# RSA中公开的模数N 实验报告

学号: SA20225172 姓名: 郭俊勇

### 实验目的

- 了解公钥加密方案的一般结构
- 深入理解RSA加密原语的密钥生成
- 编程实现对没有正确生成密钥的RSA的破解,提醒大家不要尝试自己随意实现加密原语

#### 编程语言

Java

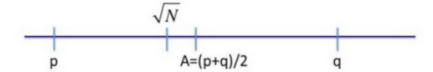
### 实验内容

- 本次实验是在公开的模数N没有被正确生成时破解RSA。这个实验是在提醒大家,千万不要自己轻易去实现一个加密原语。
- 通常,构成RSA模数N的素数p和q应该被独立地产生的。但是,假设一个开发者决定通过选择一个随机数R,并搜索其附近的两个素数作为p和q。那么,我们来证明这种方法得到的RSA的模数 N=pq能被轻易的分解。(而RSA的安全基础就是假定模数不能被轻易分解!)假
- 设给定一个合数N并知道N是两个彼此很接近的素数p和q的乘积,即p和q满足:

$$|p-q|<2N^{\frac{1}{4}} \tag{*}$$

你的任务是分解N。

- $\Diamond A$ 是两个素数的算术平均值,即 $A=rac{p+q}{2}$ 。由于p和q都是奇数,所以A一定是一个整数。
- 为了分解N,首先需要观察,在条件(\*)下 $\sqrt{N}$ 是非常接近A的。具体来讲,有: $A-\sqrt{N}<1$ 。由于A是一个整数,将 $\sqrt{N}$ 凑成最接近的整数便能获取A的值。在代码中,形式大概是 A=ceil(sqrt(N)),其中ceil是上取整函数。
- 更直观地,数字p、q、 $\sqrt{N}$ 和A有如下关系:



- 由于A是p和q的中点,所以存在一个x使得p=A-x以及q=A+x。
- 又因为 $N = pq = (A x)(A + x) = A^2 x^2$ , 因此 $x = \sqrt{A^2 N}$ .
- 现在,根据x和A,你可以找到N的p和q,即分解出了N!
- 在接下来的任务中,需要使用上述的方法来分解给定的模数。本实验需要使用一个支持多精度算数平方根运算的环境。在Python中,可以使用 $gmpy2^1$ 模块;在C++中,可以使用 $GMP^2$ 。

#### 任务一

• 模数N是两个素数p和q的乘积,满足 $|p-q|<2N^{\frac{1}{4}}$ 。(模数N请见附件task.txt)

#### 任务二

- 模数N是两个素数p和q的乘积,满足 $|p-q|<2^{11}N^{\frac{1}{4}}$ 。(模数N请见附件task.txt)
- 提示:  $\triangle A \sqrt{N} < 2^{20}$  的情况下,尝试从 $\sqrt{N}$ 向上搜索A,直到成功分解N。

## 实验原理分析

• 任务一:

$$|p-q| < 2N^{1/4} => (p-q)^2 < 4\sqrt{N}$$
 $=> (p+q)^2 - 4pq < 4\sqrt{N}$ 
 $=> (rac{(p+q)}{2})^2 - pq < \sqrt{N}$ 
因为 $A = rac{(p+q)}{2}N = pq$ 带入上式
 $=> A^2 - N < \sqrt{N} => A - \sqrt{N} < rac{\sqrt{N}}{A + \sqrt{N}}$ 
又因为 $A > \sqrt{N}$ 
因此 $A - \sqrt{N} < rac{1}{2}$ 
可推出 $0 < A - \sqrt{N} < 1$ 

• 任务二:

$$egin{aligned} |p-q| < 2^{11}\sqrt{N} => (p-q)^2 < 2^{22}\sqrt{N} \ &=> (p+q)^2 - 4pq < 2^{22}\sqrt{N} \ &=> (rac{(p+q)}{2})^2 - pq < 2^{20}\sqrt{N} \ &\equiv orall A = rac{(p+q)}{2}N = pq$$
帶 入上式 
$$=> A^2 - N < \sqrt{N} => A - \sqrt{N} < 2^{20} rac{\sqrt{N}}{A + \sqrt{N}} < 2^{19} \ &\equiv \mathbb{E} \sqrt{N} < A < \sqrt{N} + 2^{19} \end{aligned}$$

# Java代码实现 (源码)

```
public class lab4 {
   public static void main(String[] args) {
      task_1();
      System.out.println();
      task_2();
   }

   private static void task_2() {
      String

s="64845584280807166966282426534677227872634372070697626306043907037879730861808
11164627140152760614175691955873218402545206554249067198924288448418393532819729
88531310511738648965962582821502504990264452100885281673303711142296421027840289
307657458645233683357077834689715838646088239640236866252211790085787877";
      BigInteger num=new BigInteger(s);
      BigInteger rem=new BigInteger("1");
```

```
BigInteger a= Sqrt(num).add(rem);
       //num.sqrt();
       while(true){
            BigInteger ans=a.multiply(a);
            BigInteger ans2=ans.subtract(num);
            BigInteger x=Sqrt(ans2);
            BigInteger p=a.subtract(x);
            BigInteger q=a.add(x);
            if(p.multiply(q).equals(num)){
                System.out.println("task2 Success!!");
                System.out.println("task2-p:"+p);
                System.out.println("task2-q:"+q);
                break;
            }else {
                BigInteger temp=new BigInteger("72077");
                a=a.add(temp);
       }
   }
   private static void task_1() {
       String
s = "17976931348623159077293051907890247336179769789423065727343008115773267580550
56206869853794492129829595855013875371640157101398586478337786069255834975410851
96591615128057575940752635007475935288710823649949940771895617054361149474865046
711015101563940680527540071584560878577663743040086340742855278549092581";
       BigInteger num =new BigInteger(s);
       BigInteger rem=new BigInteger("1");
       //开方
       BigInteger a = Sqrt(num).add(rem);
       //a*a
       BigInteger ans=a.multiply(a);
       //a*a-N
       BigInteger ans2=ans.subtract(num);
       //对a*a-N开方获取x
       BigInteger x=Sqrt(ans2);
       //p=a-x
       BigInteger p=a.subtract(x);
       //q=a+x
       BigInteger q=a.add(x);
       if(p.multiply(q).equals(num)){
            System.out.println("task1 Success!!!!");
            System.out.println("task1-p:"+p);
            System.out.println("task1-q:"+q);
       }else{
            System.out.println("Failed!");
       }
   }
   public static BigInteger Sqrt(BigInteger x){
       BigInteger 1 = new BigInteger("0"), r = new BigInteger(x.toString());
       BigInteger ans = new BigInteger("-1");
       while(l.compareTo(r) \leftarrow 0){
            BigInteger mid = r.add(1).divide(new BigInteger("2"));
            if(mid.multiply(mid).compareTo(x) <= 0){
                ans = mid;
                1 = mid.add(new BigInteger("1"));
```

```
}else{
    r = mid.subtract(new BigInteger("1"));
}

return ans;
}
```

# 运行结果

```
**D:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...

task1 Success!!!!

task1-p:13407897929942597099574024998205846127479365820592393377723561443721764030073662768891111614362326998675040546094339320838419523375986027530

task1-q:13407807929942597099574024998205846127479365820592393377723561443721764030073778560988548930557750569660049234002192590823085163940025485114

**task2 Success!!

task2-p:25464796146996183438008816563973942229341454268524157846328581927885777969985222835143851073249573454107384461557193173304497244814071505790

task2-q:25464796146996183438008816563973942229341454268524157846328581927885777970106398054491246526970814167632563509541784734741871379856682354747

Process finished with exit code 0
```