

密码学实验三 RSA 加密算法

08111505 1120152056 李世林

密码学实验三 RSA 加密算法	1
1 算法原理	2
1.1 简介	2
1.2 RSA 加密	2
1.3 RSA 解密	2
2 C++实现 RSA 加密解密算法	4
2.1 流程图	4
2.2 RSA 加密算法详解.....	5
2.2.1 扩展 gcd 计算逆元.....	5
2.2.2 Inverse 计算 $a \bmod$ 的逆元.....	5
2.2.3 快速幂运算加密	6
2.3 RSA 解密	7
2.3.1 扩展 gcd 计算逆元.....	7
2.3.2 Inverse 计算 $a \bmod$ 的逆元.....	8
2.3.3 快速幂运算解密	8
2.4 工具函数	9
2.5 主函数和加解密函数.....	9
2.6 运行结果	13
3 实验总结	13
4 附完整代码	14

1 算法原理

1.1 简介

RSA 算法是非对称密码算法中非常经典的一种算法，使用率非常高，一般用于数据加密和数字签名。

RSA 算法加密的过程首先由接收方实例化密钥对，然后将自己的公钥公布出去，这就相当于告诉发送方，如果你要给我发送数据，请使用该公钥对明文进行加密，当接收方收到用公钥加密过后的明文后，需要使用配套的私钥进行解密，又因为该私钥只有接收方自己才有，所以就算数据在传输的过程中被黑客截取，他也不可能将数据破译出来。

1.2 RSA 加密

RSA 的加密过程可以使用一个通式来表达

密文 = 明文 $E \bmod N$

从通式可知，只要知道 E 和 N 任何人都可以进行 RSA 加密了，所以说 E、N 是 RSA 加密的密钥，也就是说 E 和 N 的组合就是公钥，我们用 (E, N) 来表示公钥

公钥 = (E, N)

1.3 RSA 解密

RSA 的解密同样可以使用一个通式来表达

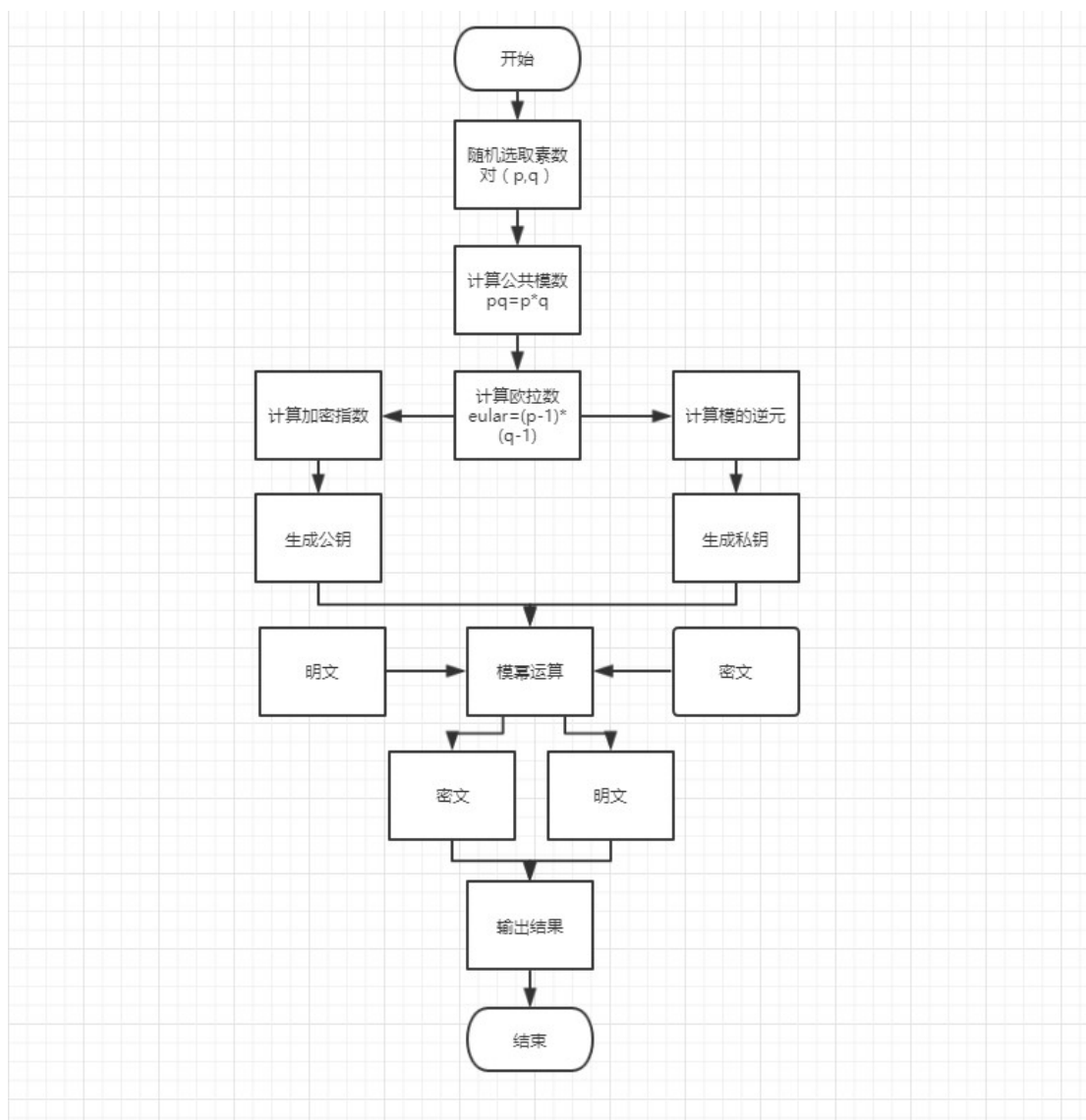
明文 = 密文 $D \bmod N$

也就是说对密文进行 D 次方后除以 N 的余数就是明文，这就是 RSA 解密过程。知道 D 和 N 就能进行解密密文了，所以 D 和 N 的组合就是私钥

私钥 = (D, N)

2C++实现 RSA 加密解密算法

2.1 流程图



2. 2RAS 加密算法详解

2.2.1 扩展 gcd 计算逆元

```
//扩展 gcd 算法，用来求逆元
ll Extended_gcd(ll a, ll b, ll &x, ll &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }

    ll d = Extended_gcd(b, a%b, x, y);

    ll t = x;
    x = y;
    y = t - a / b*y;

    return d;
}
```

2.2.2 Inverse 计算 $a\%mod$ 的逆元

```
//计算  $a\%mod$  的逆元
```

```
ll Inverse(ll a)
{
    ll x, y;
    Extended_gcd(a, mod, x, y);
    return (x%mod + mod) % mod;
}
```

2.2.3 快速幂运算加密

```
//快速幂取模, 计算  $x^k$ 
int Pow(int a, int b)
{
    int ans = 1;
    a = a%mod;
    while (b != 0)
    {
        if (b & 1)
            ans = (ans*a) % mod;
        b >>= 1;
        a = (a*a) % mod;
    }
    return ans;
}
```

```
}
```

2.3 RSA 解密

2.3.1 扩展 gcd 计算逆元

```
//扩展 gcd 算法，用来求逆元
ll Extended_gcd(ll a, ll b, ll &x, ll &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    ll d = Extended_gcd(b, a%b, x, y);
    ll t = x;
    x = y;
    y = t - a / b*y;
    return d;
}
```

2.3.2 Inverse 计算 $a \% \text{mod}$ 的逆元

```
//计算 a%mod 的逆元  
ll Inverse(ll a)  
{  
    ll x, y;  
    Extended_gcd(a, mod, x, y);  
    return (x%mod + mod) % mod;  
}
```

2.3.3 快速幂运算解密

```
//快速幂取模, 计算  $x^k$   
int Pow(int a, int b)  
{  
    int ans = 1;  
    a = a%mod;  
    while (b != 0)  
    {  
        if (b & 1)  
            ans = (ans*a) % mod;  
        b >>= 1;  
        a = (a*a) % mod;  
    }  
}
```



```
    }  
  
    return ans;  
}
```

2.4 工具函数

```
//判断是否为质数  
bool isPrime(ll x)  
{  
    if (x == 1) return false;  
    for (int i = x - 1; i >= sqrt(x) && i >= 2; i--)  
    {  
        if (x%i == 0)  
            return false;  
    }  
    return true;  
}
```

2.5 主函数和加解密函数

```
//加密, 输入为明文, 返回密文  
ll encryption(ll plain)
```

```

{
    mod = pq;
    return Pow(plain, rande);
}

//解密，输入为密文，私钥，返回明文
ll Decrypt(ll cliper, ll private_key)
{
    mod = pq;
    return Pow(cliper, private_key);
}

int main()
{
    cout << "RSA 加解密算法" << endl;
    cout << "\n 请输入素数 p,q(用空格隔开): " << endl;
    cout << "素数 p,q: ";
    cin >> p >> q;
    //判断 p,q 是否均为素数
    if (!isPrime(p) || !isPrime(q))
    {

```

```
    cout << "p,q 不为素数" << endl;

    cout << "请按 Enter 键结束" << endl;

    cin.clear();

    cin.sync();

    cin.get();

    cin.get();

    return 0;

}

euler = (p - 1)*(q - 1);

pq = p*q;

cout << "请选择一个随机数 rande" << "（小于" << euler <<
"且与之互质）" << endl;

cout << "随机数 rande: ";

cin >> rande;

ll x, y;

//计算逆元

ll temp = Extended_gcd(rande, euler, x, y);

if (temp != 1)

{

    cout << "错误!!! rande 与" << euler << "不互质" <<
endl;

    cout << "请按 Enter 键结束" << endl;
```

```
    cin.clear();

    cin.sync();

    cin.get();

    cin.get();

    return 0;

}

mod = euler;

privateKey = Inverse(rande);

cout << "公钥为(" << rande << ", " << pq << ")" << endl;
cout << "私钥为" << privateKey << endl;
cout << "请输入明文(限数字)" << endl;
cout << "明文: ";

ll m, c, tmp;//明文, 密文, 解密之后的明文
cin >> m;

c = encryption(m);

cout << "密文为" << c << endl;

cout << "-----解密-----" << endl;

tmp = Decrypt(c, privateKey);

cout << "明文为" << tmp << endl;


cout << "请按 Enter 键结束" << endl;

cin.clear();
```

```
    cin.sync();

    cin.get();

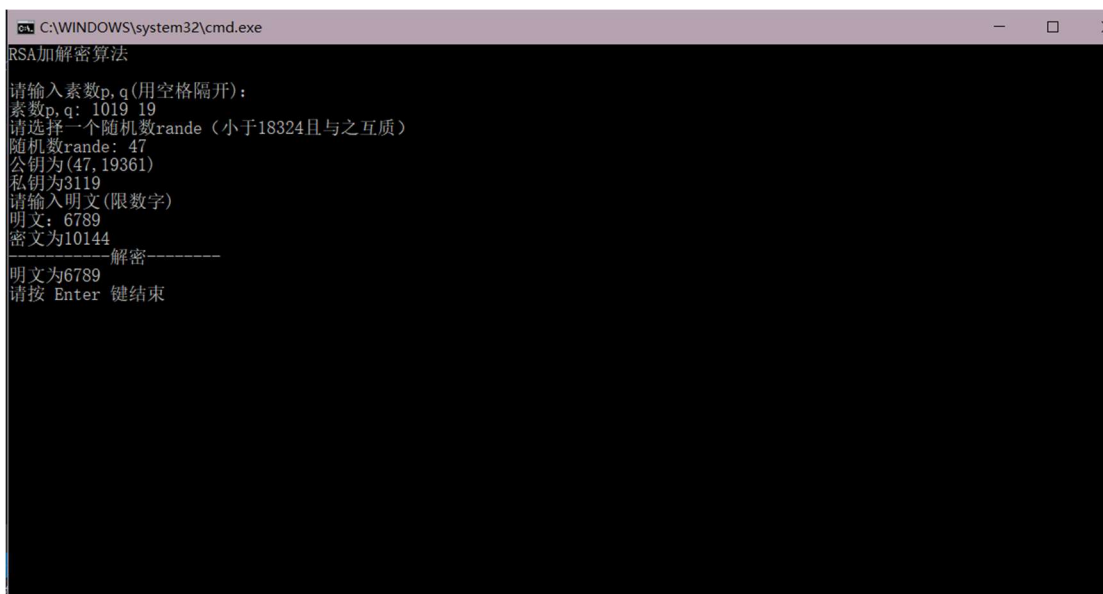
    cin.get();

    return 0;

    return 0;

}
```

2.6 运行结果



```
C:\WINDOWS\system32\cmd.exe
RSA加解密算法
请输入素数p, q (用空格隔开):
素数p, q: 1019 19
请选择一个随机数rande (小于18324且与之互质)
随机数rande: 47
公钥为 (47, 19361)
私钥为3119
请输入明文 (限数字)
明文: 6789
密文为10144
-----解密-----
明文为6789
请按 Enter 键结束
```

3 实验总结

RSA 加密和解密算法是建立在数论的基础上的，因为处理素数首先需要面对的是判断一个数是不是素数，同时还有可能面对非常大的整数，

因此会用到大整数运算以及一些数学技巧，比如快速幂等来简化计算，提高计算效率。因为能力有限，此次实验仅实现了 C++long long 即 64 位整数之间的加密和解密，对于大整数则无法进行。

4 附完整代码

```
#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<iostream>

using namespace std;

#define ll long long

ll p, q;//两个大素数

ll pq;//pq=p*q

ll rande;//随机数

ll mod;//取模的 mod

ll privateKey;//私钥

ll euler;//欧拉函数值
```

//判断是否为质数

```
bool isPrime(ll x)
{
    if (x == 1) return false;
    for (int i = x - 1; i >= sqrt(x) && i >= 2; i--)
    {
        if (x%i == 0)
            return false;
    }
    return true;
}
```

//快速幂取模, 计算 x^k

```
int Pow(int a, int b)
{
    int ans = 1;
    a = a%mod;
    while (b != 0)
    {
        if (b & 1)
            ans = (ans*a) % mod;
        b >>= 1;
    }
}
```

```

        a = (a*a) % mod;

    }

    return ans;
}

//扩展 gcd 算法, 用来求逆元
ll Extended_gcd(ll a, ll b, ll &x, ll &y)
{
    if (b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }

    ll d = Extended_gcd(b, a%b, x, y);

    ll t = x;

    x = y;
    y = t - a / b*y;

    return d;
}

//计算 a%mod 的逆元

```



```
ll Inverse(ll a)
{
    ll x, y;
    Extended_gcd(a, mod, x, y);
    return (x%mod + mod) % mod;
}

//加密, 输入为明文, 返回密文
ll encryption(ll plain)
{
    mod = pq;
    return Pow(plain, rande);
}

//解密, 输入为密文, 私钥, 返回明文
ll Decrypt(ll cliper, ll private_key)
{
    mod = pq;
    return Pow(cliper, private_key);
}
```

```
int main()
{
    cout << "RSA 加解密算法" << endl;
    cout << "\n 请输入素数 p, q(用空格隔开): " << endl;
    cout << "素数 p, q: ";
    cin >> p >> q;
    //判断 p, q 是否均为素数
    if (!isPrime(p) || !isPrime(q))
    {
        cout << "p, q 不为素数" << endl;
        cout << "请按 Enter 键结束" << endl;
        cin.clear();
        cin.sync();
        cin.get();
        cin.get();
        return 0;
    }

    euler = (p - 1)*(q - 1);
    pq = p*q;

    cout << "请选择一个随机数 rande" << " (小于" << euler <<
    "且与之互质)" << endl;
```

```
cout << "随机数 rande: ";

cin >> rande;

ll x, y;

//计算逆元

ll temp = Extended_gcd(rande, euler, x, y);

if (temp != 1)
{
    cout << "错误!!! rande 与" << euler << "不互质" <<
endl;

    cout << "请按 Enter 键结束" << endl;

    cin.clear();

    cin.sync();

    cin.get();

    cin.get();

    return 0;

}

mod = euler;

privateKey = Inverse(rande);

cout << "公钥为(" << rande << ", " << pq << ")" << endl;

cout << "私钥为" << privateKey << endl;

cout << "请输入明文(限数字)" << endl;

cout << "明文: ";
```

```
ll m, c, tmp;//明文，密文，解密之后的明文

cin >> m;

c = encryption(m);

cout << "密文为" << c << endl;

cout << "-----解密-----" << endl;

tmp = Decrypt(c, privateKey);

cout << "明文为" << tmp << endl;


cout << "请按 Enter 键结束" << endl;

cin.clear();

cin.sync();

cin.get();

cin.get();

return 0;

return 0;

}
```