSA-CRT Algorithm

1. Generate two large primes p and q, and compute $n = pq$.
2. Generate the public and private keys using the usual RSA key generation algorithm. That is,
   (a) Compute the totient of n, $\phi(n) = (p-1)(q-1)$.
   (b) Choose an integer e such that
   $$1 < e < \phi(n) \quad \text{and} \quad \gcd(e, \phi(n)) = 1.$$
   This is the public key exponent.
   (c) Compute the private key exponent d such that
   $$d \equiv e^{-1} \pmod{\phi(n)}.$$
   This can be done efficiently using the Extended Euclidean Algorithm.

   The public key is (n, e), and the private key is (n, d).
3. To encrypt a message $\mathcal{M}$, use the recipient's public key to compute
   $$C = \mathcal{M}^e \mod n.$$
4. To decrypt a message C, compute the two intermediate values:
   (a) $m_1 = C^d \mod p$.
   (b) $m_2 = C^d \mod q$.
5. Combine the two intermediate values using the Chinese Remainder Theorem to obtain the original message $\mathcal{M}$:
   $$\mathcal{M} = m_1 + q\left((m_2 - m_1)\, q^{-1} \mod p\right) \pmod{n}.$$

Chinese Remainder Theorem

Theorem 1. *Given a system of linear congruences with pairwise relatively prime moduli:*

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_n \pmod{m_1}, \end{aligned}$$

there exists a unique solution x modulo $M = m_1 m_2 \cdots m_n$:

$$x \equiv a_1 b_1 M_1 + a_2 b_2 M_2 + \cdots a_n b_n M_n \pmod{M}, \text{i.e.,}$$
$$x \equiv \sum_{i=1}^{n} a_i b_i M_i \pmod{M}$$

where $M_i = M/m_i$ and b_i is the inverse of M_i modulo m_i.

2.3 Fermat's Little Theorem

Fermat's Little Theorem

Theorem 2. *Let p is a prime number and a is an integer not divisible by p, then*

$$a^{p-1} \equiv 1 \pmod{p}.$$

In other words, if we take any non-zero integer a and raise it to the power $p-1$, then divide the result by a prime p, the remainder will always be 1. This theorem is widely used in number theory and cryptography.

Proof. We use mathematical induction and binomial theorem.

1. (Basic Step) Let $a = 1$, then $1^{p-1} = 1 \equiv \pmod{p}$, which is true.
2. (Inductive Step) Suppose that the theorem holds for some integer a. We need to show that it also holds for $a+1$. We can express $(a+1)^p$ as:

$$(a+1)^p = \sum_{i=0}^{p} \binom{p}{i} a^i 1^{p-i}$$

□

Using the binomial theorem. Since p is a prime number, we know that $\binom{p}{i}$ is divisible by p for $0 < i < p$, hence:

$$(a+1)^p \equiv a^p + 1^p \equiv a + 1 \pmod{p}$$

The last step follows from the fact that $a^{p-1} \equiv 1 \pmod{p}$. Therefore, we have shown that:

$$(a+1)^{p-1} \equiv 1 \pmod{p}$$

which completes the proof of Fermat's Little Theorem.

Proof. We use the group theory of the multiplicative group of integers modulo a prime p.

Let p be a prime number, and let a be an integer that is not divisible by p. Consider the set of integers modulo p, denoted by $\mathbb{Z}/p\mathbb{Z}$. This set forms a group under multiplication, denoted by $(\mathbb{Z}/p\mathbb{Z})^\times$. Since p is prime, the group $(\mathbb{Z}/p\mathbb{Z})^\times$ has order $p-1$, which means that it contains $p-1$ distinct elements.

Now consider the subset of $(\mathbb{Z}/p\mathbb{Z})^\times$ consisting of the multiples of a, denoted by $S_a = a, 2a, 3a, \ldots, (p-1)a$. Since a is not divisible by p, the set S_a consists of distinct elements. We claim that S_a is a permutation of the elements of $(\mathbb{Z}/p\mathbb{Z})^\times$.

To prove this claim, we need to show that every element of $(\mathbb{Z}/p\mathbb{Z})^\times$ can be expressed as a multiple of a modulo p. Suppose for the sake of contradiction that there exists an element $b \in (\mathbb{Z}/p\mathbb{Z})^\times$ that cannot be expressed as $b = ka \pmod{p}$ for any integer k. Then the integers $1a, 2a, \ldots, (p-1)a$ would not be distinct modulo p, which contradicts our assumption that S_a consists of distinct elements. Therefore, S_a is a permutation of the elements of $(\mathbb{Z}/p\mathbb{Z})^\times$.

Since S_a is a permutation of the elements of $(\mathbb{Z}/p\mathbb{Z})^\times$, we can multiply all the elements of S_a together to obtain:

$$a \cdot 2a \cdot \ldots \cdot (p-1)a \equiv 1 \cdot 2 \cdot \ldots \cdot (p-1) \pmod{p}$$

which simplifies to:

$$(p-1)!a^{p-1} \equiv (p-1)! \pmod{p}$$

Since p does not divide $(p-1)!$, we can cancel out $(p-1)!$ from both sides to obtain:

$$a^{p-1} \equiv 1 \pmod{p}$$

This completes the proof of Fermat's Little Theorem using the theory of abstract algebra. □

2.4 Euler's Theorem

Euler's Theorem is a generalization of Fermat's Little Theorem. It states that if a and n are two positive integers that are coprime, then:

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

where $\varphi(n)$ is Euler's totient function, which gives the number of positive integers less than or equal to n that are coprime with n. Euler's Theorem is useful in cryptography for testing the primality of large numbers and for computing modular exponentiations efficiently.

Proof: We can prove Euler's Theorem using the fact that the totient function is multiplicative, which means that if m and n are two coprime positive integers, then:

$$\varphi(mn) = \varphi(m)\varphi(n)$$

We can use this property to reduce the theorem to the case where n is a prime power p^k, where p is a prime and k is a positive integer. If $k = 1$, then Euler's Theorem reduces to Fermat's Little Theorem. Otherwise, we can use the Chinese Remainder Theorem to combine the solutions of $a^{\varphi(p^k)} \equiv 1 \pmod{p^k}$ for all prime powers p^k that divide n. Since the exponents $\varphi(p^k)$ are powers of p, we can use repeated squaring to compute the modular exponentiations efficiently. This completes the proof of Euler's Theorem.

Proof. Euler's Theorem is a generalization of Fermat's Little Theorem and states that if a and n are two coprime positive integers, then

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

where $\varphi(n)$ is Euler's totient function, which gives the number of positive integers less than or equal to n that are coprime with n.

To prove Euler's Theorem, we first define a group G of integers modulo n that are coprime with n. This group is denoted as:

$$G = a \in \mathbb{Z}_n | gcd(a, n) = 1$$

Next, we define the function $f(a) = ra \pmod{n}$ where r is a fixed integer coprime with n. It can be shown that f is a permutation of G and that the order of the group G is $\varphi(n)$.

Now, we consider the product of all elements in G:

$$P = \prod_{a \in G} f(a) = \prod_{a \in G} ra \pmod{n}$$

We can rewrite this product as:

$$P = r^{\varphi(n)} \prod_{a \in G} a \pmod{n}$$

Since a is coprime with n, we know that a has a unique inverse b in the group G such that $ab \equiv 1 \pmod{n}$. Therefore, we can rewrite the product as:

$$P = r^{\varphi(n)} \prod_{a \in G} a \prod_{a \in G} b = r^{\varphi(n)} \prod_{a \in G} 1 = r^{\varphi(n)}$$

where we used the fact that $ab \equiv 1 \pmod{n}$ and that a and b are both in the group G.

On the other hand, we can rearrange the terms in the product as:

$$P = \prod_{a \in G} f(a) = \prod_{a \in G} (ar) \pmod{n} = r^{\varphi(n)} \prod_{a \in G} a \pmod{n}$$

where we used the fact that ar is also in the group G since r is coprime with n.

Equating the two expressions for P, we get:

$$r^{\varphi(n)} \prod_{a \in G} a \equiv r^{\varphi(n)} \prod_{a \in G} a \pmod{n}$$

Dividing both sides by $\prod_{a \in G} a$, we obtain:

$$r^{\varphi(n)} \equiv 1 \pmod{n}$$

which is Euler's Theorem. □

2.5 Primality Test

A primality test is a method used to determine if a given positive integer is a prime number or a composite number. There are various primality tests, and they differ in their speed, accuracy, and the range of numbers they can handle.

One of the simplest and most well-known primality tests is the trial division method. This method involves dividing the number by each integer from 2 up to the square root of the number, checking if any of the divisors evenly divide the number. If no divisor is found, then the number is prime. However, this method becomes impractical for very large numbers, as the number of potential divisors to check grows with the number being tested.

Another primality test is the Fermat primality test. This test is based on Fermat's Little Theorem, which states that if p is a prime number and a is any integer not divisible by p, then $a^{p-1} \equiv 1 \pmod{p}$. The Fermat primality test uses this theorem to check if a number is prime by randomly selecting values of a and checking if the equation holds for each value. If the equation fails for any a, then the number is composite. If the equation holds for many values of a, the number is likely prime, but there is still a small chance it could be composite.

The Miller-Rabin primality test is a more sophisticated primality test that is based on the same idea as the Fermat test but is more efficient and has a higher probability of correctly identifying composite numbers. The Miller-Rabin test involves selecting a random value a and then repeatedly squaring it and checking if the resulting values satisfy the equation $a^d \equiv 1 \pmod{n}$ or $a^{2^r d} \equiv -1 \pmod{n}$ for some values of r and d. If the equation holds for many values of r and d, the number is likely prime, but if the equation fails for any r or d, the number is composite.

There are also deterministic primality tests that can determine with certainty whether a number is prime or composite. One example is the AKS primality test, which is based on a polynomial-time algorithm and can handle very large numbers. However, deterministic tests are generally more complex and slower than probabilistic tests.

2.6 EEA

2.7 CRT

IFP-based Schemes
RSA-OAEP, RSA-PSS
Random Oracle Model
How to Implement IFP-based Schemes Part 1
Integer Multiplication, Division, (Modular) Exponentiation
How to Implement IFP-based Schemes Part 2
Barrett Reduction, Montgomery Reduction
How to Solve IFP Part 1
Birthday Bound, Floyd's Cycle Detection
Pollard's p-1 method, Pollard's rho method
How to Solve IFP Part 2
Quadratic Residue mod p (Legendre symbol), Square Roots modulo p
QS, GNFS
Midterm Exam. (04.20.)
DLP-based Schemes
DH, DSA, KCDSA
How to Solve DLP Part 1
Baby-Step/Giant-Step Algorithm, Pollard's rho method
How to Solve DLP Part 2
Pohlig-Hellman Algorithm, Index Calculus Method
Elliptic Curve Cryptography Part 1
Projective Space, Elliptic Curve, Elliptic Curve Group
ECDH, ECDSA, EC-KCDSA
Elliptic Curve Cryptography Part 2
Addition, Doubling
NIST Curves, Curve25519
Post-Quantum Cryptography
NIST PQC, KpqC