二次剩余探究报告

2212000 宋奕纬

June 2024

1 引言

二次剩余（Quadratic Residue）是数论中的一个重要概念，对许多数学问题和密码学应用都有重要影响。本文将详细介绍二次剩余的定义、性质、Blum 整数，以及二次剩余在密码学中的应用。

2 相关定义与性质

2.1 二次剩余定义

在模 n 的算术中，整数 a 称为模 n 的二次剩余，如果存在整数 x 使得 $x^2 \equiv a \pmod{n}$。换句话说，a 是某个整数平方的同余类。

- **二次剩余**：存在整数 x 使得 $x^2 \equiv a \pmod{n}$，则称 a 为模 n 的二次剩余。
- **非二次剩余**：如果不存在这样的整数 x，则称 a 为模 n 的非二次剩余。

2.2 勒让德符号

勒让德符号 $\left(\frac{a}{p}\right)$ 用于判断一个数是否为模素数 p 的二次剩余，定义如下：

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{如果 } a \equiv 0 \pmod{p} \\ 1 & \text{如果 } a \text{ 是模 } p \text{ 的二次剩余} \\ -1 & \text{如果 } a \text{ 是模 } p \text{ 的非二次剩余} \end{cases}$$

勒让德符号的性质：

- 乘法性质：$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right)=\left(\frac{ab}{p}\right)$。
- 欧拉准则：$\left(\frac{a}{p}\right)\equiv a^{(p-1)/2} \pmod{p}$。
- 互反律：$\left(\frac{a}{p}\right)\left(\frac{p}{a}\right)=(-1)^{\frac{(p-1)(a-1)}{4}}$。

2.3 雅可比符号

雅可比符号 $\left(\frac{a}{n}\right)$ 是勒让德符号的推广，用于模合数 n 的情形，定义如下：

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\cdots\left(\frac{a}{p_k}\right)^{e_k}$$

其中 $n=p_1^{e_1}p_2^{e_2}\cdots p_k^{e_k}$ 是 n 的素因子分解。

雅可比符号的性质：

- 若 n 为奇数，$\left(\frac{a}{n}\right)\equiv a^{(n-1)/2} \pmod{n}$。
- 乘法性质：$\left(\frac{a}{n}\right)\left(\frac{b}{n}\right)=\left(\frac{ab}{n}\right)$。
- 如果 $a\equiv b \pmod{n}$，则 $\left(\frac{a}{n}\right)=\left(\frac{b}{n}\right)$。

2.4 Blum 整数

Blum 整数是指两个形如 $p\equiv 3 \pmod{4}$ 和 $q\equiv 3 \pmod{4}$ 的素数的乘积，即 $n=p\cdot q$。Blum 整数在密码学中有重要应用。

Blum 整数的性质

设 $n=p\cdot q$ 为一个 Blum 整数，则 $x^2\equiv a \pmod{n}$ 的解可以由模 p 和模 q 的解通过中国剩余定理得到。具体地，若 x_0 是 $x^2\equiv a \pmod{p}$ 的解，y_0 是 $x^2\equiv a \pmod{q}$ 的解，则 $x^2\equiv a \pmod{n}$ 有以下四个解：

$$x\equiv \pm x_0 \pmod{p}$$

$$x\equiv \pm y_0 \pmod{q}$$

这些解的具体形式为：

$$x \equiv \pm x_0 \pmod{n}$$

$$x \equiv \pm y_0 \pmod{n}$$

3 二次剩余的判断

3.1 模素数的二次剩余

利用勒让德符号可以判断一个数是否为模素数 p 的二次剩余。若 $\left(\frac{a}{p}\right) = 1$，则 a 是模 p 的二次剩余；若 $\left(\frac{a}{p}\right) = -1$，则 a 是模 p 的非二次剩余。

```cpp
#include <iostream>
#include <cmath>

using namespace std;

// 计算勒让德符号
int legendre(int a, int p) {
    if (a % p == 0) return 0;
    int result = 1;
    if (a < 0) {
        a = -a;
        if (p % 4 == 3) result = -result;
    }
    while (a != 0) {
        while (a % 2 == 0) {
            a /= 2;
            if (p % 8 == 3 || p % 8 == 5) result = -result;
        }
        swap(a, p);
        if (a % 4 == 3 && p % 4 == 3) result = -result;
        a %= p;
    }
    return (p == 1) ? result : 0;
}

int main() {
    int a = 10;
    int p = 13;
    int result = legendre(a, p);
    if (result == 1) {
        cout << a << " is a quadratic residue modulo " << p << endl;
    } else if (result == -1) {
```

```cpp
        cout << a << " is not a quadratic residue modulo " << p << endl;
    } else {
        cout << a << " is divisible by " << p << endl;
    }
    return 0;
}
```

3.2 模合数的二次剩余

对于合数，可以利用雅可比符号进行判断。如果 $\left(\frac{a}{n}\right) \neq a^{(n-1)/2} \pmod{n}$，则 a 不是模 n 的二次剩余。

```cpp
#include <iostream>
#include <cmath>

using namespace std;

// 计算雅可比符号
int jacobi(int a, int n) {
    if (a == 0) return 0;
    if (a == 1) return 1;
    int ans;
    if (a % 2 == 0) {
        ans = jacobi(a / 2, n);
        if (n % 8 == 3 || n % 8 == 5) ans = -ans;
    } else {
        ans = jacobi(n % a, a);
        if (a % 4 == 3 && n % 4 == 3) ans = -ans;
    }
    return ans;
}

int main() {
    int a = 10;
    int n = 21;
    int result = jacobi(a, n);
    if (result == 1) {
        cout << a << " is a quadratic residue modulo " << n << endl;
    } else if (result == -1) {
        cout << a << " is not a quadratic residue modulo " << n << endl;
    } else {
        cout << a << " is divisible by " << n << endl;
    }
    return 0;
}
```

3.3 中国剩余定理

对于合数 $n = p \cdot q$，可以通过分别判断 a 是否是模 p 和模 q 的二次剩余，然后利用中国剩余定理来综合判断。

```cpp
#include <iostream>
#include <cmath>

using namespace std;

// 计算勒让德符号
int legendre(int a, int p) {
    if (a % p == 0) return 0;
    int result = 1;
    if (a < 0) {
        a = -a;
        if (p % 4 == 3) result = -result;
    }
    while (a != 0) {
        while (a % 2 == 0) {
            a /= 2;
            if (p % 8 == 3 || p % 8 == 5) result = -result;
        }
        swap(a, p);
        if (a % 4 == 3 && p % 4 == 3) result = -result;
        a %= p;
    }
    return (p == 1) ? result : 0;
}

// 判断a是否为模p和模q的二次剩余
bool isQuadraticResidue(int a, int p, int q) {
    return legendre(a, p) == 1 && legendre(a, q) == 1;
}

int main() {
    int a = 10;
    int p = 7;
    int q = 11;
    if (isQuadraticResidue(a, p, q)) {
        cout << a << " is a quadratic residue modulo " << p << " and " << q
    << endl;
    } else {
        cout << a << " is not a quadratic residue modulo " << p << " and "
    << q << endl;
    }
    return 0;
```

```
}
```

4 二次剩余的应用

4.1 密码学

二次剩余在密码学中有着广泛的应用，特别是在构造公钥加密算法和零知识证明时。例如，Rabin 加密算法和 Goldwasser-Micali 加密算法都利用了二次剩余的性质进行消息加密和解密。

4.2 Rabin 加密算法

Rabin 加密算法基于 Blum 整数。其安全性依赖于整数分解的难题。加密过程如下：

- 选择两个大素数 p 和 q，满足 $p \equiv 3 \pmod 4$ 和 $q \equiv 3 \pmod 4$。
- 计算 $n = p \cdot q$。
- 公钥为 n，私钥为 (p, q)。
- 对于明文 m，加密为 $c \equiv m^2 \pmod n$。

解密时，利用中国剩余定理，从 c 的四个平方根中选出正确的明文 m。以下是 Rabin 加密算法的 C++ 实现：

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cmath>
#include <vector>

using namespace std;

// 计算最大公约数
int gcd(int a, int b) {
    if (b == 0) return a;
    return gcd(b, a % b);
}

// 快速幂模运算
long long power(long long base, long long exp, long long mod) {
```

```
    long long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}

// Rabin加密算法的加密函数
long long rabinEncrypt(long long m, long long n) {
    return (m * m) % n;
}

// Rabin加密算法的解密函数
vector<long long> rabinDecrypt(long long c, long long p, long long q) {
    long long n = p * q;
    long long mp = power(c, (p + 1) / 4, p);
    long long mq = power(c, (q + 1) / 4, q);

    long long y1 = (mp * q * power(q, p - 2, p) + mq * p * power(p, q - 2, q)) % n;
    long long y2 = (mp * q * power(q, p - 2, p) - mq * p * power(p, q - 2, q)) % n;
    long long y3 = n - y1;
    long long y4 = n - y2;

    vector<long long> roots = {y1, y2, y3, y4};
    return roots;
}

int main() {
    long long p = 7;
    long long q = 11;
    long long n = p * q;
    long long m = 20;

    long long c = rabinEncrypt(m, n);
    cout << "Encrypted message: " << c << endl;

    vector<long long> decrypted = rabinDecrypt(c, p, q);
    cout << "Decrypted messages: ";
    for (long long root : decrypted) {
```

```
        cout << root << " ";
    }
    cout << endl;

    return 0;
}
```

4.3 Goldwasser-Micali 加密算法

Goldwasser-Micali 加密算法利用二次剩余和非二次剩余的性质实现加密。其安全性依赖于二次剩余判定的难题。具体过程如下：

- 选择一个 Blum 整数 $n = p \cdot q$。
- 选择一个非二次剩余 y。
- 公钥为 (n, y)，私钥为 (p, q)。
- 对于每个比特 b，选择一个随机数 r，加密为 $c \equiv r^2 \cdot y^b \pmod{n}$。

解密时，利用私钥 (p, q) 判断 c 是否为二次剩余即可恢复比特 b。以下是 Goldwasser-Micali 加密算法的 C++ 实现：

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <cmath>

using namespace std;

// 计算勒让德符号
int legendre(int a, int p) {
    if (a % p == 0) return 0;
    int result = 1;
    if (a < 0) {
        a = -a;
        if (p % 4 == 3) result = -result;
    }
    while (a != 0) {
        while (a % 2 == 0) {
            a /= 2;
            if (p % 8 == 3 || p % 8 == 5) result = -result;
        }
        swap(a, p);
```

```cpp
        if (a % 4 == 3 && p % 4 == 3) result = -result;
        a %= p;
    }
    return (p == 1) ? result : 0;
}

// 判断是否为二次剩余
bool isQuadraticResidue(int a, int p, int q) {
    return legendre(a, p) == 1 && legendre(a, q) == 1;
}

// Goldwasser-Micali加密
int gmEncrypt(int b, int n, int y) {
    srand(time(0));
    int r = rand() % n;
    int c = (r * r) % n;
    if (b == 1) {
        c = (c * y) % n;
    }
    return c;
}

// Goldwasser-Micali解密
int gmDecrypt(int c, int p, int q) {
    if (isQuadraticResidue(c, p, q)) {
        return 0;
    } else {
        return 1;
    }
}

int main() {
    int p = 7;
    int q = 11;
    int n = p * q;
    int y = 5; // 一个非二次剩余

    int b = 1; // 待加密的比特

    int c = gmEncrypt(b, n, y);
    cout << "Encrypted bit: " << c << endl;

    int decrypted = gmDecrypt(c, p, q);
    cout << "Decrypted bit: " << decrypted << endl;

    return 0;
```

```
}
```

5 总结

二次剩余是数论中的重要概念，在密码学中具有广泛的应用。通过利用二次剩余和 Blum 整数的性质，可以构造高效且安全的加密算法，如 Rabin 加密和 Goldwasser-Micali 加密。理解和研究二次剩余的性质，将有助于在数论和密码学领域取得更大的进展。