

哈尔滨工业大学（深圳）

《密码学基础》实验报告

RSA 密码算法实验

学 院: 计算机科学与技术

姓 名: 房煊梓

学 号: 210010101

专 业: 智能强基—计算机

日 期: 2023-10-15

一、实验步骤

请说明你的字符分组方式，以及关键的算法例如扩展欧几里德，素数检测，快速幂等函数的实现。

1. 字符分组方式

(1) 对于明文，将字符的 ascii 码值乘以 10 的值加上 ascii 码取余 10 的值作为相应的 4 位十进制数，即直接将字符的 ASCII 码两个字符组成一个四位十进制数处理。加密时，明文 4 个十进制位一组明文加密，也就是一个 4 位数。

(2) 对于密文，设置明文加密后的密文的分组长度为最大可能的长度 $\log_2 n + 1$ 。

2. 求两个数的最大公因子

利用辗转相除法，求两个数 a, b 的最大公因子并输出。

(1) 首先判断 a 和 b 的大小，当 $a < b$ 时，需要交换 a 和 b 的值，确保 $a \geq b$ 。

(2) 接下来再判断 $a \% b$ ，若不为 0，则递归求 b 和 $a \% b$ 的最大公因子。否则，返回 b。

```
/**gcd, 求两个数的最大公因子*/
2 usages
public static BigInteger gcd(BigInteger a, BigInteger b){
    if (a.compareTo(b)<0) {
        //当a<b时，交换一下
        BigInteger k = a;
        a = b;
        b = k;
    }
    if (a.mod(b).equals(number_0)==false) {
        return gcd(b, a.mod(b));
    } else {
        return b;
    }
}
```

3. 扩展欧几里得算法

利用递归实现扩展欧几里得算法。

- (1) 首先判断 b, 若 b 为 0, 则 $ax+by=a$, 此时 a 为最大公因子, x 为 1, y 为 0。此处为递归出口, 返回 x。
- (2) 如果不是以上情况, 则递归调用扩展欧几里得算法, 但是传入的参数变为 b 和 $a \% b$ 。然后将原本的 y 值赋给 x, 将原本的 x 值减去 $a/b*y$ 的值赋给 y, 再返回 x。
- (3) 需要注意, 当最后算出的 d 小于 0 时, 要加上 $\phi(n)$ 。

```

/**扩展欧几里得算法*/
4 usages
public static BigInteger x,y;
2 usages
public static BigInteger extendedEuclid(BigInteger a, BigInteger b){
    if(b.equals(number_0)){
        x=number_1;
        y=number_0;
    }else{
        //递归
        BigInteger d=extendedEuclid(b,a.mod(b));
        BigInteger t=x;
        x=y;
        y=t.subtract(a.divide(b).multiply(y));
    }
    return x;
}

```

4. 素数检测算法

Miller-Rabin 算法判断输入的 n 是否为素数。

- (1) 找出整数 k, q, 其中 $k > 0$, q 是奇数, 使得 $n-1 = 2^k * q$ 。首先设 k 为 0, q 为 n-1, 再利用 while 循环, 当 q 为偶数时, k 加 1, q 除以 2, 直到 q 为奇数。
- (2) 随机选取整数 a, $1 < a < n-1$
- (3) if $a^q \bmod n == 1$ 或者 $n-1$, 返回“很可能为素数”, 这里定为 true
- (4) For j=1 to k-1 do {If $a^{(2^j * q)} \bmod n == n-1$ 返回“很可能为素数”}
- (5) 如果不是以上情况, 返回“合数”, 定为 false

```

/**Miller_Rabin算法判断n是否是素数*/
3 usages
public static boolean millerRabin(BigInteger n){
    //找出整数k,q,其中k>0,q是奇数,使得n-1==2^k*q
    BigInteger k=new BigInteger( val: "0");
    BigInteger q=n.subtract(number_1);
    while((q.mod(number_2)).equals(number_0)){
        k=k.add(number_1);
        q=q.divide(number_2);
    }
    //System.out.println("k:"+k+" "+"q:"+q);

    //随机选取整数a,1<a<n-1 所以a∈[2,n-2]
    BigInteger minLimit = number_2;
    BigInteger maxLimit = n.subtract(number_2);
    maxLimit = maxLimit.multiply(minLimit);
    BigInteger bigInteger = maxLimit.subtract(minLimit);
    Random randNum = new Random();
    int len = maxLimit.bitLength();
    BigInteger a = new BigInteger(len, randNum);
    if (a.compareTo(minLimit) < 0){
        a= a.add(minLimit);
    }
    if (a.compareTo(bigInteger) >= 0){
        a = a.mod(bigInteger).add(minLimit);
    }
    //System.out.println("a:"+a);

    //if a^q mod n ==1或者n-1, 返回"很可能为素数", 这里定为true
    if(mod(a,q,n).equals(number_1)||mod(a,q,n).equals(n.subtract(number_1))){
        //System.out.println("它很可能是素数");
        return true;
    }

    //for循环
    for(BigInteger j=number_0;j.compareTo(k)<0;j=j.add(number_1)){
        int l = j.intValue();
        //System.out.println("j:"+j);
        if(mod(a,(number_2.pow(l)).multiply(q),n).equals(n.subtract(number_1))){
            //System.out.println("它很可能是素数");
            return true;
        }
    }

    //合数, 定为false
    return false;
}

```

5. 快速幂

按照实验指导书的提示，计算 $a^b \bmod p$ 的结果。在 `while(true)` 循环里：

- (1) 判断 `b`，若 `b` 为 0，则返回结果 `res`
- (2) 否则进入 `while` 循环，当 `b` 是一个大于 0 的偶数时，重复：把 $(a^2) \bmod p$ 的值赋给 `a`，把 `b/2` 的值赋给 `b`。
- (3) 退出以上 `while` 循环后，`b` 值减 1，把 $(a * res) \bmod p$ 的值赋给 `res`。

```
/**快速幂运算：计算 $a^b \bmod p$ 的结果*/
5 usages
public static BigInteger mod(BigInteger num1, BigInteger num2, BigInteger num3){
    BigInteger a=num1;
    BigInteger b=num2;
    BigInteger p=num3;
    BigInteger res = new BigInteger("1");
    while(true){
        if(b.equals(number_0)){
            return res;
        }
        while((b.compareTo(number_0)>0)&&(b.mod(number_2)).equals(number_0)){
            a=(a.multiply(a)).mod(p);
            b=b.divide(number_2);
        }
        b=b.subtract(number_1);
        res=(a.multiply(res)).mod(p);
    }
}
```

```
开始解密
根据密文和私钥(d,n)进行解密
解密结束，解密用时：1183ms
得到明文：
2092 A.M. TURING AWARD. RSA, an acronym for Rivest, Shamir and Adleman, uses algorithmic number theory to provide an efficient realization of a
public-key cryptosystem, a concept first envisioned theoretically by Whitfield Diffie, Martin Hellman and Ralph Merkle. RSA is now the most widely
used encryption method, with applications throughout the Internet to secure on-line transactions. It has also inspired breakthrough work in both
theoretical computer science and mathematics.

Process finished with exit code 0
```

三、总结

1、实验过程中遇到的问题有哪些？你是怎么解决的？

在实验过程中主要遇到以下问题：

(1) 使用 BigInteger 类型不熟练的问题

解决方法：上网查找相关的资料辅助代码的编写。

(2) 遇到在生成的大数不是素数时的处理问题

解决方法：利用 do-while 循环，先生成一次大数，再循环判断是否为素数，若不是，则重新生成。

(3) 加密时最后一组长度不足 4 位，进行填充，遇到解密时的处理问题

解决方法：设置一个 flag 来标记是否进行了填充，初始值为 0。若进行填充，则在加密时将其置为 1，解密时根据 flag 的值判断是否需要删除填充。

2、关于本实验的意见或建议。

关于本次实验，建议给出加密解密的流程的模板，对需要实现的函数给出更具体的要求。