

实验三——公钥密码算法 RSA

1611532 刘一静 信息安全

实验要求： 通过实际编程了解公钥密码算法 RSA 的加密和解密过程，加深对公钥密码算法的了解和使用。

目录

一、RSA 简介.....	1
二、密钥的生成	2
三、加解密原理	4
四、关键代码	4

一、RSA 简介

RSA 是一种非对称加密，也就是需要一对密钥，公钥用于加密，私钥用于解密。

RSA 算法涉及五个关键参数：

公钥： e, N ;

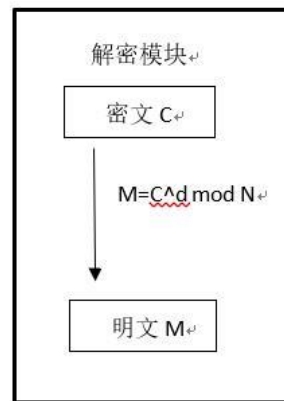
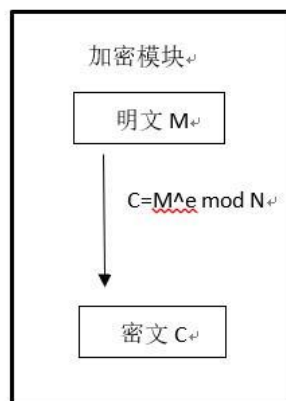
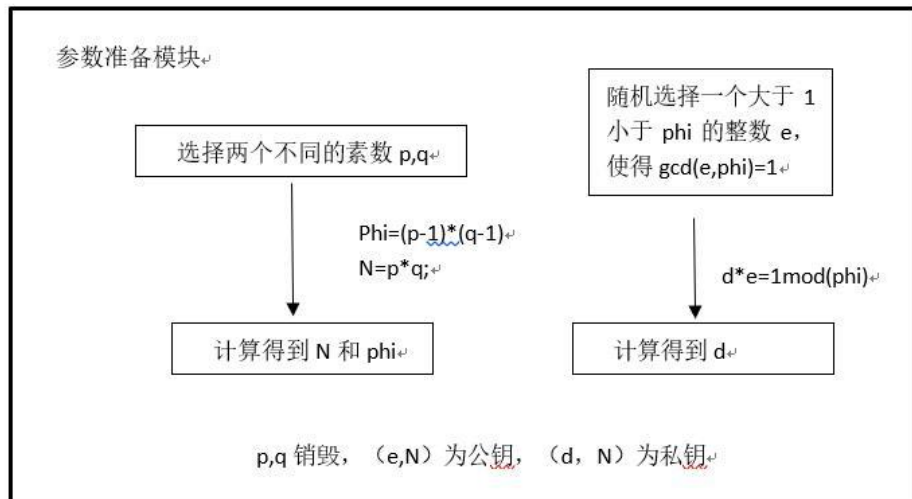
私钥： d, N ;

明文： m ;

密文： c ;

其中 N 是两个大素数 p, q 的积， e, d 满足 $e \cdot d \bmod ((p-1) \cdot (q-1)) = 1$ 。

实现 RSA 算法的完整流程图如下：

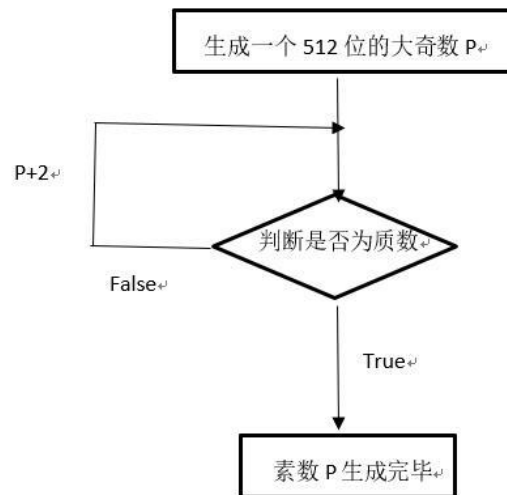


二、密钥的生成

RSA 算法密钥的生成是很麻烦的，因为生成大素数是一个不太容易的事情，同样，在破解 RSA 时，如果攻击者获得了 n 、 e 和密文 c ，为了破解密文必须计算出私钥 d ，为此需要先分解 n 。当 n 的长度为 512 比特时，在目前还是安全的，但从因式分解技术的发展来看，512 比特并不能保证长期的安全性。为了达到更高的安全性，要求在一般的商业应用中使用 1024 比特的长度，在更高级别的使用场合，要求使用 2048 比特长度。

本次实验的第一个关键问题就是如何快速生成 512 比特的素数。

流程图：



1、随机数的生成

除了 2 之外的素数都是都是奇数，因此首先生成一个奇数，然后判断它是否为一个素数，若不是，则将其加 2，用该相邻的奇数继续判断，只到通过素性检验，即可认定生成了一个大素数。

2、Miller—Rabin 素性检验

在进行素性检验时，一般采用 Miller-Rabin 素性检验算法。若该算法返回值为 false，则说明输入的 n 一定是合数；若返回值为 true，也不能肯定 n 一定是一个质数，要多检验几次，一般来说检测 5 次，若 5 次返回值均为 true，则可认为输入的 n 为一个质数。

Miller-Rabin 算法的理论基础：如果 n 是一个奇素数，将 $n-1$ 表示成 $2^s \cdot r$ 的形式(r 是奇数)， a 是和 n 互素的任何整数，那么 $a^r \equiv 1 \pmod{n}$ 或者对某个 $j(0 \leq j \leq s-1, j \in \mathbb{Z})$ 等式 $a^{2^j \cdot r} \equiv -1 \pmod{n}$ 成立。这个理论是通过一个事实经由 Fermat 定理推导而来： n 是一个奇素数，则方程 $x^2 \equiv 1 \pmod{n}$ 只有 ± 1 两个解。

该算法的伪代码如下：

```
1  Miller-Rabin(n,t)
2  输入：一个大于3的奇整数n和一个大于等于1的安全参数t
3  输出：返回n是否是素数，bool类型返回值
4  输入的n-1;
5  二进制形式表示为:  $b_k, b_{k-1}, \dots, b_0$ ;
6  i from 0 to t
7      随机生成一个随机数a, 且  $1 < a < n-1$ 
8      d ← 1
9      j from k to 0
10         x ← d;
11         d = (d*d) % n;
12         如果 d == 1 并且 x != 1 并且 x != m: 返回false;
13         如果 b_j 等于1: d = (a*d) % n;
14     如果 d 不等于1: 返回false
15  返回true;
```

三、加解密原理

1、加密算法

对于明文 m ，由 $c = m^e \bmod n$ ，得到密文 c 。

2、解密算法

对于密文 c ，由 $m = c^d \bmod n$ ，得到明文 m 。

四、关键代码

1、大整数类的定义与实现

本次实验中需要自己定义 BigInt 大整数类，来支持 512 比特数据的加减乘数等基本运算。

具体代码不在实验报告中赘述，具体定义见源码中 BigInt.h 文件。

2、大素数的产生

这模块在源码中的 RSA2 类中进行定义，下面为四个核心的函数：

```
//产生一个素数
+BigInt RSA2::createPrime(unsigned int n, unsigned int k) [ {... } ]

//产生奇数
+BigInt RSA2::random_odd(unsigned int n) [ {... } ]

//判断是否为素数
+bool RSA2::isPrime(const BigInt & n, unsigned int k) [ {... } ]

//素性检验中需要产生一个大于1小于n-1的随机数
+BigInt RSA2::random_a(const BigInt & n) [ {... } ]
```

3、快速模、乘法逆元

这模块在源码中的 match 类中进行定义，下面为四个主要的函数：

```
BigInt gcd(BigInt a, BigInt b); //欧几里德求GCD
BigInt mod_fast(BigInt a, BigInt b, BigInt p); //快速幂模
BigInt extgcd(BigInt a, BigInt b, BigInt &x, BigInt &y); //扩展欧几里德实现乘法逆
BigInt inverse(BigInt a, BigInt m); //求a模m的逆元
```

4、加解密的实现

该模块的实现比较简单，调用上述基本函数即可完成加解密。

加解密函数定义在 RSA2 类中，代码如下：

```
BigInt RSA2::encrypt(const BigInt & m)
{
    //c=m^e (modn)
    BigInt c;
    match mat;
    c = mat.mod_fast(m, e, N);
    return c;
}

BigInt RSA2::decode(const BigInt & c)
{
    //m=c^d(modn)
    BigInt m;
    match mat;
    m = mat.mod_fast(c, d, N);
    return m;
}
```

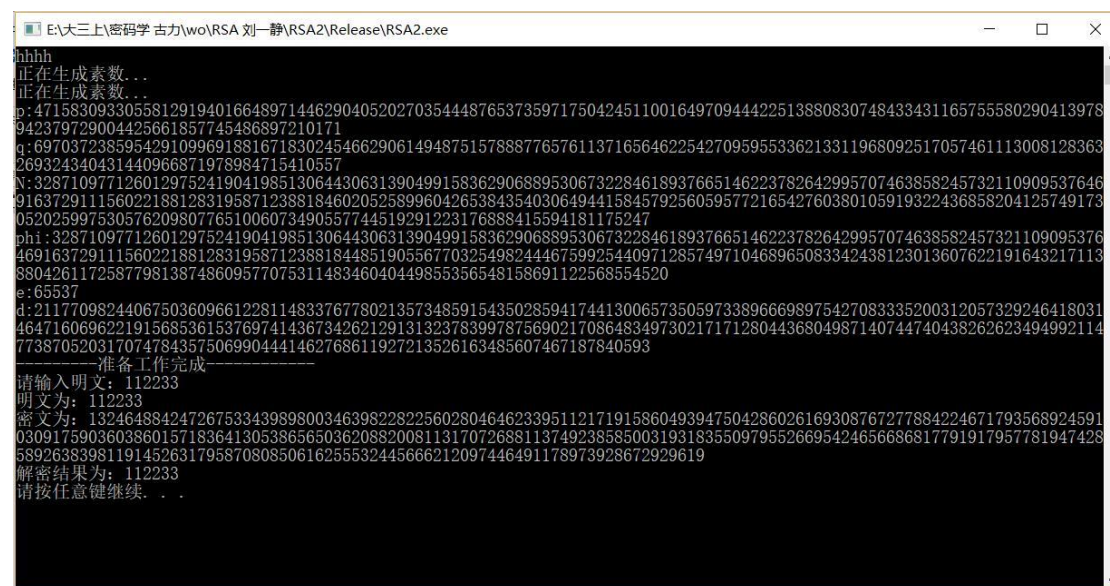
五、最终效果

所有函数都正确编写完成后进行测试：生成一个 512 比特素数的时间存在很大的随机性，平均每 15 秒左右可以产生一个素数。算法效率还可以。

生成的素数经在线检测确定为素数，证明了算法的正确性。

(<https://www.alpertron.com.ar/ECM.HTM>)

下面是运行结果：



```
E:\大三上\密码学 古力\wo\RSA 刘一静\RSA2\Release\RSA2.exe
hhhh
正在生成素数...
正在生成素数...
p: 4715830933055812919401664897144629040520270354448765373597175042451100164970944422513880830748433431165755580290413978
942379729004425661857745486897210171
q: 6970372385954291099691881671830245466290614948751578887765761137165646225427095955336213311968092517057461113008128363
269324340431440966871978984715410557
N: 3287109771260129752419041985130644306313904991583629068895306732284618937665146223782642995707463858245732110909537646
916372911156022188128319587123881846020525899604265384354030649441584579256059577216542760380105919322436858204125749173
0520259975305762098077651006073490557744519291223176888415594181175247
phi: 32871097712601297524190419851306443063139049915836290688953067322846189376651462237826429957074638582457321109095376
469163729111560221881283195871238818448519055677032549824446759925440971285749710468965083342438123013607622191643217113
880426117258779813874860957707531148346040449855356548158691122568554520
e: 65537
d: 2117709824406750360966122811483376778021357348591543502859417441300657350597338966698975427083335200312057329246418031
464716069622191568536153769741436734262129131323783997875690217086483497302171712804436804987140744740438262623494992114
7738705203170747843575069904441462768611927213526163485607467187840593
-----准备工作完成-----
请输入明文: 112233
明文为: 112233
密文为: 1324648842472675334398980034639822822560280464623395112171915860493947504286026169308767277884224671793568924591
030917590360386015718364130538656503620882008113170726881137492385850031931835509795526695424656686817791917957781947428
5892638398119145263179587080850616255532445666212097446491178973928672929619
解密结果为: 112233
请按任意键继续. . .
```

解密加密之后的明文可以得到正确的明文，正确实现了 RSA 算法的加密与解密。