南开大学 Nankai University

南　开　大　学

网络空间安全学院

密码学课程报告

第三次实验报告

——公钥密码算法 RSA

学号：1611519

姓名：周子祎

年级：2016 级

专业：信息安全-法学

2018 年 12 月 9 日

密码学第三次实验报告

——公钥密码算法 RSA

一、 实验目的

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。

二、 实验原理

RSA 算法描述如下:

1. 公钥

选择两个不同的大素数 p 和 q，n 是二者的乘积，即 n = pq，使

$$\varphi(n) = (p-1)(q-1)$$

$\varphi(n)$ 为欧拉函数。随机选取正整数 e，使其满足 $(e,\varphi(n)) = 1$，即 e 和 $\varphi(n)$ 互素，则将（n，e）作为公钥。

2. 私钥

求出正数 d，使其满足 $e \times d \equiv 1 \bmod \varphi(n)$，则将（n，d）作为私钥。

3. 加密算法

对于明文 m，由 $c \equiv m^e \bmod n$，得到密文 c。

4. 解密算法

对于密文 c，由 $m \equiv c^d \bmod n$，得到明文 m。

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。

三、 实验要求

1. 编写一个程序，用于生成 512 比特的素数。写出生成素数的原理，包括随机数的生成原理和素性检测的内容，并给出程序框图
2. 利用 2 中程序生成的素数，构建一个 n 的长度为 1024 比特的 RSA 算法，利用该算法实现对明文的加密和解密。要求分别实现加密和解密两个功能，并分别给出程序框图
3. 运行对话框程序 RSATool, 运行这个程序加密一段文字, 了解 RSA 算法原理。

四、 实验内容

（一）大素数生成程序

1. 大素数生成原理

RSA 算法的实现首先需要考虑如何生成两个大素数 p 和 q。为了防止敌手通过穷搜索发现 p、q，这两个素数应是在一个足够大的整数集合中选取的大数。

在本次实验中，大素数的生成通过以下步骤完成：

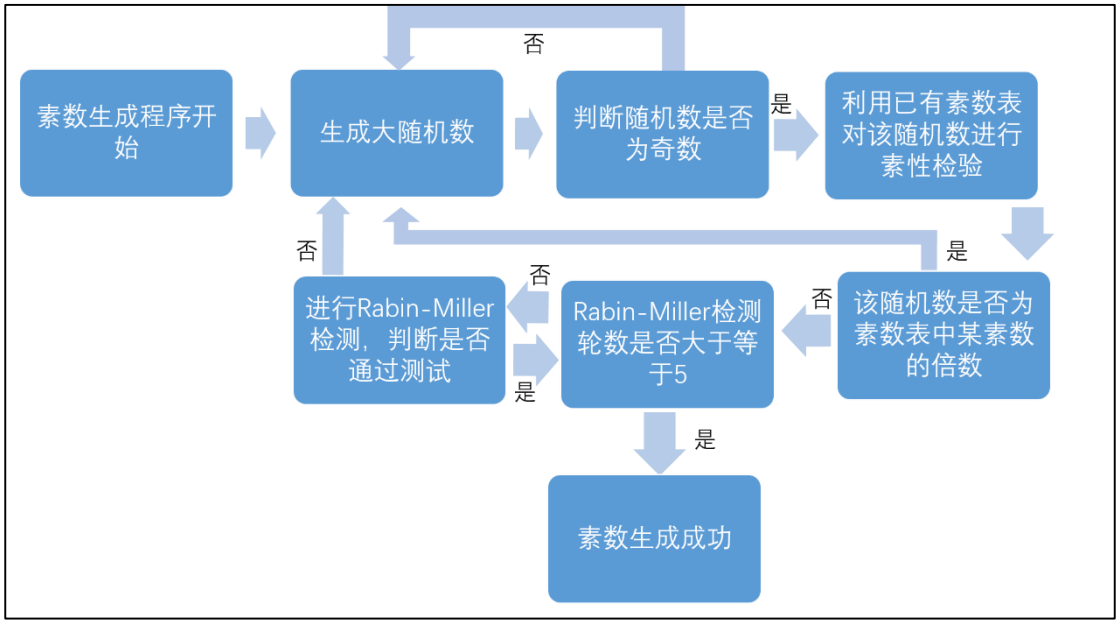
(1) 生成一个随机数

(2) 判断该随机数是否为奇数，如果是奇数，继续进行步骤(3)；如果不是奇数，返回去执行步骤(1)

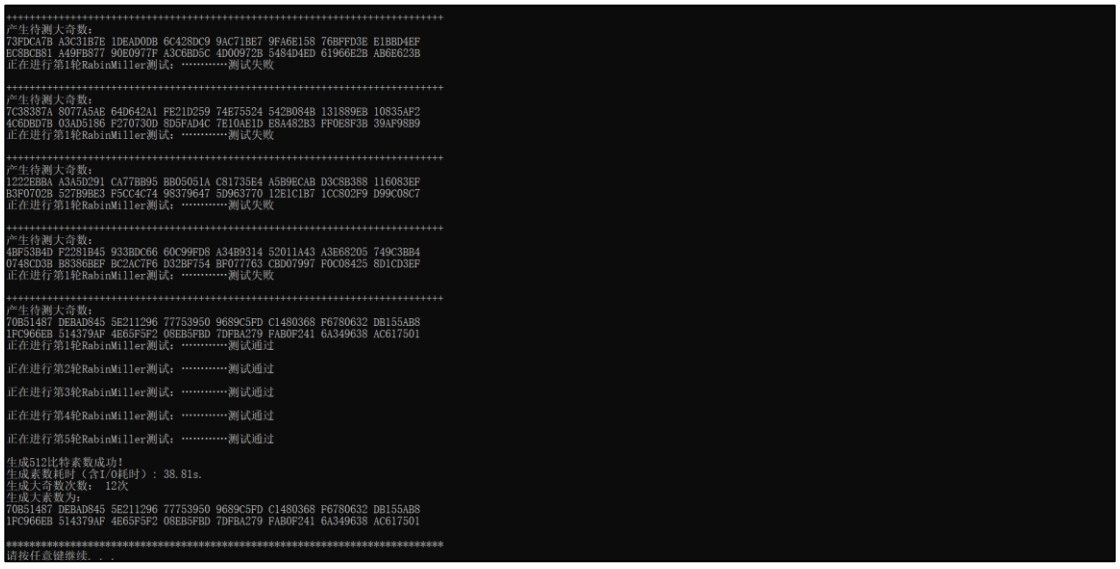
(3) 利用预置的素数表该随机数的素性进行初步检测，如果初步检验通过，继续进行步骤(4)；如果初步检验失败，返回去执行步骤(1)

(4) 利用 Rabin-Miller 概率检测法进行 5 次素性检测，如果检验不通过，返回去执行步骤(1)；如果 5 次检验全部通过，则认为成功生成大素数

2. 大素数生成程序流程图



3. 大素数生成程序执行结果：



4. 大素数生成程序部分代码：

主函数部分代码如下，其余各功能函数的详细代码请参见工程文件 hw3_1

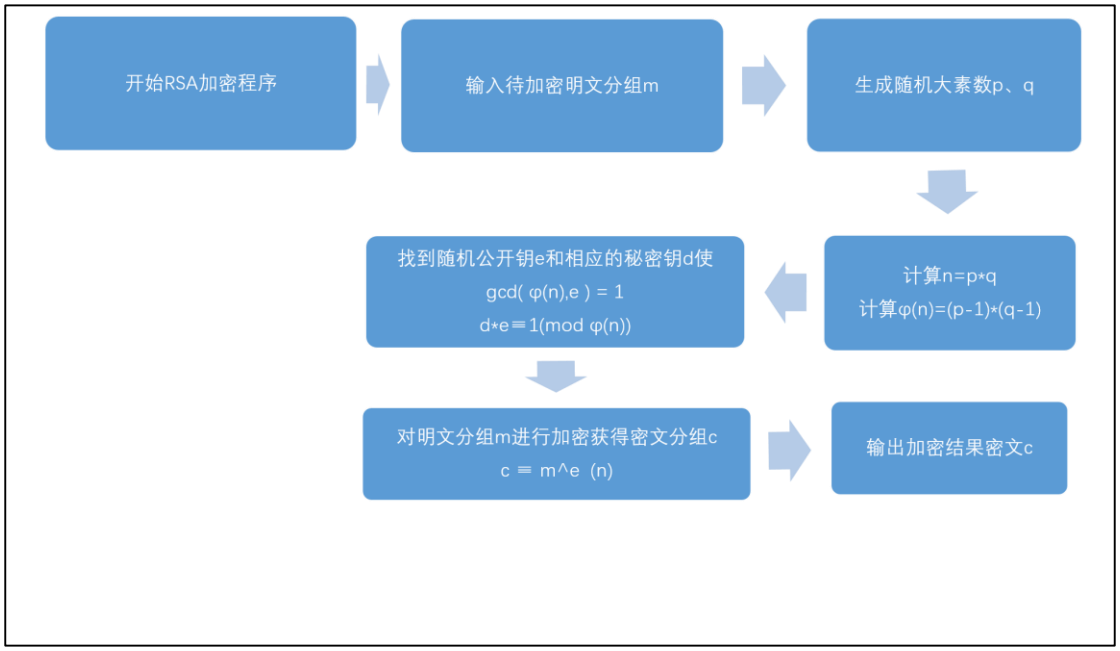
```
#include <iostream>
```

```
#include <fstream>
#include <time.h>
#include <stdlib.h>
#include "BigInt.h"
#include "PrimerTable.h"
#include "PrimerGen.h"
#include<ctime>

using namespace std;

int main()
{
    srand((unsigned)time(NULL));

    cout << "************************************************" << endl;
    cout << "*                    素数生成测试                    *" << endl;
    cout << "************************************************" << endl;
    cout << "开始测试" << endl;
    cout << "------------------------------------------------------------
-----------------" << endl;

    //记录生成了多少次大奇数
    int time = 0;

    //开始计时
    clock_t start = clock();

    //产生大素数
    BigInt p = GeneratePrime(time);

    //结束计时
    clock_t finish = clock();

    cout <<"生成 512 比特素数成功！"<< endl;
    cout << "生成素数耗时（含 I/O 耗时）: " << (double)(finish - start) /
CLOCKS_PER_SEC << "s." << endl;
    cout << "生成大奇数次数:  " << time<< "次" << endl;
    cout << "生成大素数为:  "<< endl;

    //16 进制形式显示
    p.display();

    cout << endl;
    cout <<
"************************************************************************
*****" << endl;

    system("pause");
    return 0;
}
```

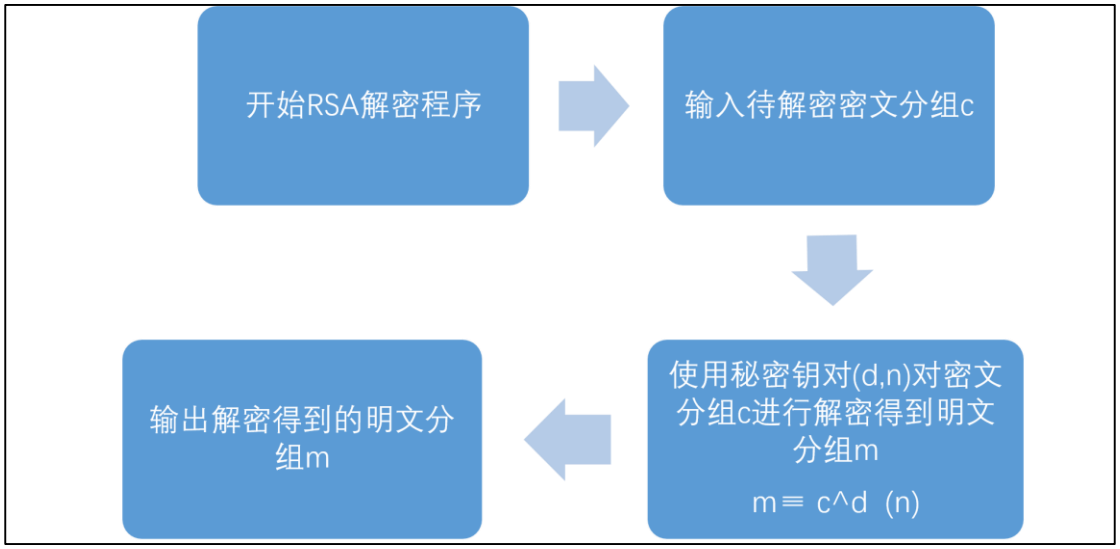
（二）RSA 加解密程序

1. RSA 加解密流程图

（1）RSA 加密过程流程图



（2）RSA 解密过程流程图



2. RSA 程序实现结果

相关源代码请参见工程文件 hw3_2

（1）生成大素数 P

开始生成 p:

```
*                                    RSA测试                                    *
********************************************************************************
开始测试
********************************************************************************
生成大素数 p
--------------------------------------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数:
58C311A0 D66A24B0 AD387EF3 580D6D7E A32131E4 51CE0F85 2E08FB17 42DBAF91
D0EF3CB1 AA1CC51F EE851302 CA3B44AA 8C86DA2D 55F6A712 3A197255 18B558CD

正在进行第1轮RabinMiller测试: ·····测试失败

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数:
B85138AB 3FCC8CBF 596C9D2E A42A7716 822825E5 9A6FF9F1 ECACEB59 C0A43A59
36366DC2 E69CDFF2 806DB70E A4CB2DC1 8546DDB7 A786B0D3 6C2B8FA0 F5285D5F

正在进行第1轮RabinMiller测试: ·····测试失败

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数:
E676846F B27C8777 2DACF76F 77C12D2B 6EF4FA31 8EEFD9B8 68DD1E4F E3C8ED79
BA3FD9D0 37D26F2E 1CD0F1D9 71D53197 FEA4BBAC D6715167 B04671DC 4B95C2BD

正在进行第1轮RabinMiller测试: ·····测试失败

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数:
1E01DADA FEB3EE64 1C9671AB 9BFB6E58 88F0990D 05E2E124 0B8268A1 1AB6DB3B
27BC1876 5EDAA1D3 15C3109F F3E74217 00F5AC07 C9751E7D A3FE59FF B0067F45

正在进行第1轮RabinMiller测试: ·····测试失败

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数:
34393317 6CFE171E 95FAEF5E 82E59E09 28ECD0DC F77C7CC8 679CAE13 E07C40D7
14851C64 0312AC38 A7B8E7BA 674B82DB ABCEFF32 A00CECE2 19ACF850 BB0938FB

正在进行第1轮RabinMiller测试: ·····测试失败

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数:
C48C26F6 B16D228F 3138909F 152AD888 06DFC5A7 15BDA3BC 58003E6C 6CFA8C5A
0A218B9E FC256B6C C02241AE 96914CB5 35C44E5C 9D39BB6F 3ED50752 C5A95A85

正在进行第1轮RabinMiller测试: ·····测试失败
```

生成 p 成功:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数：
E7AE2ADC B07B0755 DF8997EC 38BB0B42 3BAA3018 DA5AD560 BD8D9479 D0F973C9
8CB09031 72FA4536 16FAE051 83195181 91334AD2 137B0954 0F74B5E7 B5C7879D

正在进行第1轮RabinMiller测试：·····测试通过

正在进行第2轮RabinMiller测试：·····测试通过

正在进行第3轮RabinMiller测试：·····测试通过

正在进行第4轮RabinMiller测试：·····测试通过

正在进行第5轮RabinMiller测试：·····测试通过

生成512比特素数p成功！
生成大素数p耗时（含I/O耗时）：142.255s.
生成大奇数次数：  49次
生成大素数p为：
E7AE2ADC B07B0755 DF8997EC 38BB0B42 3BAA3018 DA5AD560 BD8D9479 D0F973C9
8CB09031 72FA4536 16FAE051 83195181 91334AD2 137B0954 0F74B5E7 B5C7879D
```

（2）生成大素数 q

```
**********************************************************************************
生成大素数 q
--------------------------------------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数：
350B32EF 0DB90F2D 0E206361 63A75FB8 138E3845 7C8C44D7 0C759039 70E08914
1D4EF409 5FD01BF0 E8EBBAD1 CD3F7FD5 1CDFB91D 3927AFED E0633223 3EDB2695

正在进行第1轮RabinMiller测试：·····测试失败

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数：
DA2A0A5D 3F743A5F 3D2EA46C 0E1A202E 55F76B17 9E2C36C1 6B67F5DC 92F86937
9B08BABE 0D061C39 DF5DB222 5A570F28 FB0FAD9F 9DBF4186 EEE4FE76 3F1B73F3

正在进行第1轮RabinMiller测试：·····测试失败
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
产生待测大奇数：
D6ACB760 83113911 1CBEFD0A 74983C73 C0FEEBF4 C075DBE0 44878364 041B864B
0A676520 BD4C3A44 ABE0E618 C1F8F874 5FC1BC0E C0B5D2D5 24BAA9CB DD39A403

正在进行第1轮RabinMiller测试：·····测试通过

正在进行第2轮RabinMiller测试：·····测试通过

正在进行第3轮RabinMiller测试：·····测试通过

正在进行第4轮RabinMiller测试：·····测试通过

正在进行第5轮RabinMiller测试：·····测试通过

生成512比特素数q成功！
生成大素数q耗时（含I/O耗时）：15.496s.
生成大奇数次数：  4次
生成大素数q为：
E7AE2ADC B07B0755 DF8997EC 38BB0B42 3BAA3018 DA5AD560 BD8D9479 D0F973C9
8CB09031 72FA4536 16FAE051 83195181 91334AD2 137B0954 0F74B5E7 B5C7879D
```

（3）计算公开钥 n = p * q

```
******************************************************************************
计算公开钥n = p * q
------------------------------------------------------------------------------
公开钥n为：
C247E6CE 205EF5D9 E206E644 CE011342 B46F09EA 497287E8 48185DA8 CA8E7FEE
29884B5D 215F3F7A 9E4E2E9A 3F221E6A 53F455AD EA55BBBB 3FD28747 7F30F405
351111FD 2035ADE4 5E877D44 3327F5BE F57715FF 61E6D77D A35A79C9 CDE3B417
8F6ECA89 07606165 37D4CBDE 8530F552 CF72D9E7 4D4A1F51 F53C7A29 AF2C2AD7
```

（4）计算公开钥 e 和秘密钥 d

```
******************************************************************************
计算公开钥e 和秘密钥d
------------------------------------------------------------------------------
公开钥e为：
47BD1657 841350ED B2C97DF4 248A2A6F 8BD1E1AA 0F98AE4C 650579C4 76D9FC3C
5413A3CE 35546873 F10CD349 75F4375C 43BCFBA2 DA2B2758 4E53DA2B 479A60C3


秘密钥d为：
2BA5F346 E170DDF0 EE2BD9FF 8134FB9E D18762B7 B944A2B2 5D2AEF99 66CCBB3C
E587DECE 7B1411A5 4B6C93B4 1194F5C4 632589B6 F01455FD A157BB0D 21D8751F
7E01CE72 ABE1BAB0 8BF29641 2F74BFBB B7D0A5BF 7F7CD96C 6B99B51C 3CC8050B
6BAAC307 F94EAFDB FC484B3C 77685CC4 861C32B8 BE77ECE2 6815907F D0FE7B33
```

（5）随机生成明文分组 m

（此处处于测试效率考虑未设置用户输入环节，直接随机生成 512bit 明文）

```
**************************************************************************
随机生成明文分组m
--------------------------------------------------------------------------
明文分组m为:
B3445FEA 5599759E 2A9E1648 12C8BF87 D1773575 EDA703F8 2390123F 911F68CA
CF35FCA1 20B59D68 8132E19A 84563601 EAF6A0D2 9A097386 DD5F757E 2FFBA91F
```

（6）用秘密钥 e 对 m 加密,得到密文分组 c

```
**************************************************************************
用秘密钥e对m加密,得到密文分组c
--------------------------------------------------------------------------
密文分组c为:
9E4F7C05 A1C03FDE 5A22DA67 F0BC288D 0CEABF15 D664941F 42698B2E 14AC0337
868A56C6 E3FE2299 401C325F E5CAE23D 3DAF52D3 5AC60DFD 7D69F8BF A295B3E8
8970B17E CA7C7B67 CAE2E3C9 B570BE79 337378C7 2DD56926 20A57777 DD2E5295
DCE3EEE0 1FBDF233 2B777082 2BEAE349 5DD9BBC8 09E02628 9E6DA6E3 66C1A192
```

（7）用公开钥 d 对 c 解密，得到明文分组 m2

```
**************************************************************************
用公开钥d对c解密，得到明文分组m2
--------------------------------------------------------------------------
明文分组m2为:
B3445FEA 5599759E 2A9E1648 12C8BF87 D1773575 EDA703F8 2390123F 911F68CA
CF35FCA1 20B59D68 8132E19A 84563601 EAF6A0D2 9A097386 DD5F757E 2FFBA91F
```

5. RSA 加解密程序部分代码：

主函数部分代码如下，其余各功能函数的详细代码请参见工程文件 hw3_2

```
#include <iostream>
#include <fstream>
#include <time.h>
#include <stdlib.h>
#include "BigInt.h"
#include "PrimerTable.h"
#include "PrimerGen.h"
#include<ctime>

using namespace std;

int main()
{
    srand((unsigned)time(NULL));
    ofstream outfile("test.txt");
```

```
    cout <<
"**************************************************************************
*****" << endl;
    cout << "*                                    RSA 测试
*" << endl;
    cout <<
"**************************************************************************
*****" << endl;
    cout << "开始测试" << endl;

    cout <<
"**************************************************************************
*****" << endl;
    cout << "生成大素数 p" << endl;
    cout << "-----------------------------------------------------------
-----------------" << endl;
    //记录生成了多少次大奇数
    int time = 0;

    //开始计时
    clock_t start = clock();

    //产生大素数
    BigInt p = GeneratePrime(time);

    //结束计时
    clock_t finish = clock();

    cout <<"生成 512 比特素数 p 成功！"<< endl;
    cout << "生成大素数 p 耗时（含 I/O 耗时）： " << (double)(finish - start)
/ CLOCKS_PER_SEC << "s." << endl;
    cout << "生成大奇数次数：  " << time<< "次" << endl;
    cout << "生成大素数 p 为：  "<< endl;

    //16 进制形式显示
    p.display();
    outfile << "大素数 p:" << endl;
    outfile << p;
    cout << endl;

    cout <<
"**************************************************************************
*****" << endl;
    cout << "生成大素数 q" << endl;
    cout << "-----------------------------------------------------------
-----------------" << endl;
    //记录生成了多少次大奇数
    int time2 = 0;

    //开始计时
    clock_t start2 = clock();

    //产生大素数
    BigInt q = GeneratePrime(time2);
```

```
    //结束计时
    clock_t finish2 = clock();

    cout <<"生成 512 比特素数 q 成功！"<< endl;
    cout << "生成大素数 q 耗时（含 I/O 耗时）： " << (double)(finish2 -
start2) / CLOCKS_PER_SEC << "s." << endl;
    cout << "生成大奇数次数： " << time2<< "次" << endl;
    cout << "生成大素数 q 为： "<< endl;

    //16 进制形式显示
    p.display();
    outfile << "大素数 q:" << endl;
    outfile << q;
    cout << endl;

    cout <<
"**************************************************************************
*****" << endl;
    cout << "计算公开钥 n = p * q" << endl;
    cout << "-----------------------------------------------------------
----------------" << endl;
    BigInt n = p*q;

    cout << "公开钥 n 为： "<< endl;
    //16 进制形式显示
    n.display();
    outfile << "公开钥 n 为：" << endl;
    outfile << n;
    cout << endl;

    cout <<
"**************************************************************************
*****" << endl;
    cout << "计算公开钥 e 和秘密钥 d " << endl;
    cout << "-----------------------------------------------------------
----------------" << endl;
    BigInt t = (p - 1)*(q - 1);

    //e 为公开钥
    BigInt e;

    //d 为秘密钥，即 e 模 t 的乘法逆元
    BigInt d;

    //y 用于参与扩展欧几里得运算，存储 t 模 e 的乘法逆元
    BigInt y;

    BigInt temp;

    while (1)
    {
        //产生与 t 互质的 e
        e.Random();
```

```
        while (!(Gcd(e, t) == 1))
        {
            e.Random();
        }

        //用扩展欧几里德算法试图求出 e 模 t 的乘法逆元
        temp = ExtendedGcd(e, t, d, y);

        //e*d 模 t 结果为 1，说明 d 确实是 e 模 t 的乘法逆元
        temp = (e*d) % t;
        if (temp == 1)
            break;

        //否则重新生成 e
    }


    cout << "公开钥 e 为：  "<< endl;
    //16 进制形式显示
    e.display();
    outfile << "公开钥 e:" << endl;
    outfile << e;
    cout << endl;

    cout << "秘密钥 d 为：  "<< endl;
    //16 进制形式显示
    d.display();
    outfile << "秘密钥 d:" << endl;
    outfile << d;
    cout << endl;

    cout <<
"**************************************************************************
*****" << endl;
    cout << "随机生成明文分组 m  " << endl;
    cout << "-----------------------------------------------------------
-----------------" << endl;
    BigInt m;
    m.Random();
    cout << "明文分组 m 为："<< endl;
    //16 进制形式显示
    m.display();
    outfile << "明文分组 m 为：" << endl;
    outfile << m;
    cout << endl;

    cout <<
"**************************************************************************
*****" << endl;
    cout << "用秘密钥 e 对 m 加密,得到密文分组 c  " << endl;
    cout << "-----------------------------------------------------------
-----------------" << endl;
    BigInt c = PowerMode(m, e, n);
    cout << "密文分组 c 为："<< endl;
```

```
    //16 进制形式显示
    c.display();
    outfile << "密文分组 c 为：" << endl;
    outfile << c;
    cout << endl;

    cout <<
"**********************************************************************
*****" << endl;
    cout << "用公开钥 d 对 c 解密，得到明文分组 m2 " << endl;
    cout << "-----------------------------------------------------------
-----------------" << endl;
    BigInt m2 = PowerMode(c, d, n);
    cout << "明文分组 m2 为："<< endl;
    //16 进制形式显示
    m2.display();
    outfile << "明文分组 m2 为：" << endl;
    outfile << m2;
    cout << endl;

    system("pause");
    return 0;
}
```