

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

1. 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。

1. 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

为欧拉函数。随机选取正整数e，使其满足，即e和互

素，则将（n，e）作为公钥。

1. **私钥**

求出正数d，使其满足，则将（n，d）作为私钥。

1. **加密算法**

对于明文m，由，得到密文c。

1. **解密算法**

对于密文c，由，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。

1. 实验要求
2. 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
3. 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
4. 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。
5. 实验内容

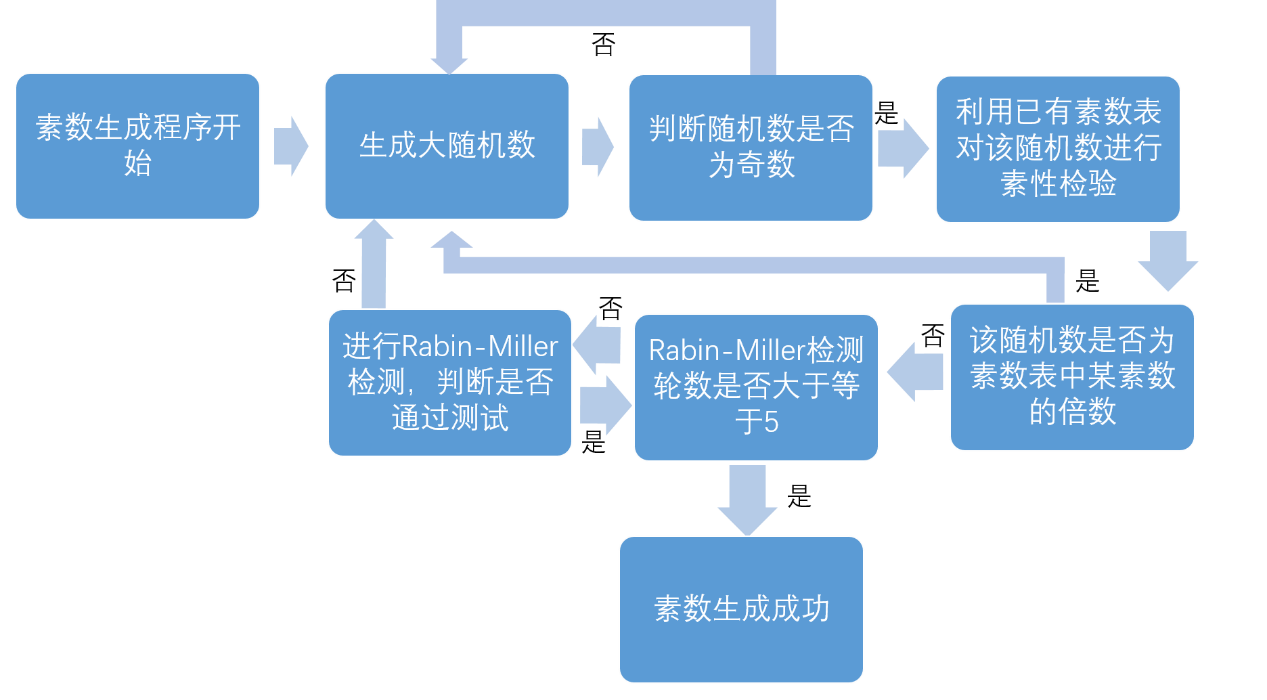
**（一）大素数生成程序**

1. **大素数生成原理**

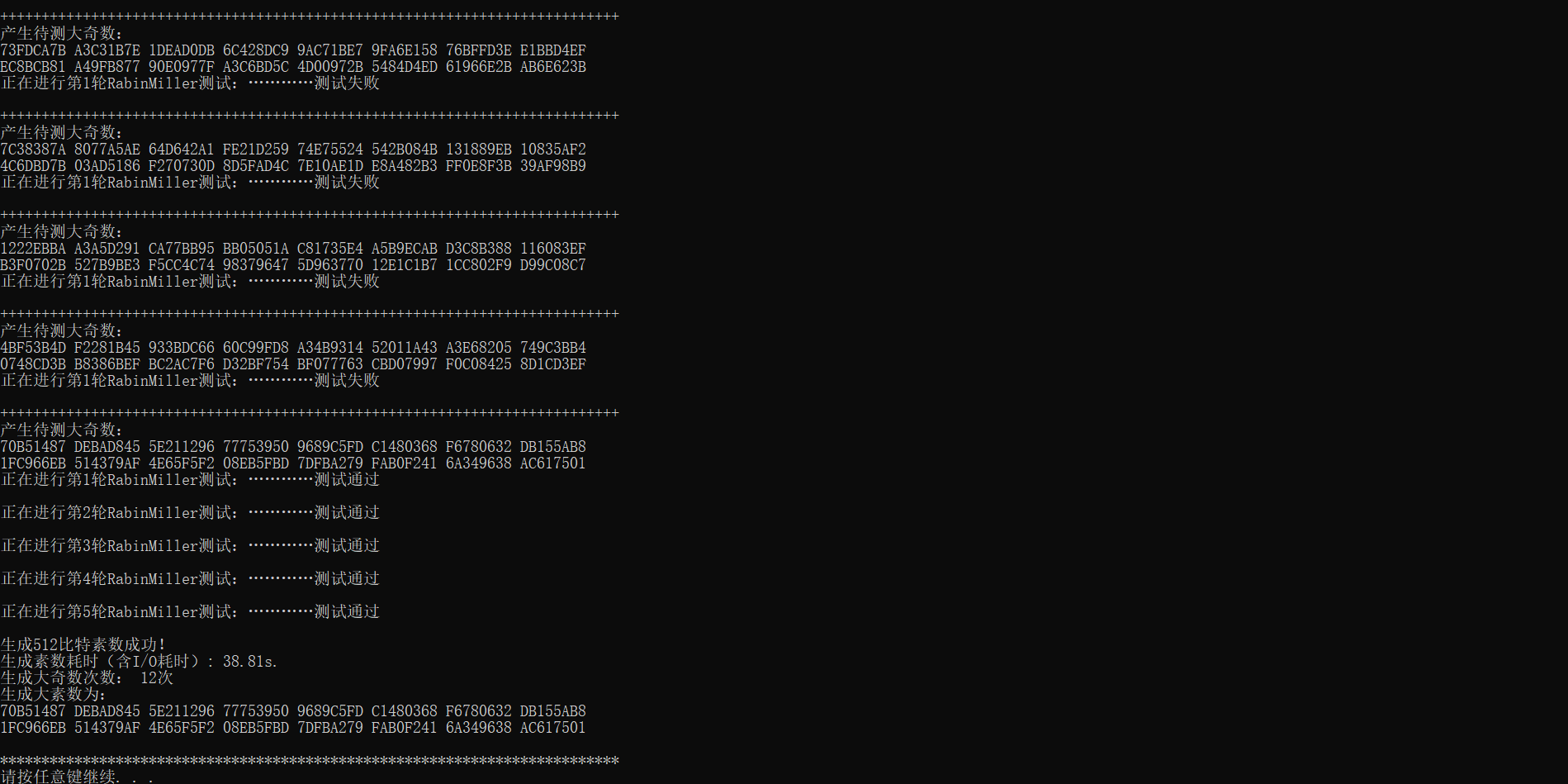
RSA算法的实现首先需要考虑如何生成两个大素数p和q。为了防止敌手通过穷搜索发现p、q，这两个素数应是在一个足够大的整数集合中选取的大数。

**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
5. **大素数生成程序流程图**



1. **大素数生成程序执行结果：**



1. **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");

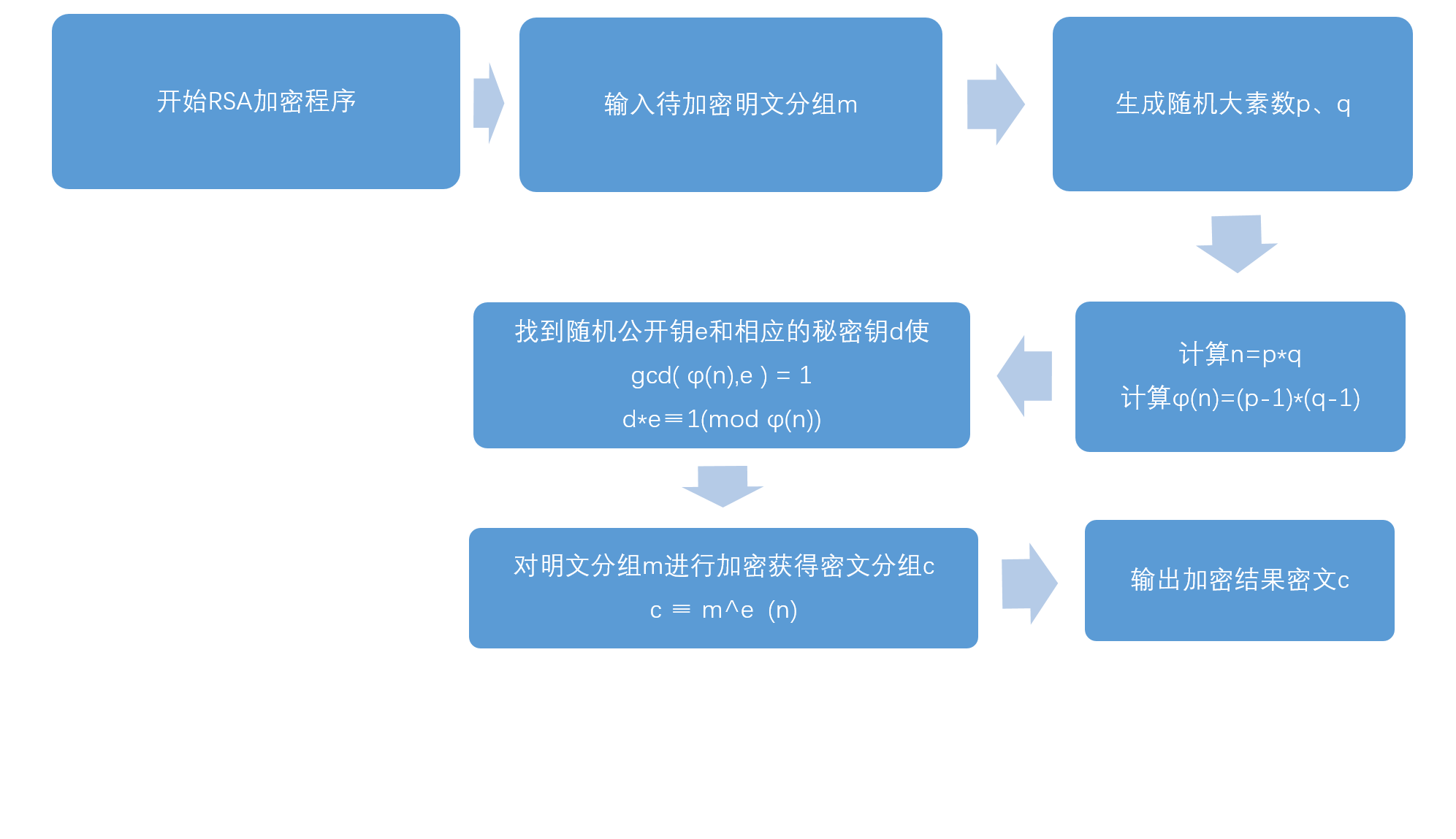
    return 0;

}

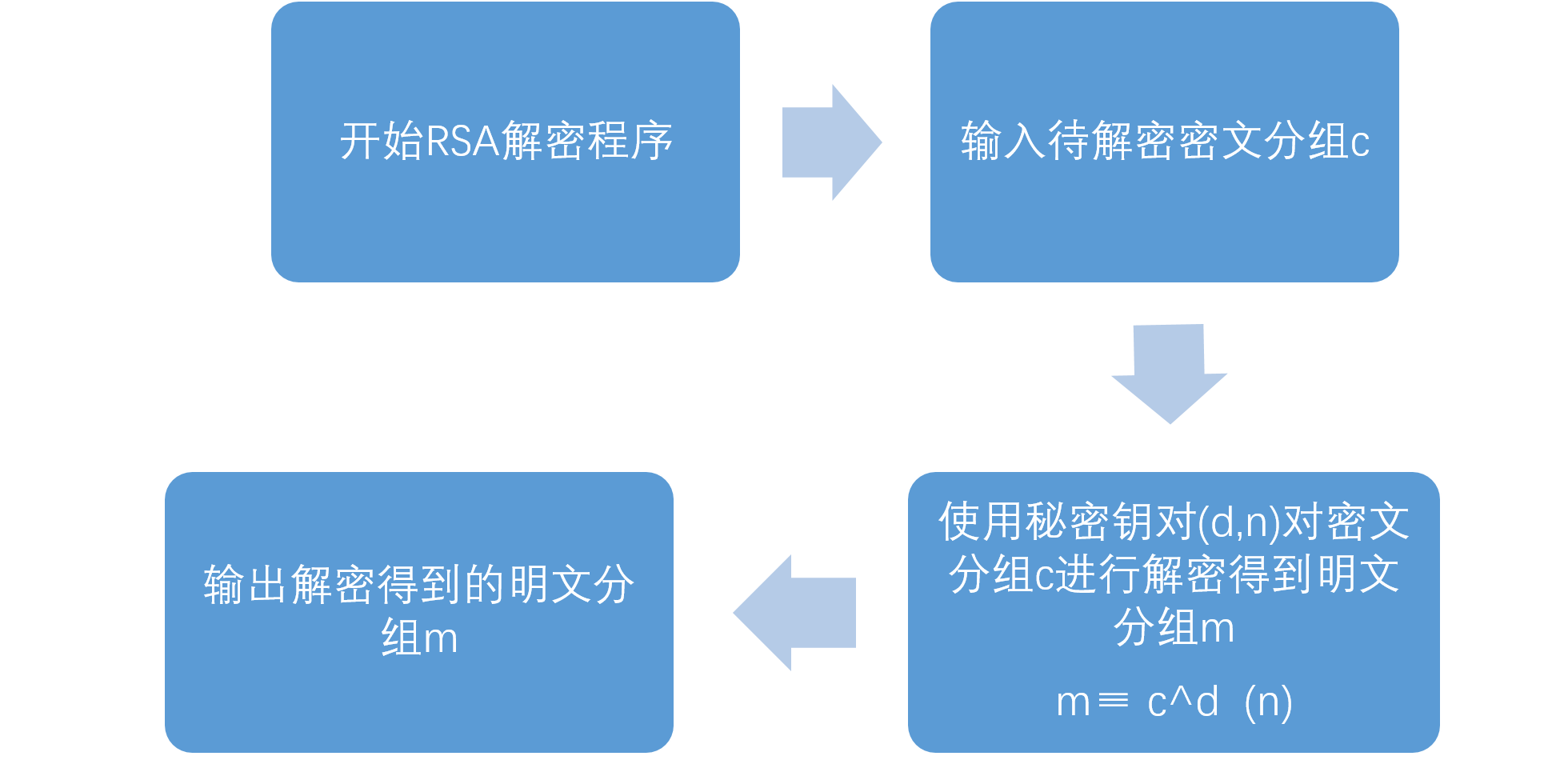
**（二）RSA加解密程序**

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**

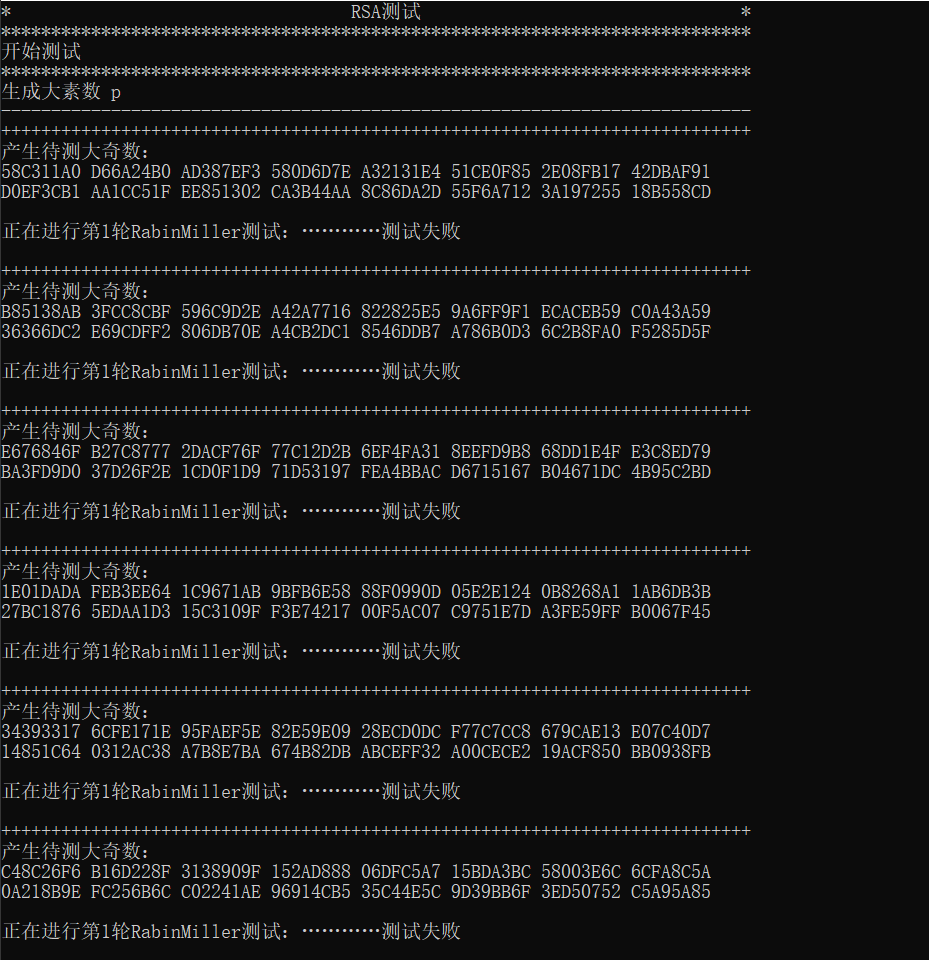
****

1. **RSA程序实现结果**

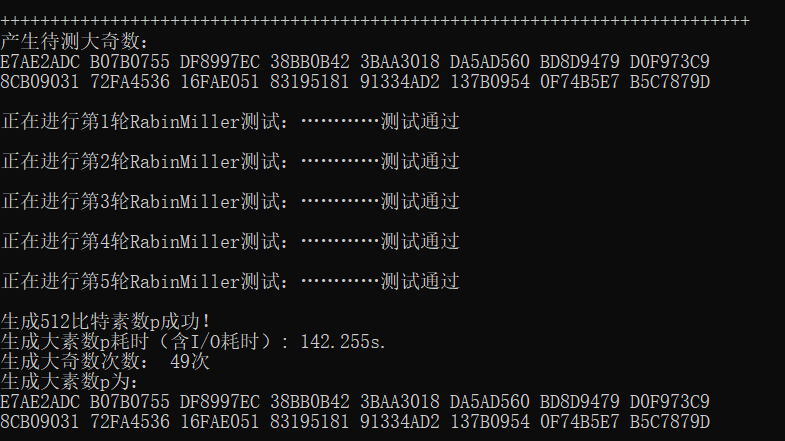
相关源代码请参见工程文件hw3\_2

**（1）生成大素数P**

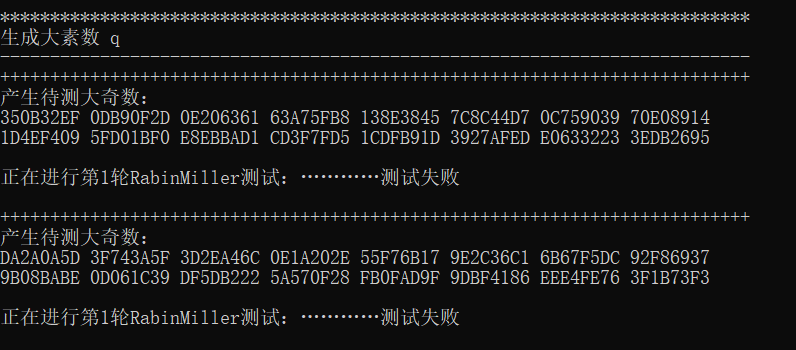
开始生成p：

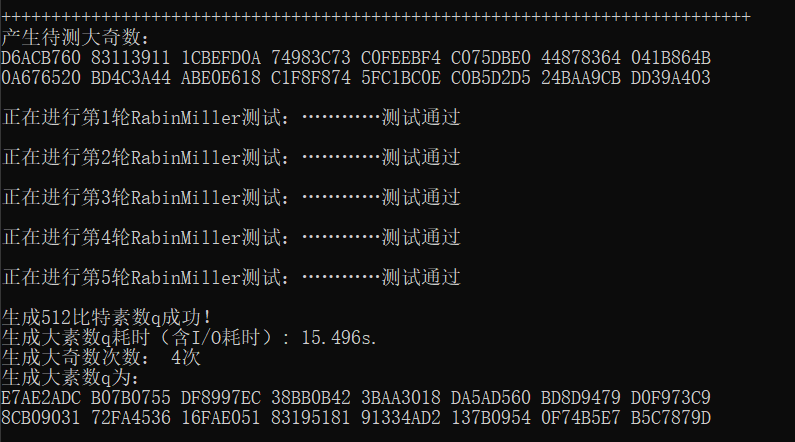


生成p成功：

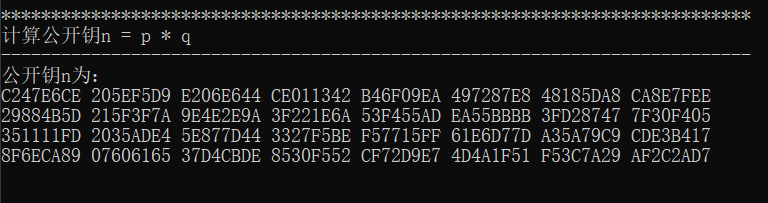


**（2）生成大素数q**

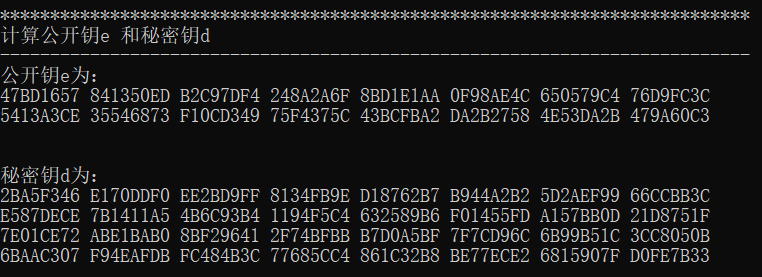




**（3）计算公开钥n = p \* q**

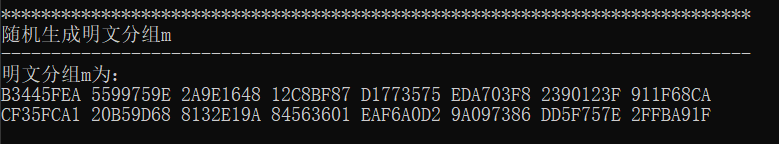


**（4）计算公开钥e 和秘密钥d**

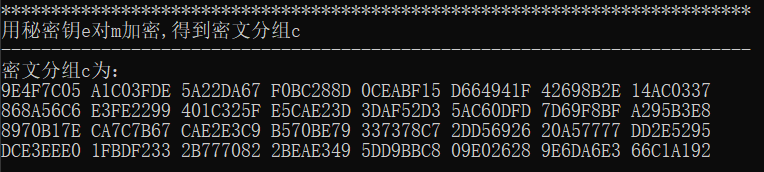


**（5）随机生成明文分组m**

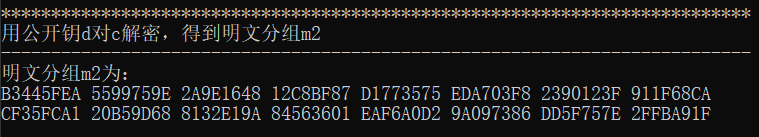
（此处处于测试效率考虑未设置用户输入环节，直接随机生成512bit明文）



**（6）用秘密钥e对m加密,得到密文分组c**



**（7）用公开钥d对c解密，得到明文分组m2**



1. **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;

}