北京师范大学珠海分校

本科生毕业论文

论文题目：RSA 加密算法的实现与应用

学　　院　信息技术学院

专　　业　软件工程

学　　号　0901030140

学 生 姓 名　叶泽鸿

指导教师姓名　陈红顺

指导教师职称　讲师

指导教师单位　信息技术学院

2013 年 03 月 21 日

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

北京师范大学珠海分校学位论文写作声明和使用授权说明

优秀毕业论文

学位论文写作声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名： 日期： 年 月 日

（范例）

学位论文使用授权说明

样本

本人完全了解北京师范大学珠海分校关于收集、保存、使用学位论文的规定，即：按照学校要求提交学位论文的印刷本和电子版本；学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务；学校可以采用影印、缩印、数字化或其它复制手段保存论文；在不以赢利为目的的前提下，学校可以将学位论文编入有关数据库，提供网上服务。（保密论文在解密后遵守此规定）

仅供参考

论文作者签名： 导师签名：

日期： 年 月 日

严禁抄袭

RSA 加密算法的实现与应用

优秀毕业论文

摘 要

本文介绍 RSA 加密算法，并分析 RSA 加密算法的应用现状。论证 RSA 加密算法用于文件加密的可行性与意义。设计一套 RSA 算法用于文件加密的完整解决方案，实现具体的编码。研究 RSA 算法，使用 C++实现 RSA 算法以及与其相关的大数类。素数测试算法选用米勒-拉宾素数测试算法。经过加密或解密的文件均为文本文件。给出关键类类图、整个应用程序的结构描述文档、关键模块流程图、较详细的接口文档、所有源代码及代码注释。对应用程序进行测试，并对测试结果进行分析研究，进而优化应用程序。对关键的算法进行尽可能的优化。最终得到一个在 windows 系统上运行的可指定密钥对任意文本文件进行 RSA 加密并可解密的完整应用程序。

关键词：RSA；加密；算法；文件

样本

仅供参考

严禁抄袭

Implementation and Application of the RSA encryption algorithm

优秀毕业论文（设计）

ABSTRACT

This article describes the RSA encryption algorithm, and analyzes the application status of the RSA encryption algorithm. Argues RSA encryption algorithm for file encryption feasibility and significance. RSA algorithm design a complete solution for file encryption, to achieve specific coding. Research RSA algorithm, using the C + + implementation of the RSA algorithm and the associated class of large numbers. Prime testing the algorithm selection Miller - Rabin prime testing algorithm. Encrypted or decrypted files are text files. Given class diagram of key classes, the entire application schema document, the key module flow chart, detailed interface documentation, all source code and code comments. Application test, and the test results were analyzed to optimize the application. Of key algorithm optimized as much as possible. Eventually get a run in the windows system can be specified key to the full application of the RSA encryption and decryption of any text file.

Key words: RSA; encryption; algorithm; File

样本

仅供参考

严禁抄袭

目　录

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

目录

1. 绪论

信息加密的基本过程就是对原来为明文按某种算法进行处理，使其成为不可读的一段字符，通常称为“密文”。只能在输入相应的密钥之后才能显示出本来内容。其逆过程为解密，即将该“密文”转化为其原来数据的过程。加密技术通常分为两大类：“对称式”和“非对称式”。 DES（Data Encryption Standard）：对称算法，数据加密标准，速度较快，适用于加密大量数据的场合。RSA：是一个支持变长密钥的公共密钥算法，需要加密的文件块的长度也是可变的，非对称算法，速度较慢，适用于小数据量加密。

RSA 公钥加密算法是第一个既能用于数据加密也能用于数字签名的算法。它易于理解和操作，也十分流行。算法的名字以发明者的姓氏首字母命名：Ron Rivest, Adi Shamir 和 Leonard Adleman。虽然自 1978 年提出以来，RSA 的安全性一直未能得到理论上的证明，但它经历了各种攻击，至今（2006 年）未被完全攻破。随着越来越多的商业应用和标准化工作，RSA 已经成为最具代表性的公钥加密技术。VISA、MasterCard、IBM、Microsoft 等公司协力制定的安全电子交易标准（Secure Electronic Transactions，SET）就采用了标准 RSA 算法，这使得 RSA 在我们的生活中几乎无处不在。网上交易加密连接、网上银行身份验证、各种信用卡使用的数字证书、智能移动电话和存储卡的验证功能芯片等，大多数使用 RSA 技术。

当今公钥加密更广泛应用于互联网身份认证，本课题将公钥加密算法 RSA 应用于信息加密。将任意信息加密成文本的解决方案，使其使用更加灵活。整个工程的分层设计，给后续开发带来便利。

样本
仅供参考
严禁抄袭

2. RSA 应用现状及应用于文件加密的分析

2.1 RSA 算法的介绍与应用现状

RSA 加密算法是基于一个十分简单的数论事实：将两个大素数相乘很容易，但要对其乘积进行因式分解则十分的困难。因此，可将两个大素数的乘积作为公开的密钥用于加密。RSA 加密算法是一种非对称密码算法。所谓非对称，就是指该算法需要一对密钥，使用一个用于加密，另一个用于解密[1]。道用于加密的密钥，也不能通过该密钥解得加密的内容。必须拥有解密的密钥才能解得加密的内容。RSA 加密与解密流程如图 1 和图 2。

RSA 加密算法[2]可简单的表述为：
<获取密钥>
取两个随机的大素数 P 和 Q，至少为 1024 位;
N = P * Q;
取 E1，使得 E1 与（P - 1）*（Q - 1）互质;
再取 E2，使得（E1 * E2）MOD（(P -1）*（Q – 1)）= 1;
取（N, E1）为公钥，（N, E2）为密钥;
<加密与解密>
设 A 为明文，B 为密文;
加密：B = A ^ E1 mod N;
解密：A = B ^ E2 mod N;

1

样本
仅供参考
严禁抄袭

[1] (美)Steve Burnett(美)Stephen Paine 著冯登国等译. 密码工程实践指南. 清华大学出版社，2001
[2] (美)斯托林斯著. 白国强等译. 网络安全基础. 清华大学出版社, 2011

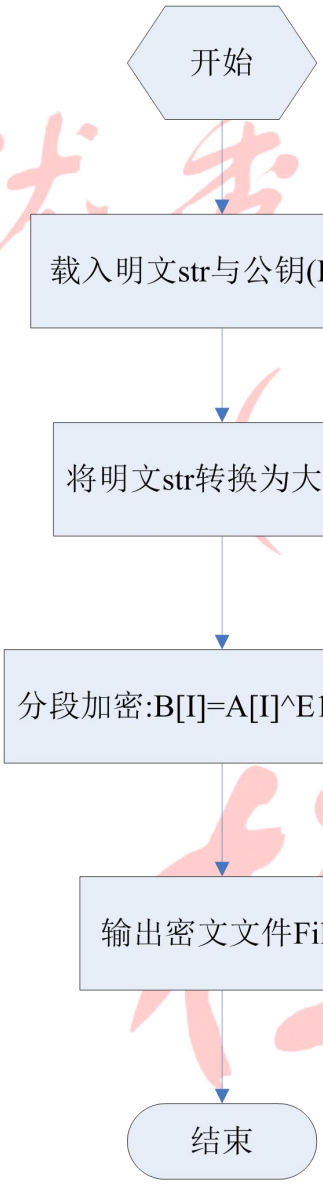

图 1　RSA 加密过程

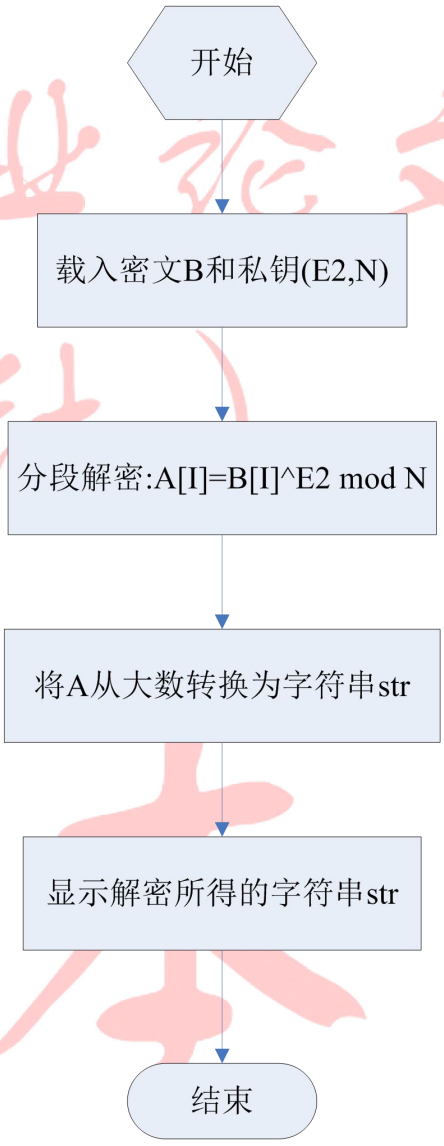

图 2 RSA 解密过程

RSA 在软件方面的应用，主要集中在互联网上。加密连接、数字签名和数字证书的核心算法广泛使用 RSA。日常应用中，比较著名的工具包有 Open SSL(SSL，Security Socket Layer,是一个安全传输协议，在 Internet 上进行数据保护和身份确认。Open SSL 是一个开放源代码的实现了 SSL 及相关加密技术的软件包，由加拿大的 Eric Yang 等发起编写的[1]。绍见 http://www.openssl.org/about/)。Open SSL 应用 RSA 实现签名和密钥交换，已经在各种操作系统得到非常广泛的应用。另外，家喻户晓的 IE 浏览器，自然也实现了 SSL 协议，集成了使用 RSA 技术的加密功能，结合 MD5 和 SHA1，主要用于数字证书和数字签名，对于习惯于使用网上购物和网上银行的用户来说，几乎天天都在使用 RSA 技术。

RSA 更出现在要求高度安全稳定的企业级商务应用中。在当今的企业级商务应用

[1] (美)卡茨,(以色列)耶胡达•林德尔著.任伟译.现代密码学：原理与协议.国防工业出版社,2012

中，不得不提及使用最广泛的平台 j2ee。事实上，在 j2se 的标准库中，就为安全和加密服务提供了两组 API：JCA 和 JCE。 JCA (Java Cryptography Architecture)提供基本的加密框架，如证书、数字签名、报文摘要和密钥对产生器；JCA 由几个实现了基本的加密技术功能的类和接口组成，其中最主要的是 java.security 包，此软件包包含的是一组核心的类和接口，Java 中数字签名的方法就集中在此软件包中。JCE(Java Cryptography Extension) 在 JCA 的基础上作了扩展，JCE 也是由几个软件包组成，其中最主要的是 javax.crypto 包，此软件包提供了 JCE 加密技术操作 API。javax.crypto 中的 Cipher 类用于具体的加密和解密。在上述软件包的实现中，集成了应用 RSA 算法的各种数据加密规范(RSA 算法应用规范介绍参见：http://www.rsasecurity.com/rsalabs/node.asp?id=2146，这些 API 内部支持的算法不仅仅只有 RSA，但是 RSA 是数字签名和证书中最常用的)，用户程序可以直接使用 java 标准库中提供的 API 进行数字签名和证书的各种操作。

2.2 RSA 算法用于信息加密的分析

当今 RSA 算法常用于数字签名和证书方面。之所以只用在这些短小数据的加密，是因为其加密和解密的速度极慢。因此，RSA 用于文件加密常被忽略。文件通常都被想象成一些很大的数据块集合。但是，像一些极其重要的文件。如：银行卡账号、各类密码、游戏账号等等。这些都是一些几 K 的文件。对这些文件进行加密相当于对一串长度很小的字符串进行加密。其时间是可观的。若要对大文件进行加密，可先将其切分为多块数据集合。再对每一块数据进行加密解密。其所消耗的时间也是可接受的。

若不经加密，便将上述提及的重要文件置于联网的或公共计算机上。这样是十分不安全的。使用对称加密算法加密重要的文件只适合部分情况。若文件被置于公共计算机上，任何人只要知道加密的密钥，便可得到正文内容。如以下情况：张三使用对称加密算法将的加密的密钥留在学校机房的公共计算机上，供李四使用。若王五使用了该台计算机，便可获得李四的留给张三的正文。但是，若张三使用的是 RSA 加密算法。即使，王五知道加密的密钥也无法获得张三留给李四的正文。这就类似于，你想在某某论坛上留言给某人，并且不想让其他无关的人获得正文。使用 RSA 加密算法便可达到这一点。

仅供参考
严禁抄袭

3. RSA 文件加密软件的设计与实现

3.1 需求分析与总体设计

a. 可按用户输入的位数生成相应位数的密钥对；
b. 可保存和加载密钥对，且保存为纯文本；
c. 可对任意字符串进行加密，加密后的密文保存为纯文本；
d. 可对任意密文进行解密，解密后的正文保存为纯文本呢
以下给出关键类图如图 3：

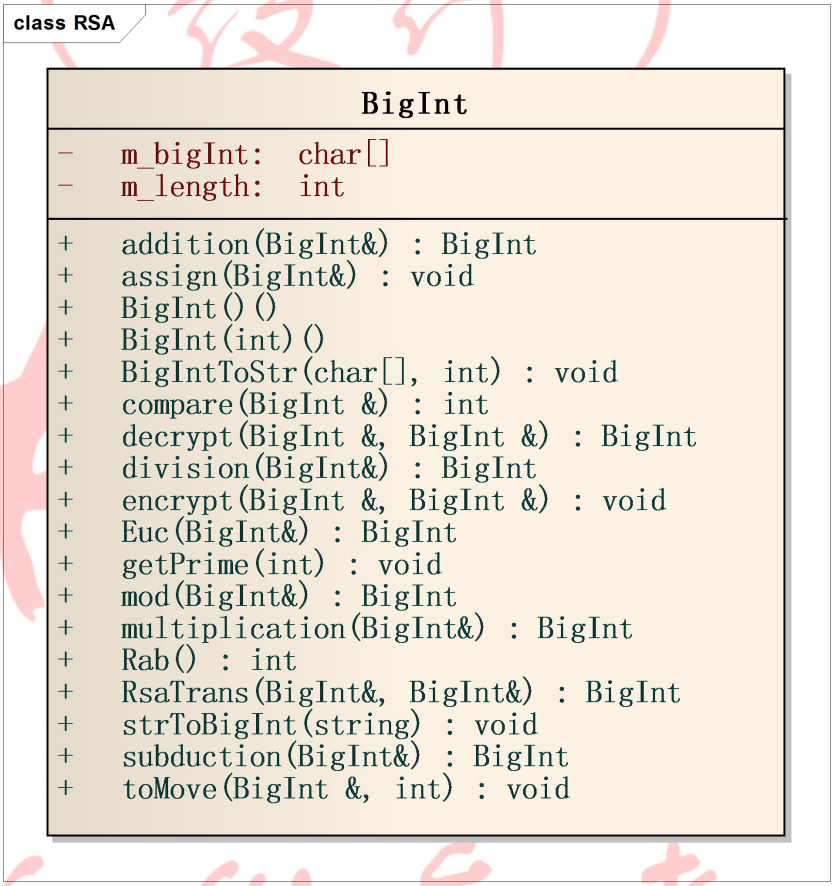

图 3 大数类 BigInt 类图

3.2 各部分的设计与开发

3.2.1 大数类的设计与实现

RSA 算法要求进行大素数的各种运算。为此，需要首先实现大数的存储和其四则运

算。在此基础之上，再扩展 RSA 所需的大数运算。以下介绍大数的存储和其四则运算。

大数的存储：

存储功能由 BigInt 类提供。每一个大数对应一个 BigInt 的实例。每个 BigInt 实例包含一个用于指定大数长度的 int 型变量和一个用于存储大数的 char 数组。数组的最大容量为 128。对于 32 位的 OS，一个 1024 位的大数需要长度为 128 的 char 数组。对于 64 位的 OS，一个 1024 为的大数需要长度为 64 的 char 数组。数组的起始位表示大数的低位。即数组的大索引对应大数的高位。如：0 表示个位，1 表示十位，2 表示百位等等。数组中的每一位数字的大小都介于 0 到 9 之间。

大数的四则运算：

加法与减法：

大数的加减法运算相对于其他运算较简单。均使用“竖式运算”模拟。即从低位开始运算。两种算法的区别在于加法的当前位相加大于十位向高一位进一，而减法的当前位相减时，需比较大小。若被减数小于减数，则向高一位借一。并且，减法运算所得结果可能出现 001 此类情况。因此，减法运算结束后，还需检查结果的有效位。以下详细介绍大数的加法运算：将大数 A 和大数 B 对应相同位上的值相加。若当前位相加所得的值置于 Sum。用 10 整除 Sum 得出进位值。用 10 与 Sum 进行模运算得出当前值。如此类推，遍历完较长数的每一位为止。加法的流程图如图 4。以下为大数加法的代码:

```
BigInt BigInt::addition( BigInt& num )
{
    BigInt result;          //相加的和
    int sum = 0;            //临时存储空间
    int carry = 0;          //进位值
    //取较大值的长度为相加结果的长度

    if (m_length < num.m_length)
    {
        result.m_length = num.m_length;
    }
    else
    {
        result.m_length = m_length;
    }


    //将对应为相加
    for (int index = 0; index < result.m_length; ++index)
    {
        //96 为字符'0'和'0'相加的 int 型值
        //将相加所得的和与 96 取余可得字面意义上相加的和
        sum = (m_bigInt[index] + num.m_bigInt[index]) % 96 + carry;
```

```
            if (10 <= sum)
            {
                carry = 1;
                sum = sum % 10;              //与 10 取余得到个位的值,既当前位的值
            }
            else
            {
                carry = 0;
            }

            result.m_bigInt[index] = (char)(sum + ZERO);
        }
        //处理最后一次对应位相加的进位值不为 0
        if (carry)
        {
            ++result.m_length;
            result.m_bigInt[result.m_length - 1] = (char)(carry + ZERO);
        }

        return result;
    }
```

优秀毕业论文（设计）
样本
仅供参考
严禁抄袭

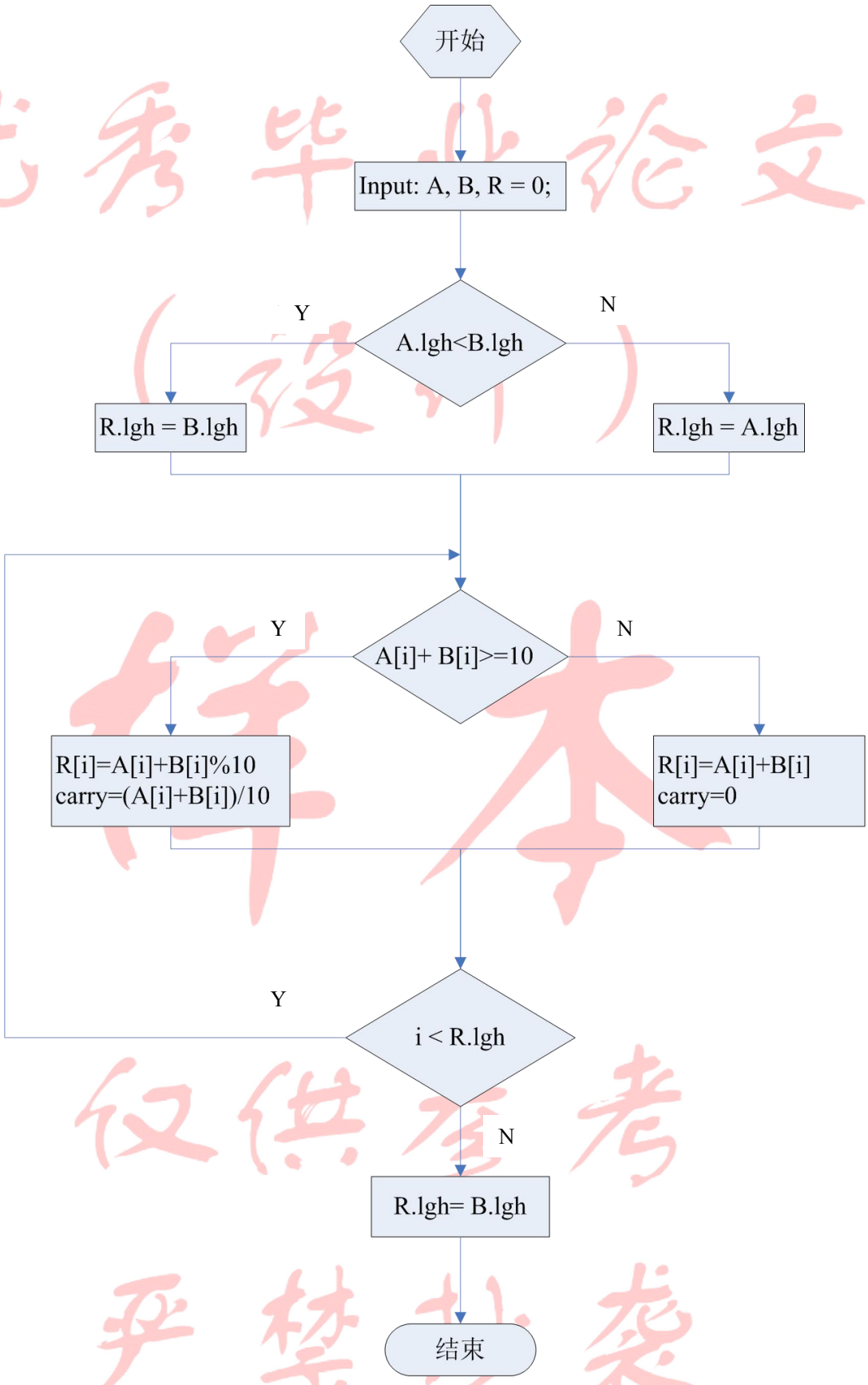
开始
Input: A, B, R = 0;
A.lgh<B.lgh
Y
N
R.lgh = B.lgh
R.lgh = A.lgh
A[i]+ B[i]>=10
Y
N
R[i]=A[i]+B[i]%10
carry=(A[i]+B[i])/10
R[i]=A[i]+B[i]
carry=0
i < R.lgh
Y
N
R.lgh= B.lgh
结束

图 4 加法流程图

乘法：

大数的乘法使用“竖式运算”模拟。循环使用乘数 B 的一位 Bi 乘被乘数 A 的每一位。将所得积 Temp 与乘数的上一位乘得的积 Result 从对应位 i 开始相加。乘数 B 与被乘数 A 的对应位 i 相乘得出 Sum。用 10 整除 sum 得出进位值。用 10 与 Sum 进行模运算得出当前值。乘法的流程图如图 5。以下为大数乘法的代码：

```
BigInt BigInt::multiplication( BigInt& num )
{
    BigInt result;
    BigInt temp;
    char str = NULL;
    int A = 0;
    int B = 0;
    int sum = 0;
    int carry = 0;

    //乘积的长度等于乘数的长度 + 被乘数的长度
    result.m_length = m_length + num.m_length;

    //对应为相乘
    for (int index_n = 0; index_n < num.m_length; ++index_n)
    {
        str = num.m_bigInt[index_n];
        A = atoi(&str);
        int index_r = 0;
        temp.assign(BigInt(0));

        while ( index_r < m_length )
        {
            str = m_bigInt[index_r];
            B = atoi(&str);
            sum = A * B + carry;
            carry = sum / 10;
            sum = sum % 10;
            temp.m_bigInt[index_r] = (char)(sum + ZERO);
            ++index_r;
        }

        if (carry)
        {
            temp.m_bigInt[index_r] = (char)(carry + ZERO);
            carry = 0;
        }

        int index = index_n;
        int index_t = 0;

        while ( index < result.m_length )
```

```
            {
                sum = (result.m_bigInt[index] + temp.m_bigInt[index_t++]) % 96 + carry;
                result.m_bigInt[index] = (char)(sum % 10 + ZERO);
                carry = sum / 10;
                ++index;
            }

            if (carry)
            {
                result.m_bigInt[index] = (char)(carry + ZERO);
                carry = 0;
            }
        }

        for (int index = result.m_length - 1; index > 0; --index)
        {
            if ('0' != result.m_bigInt[index])
            {
                break;
            }
            --result.m_length;
        }

        return result;
    }
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

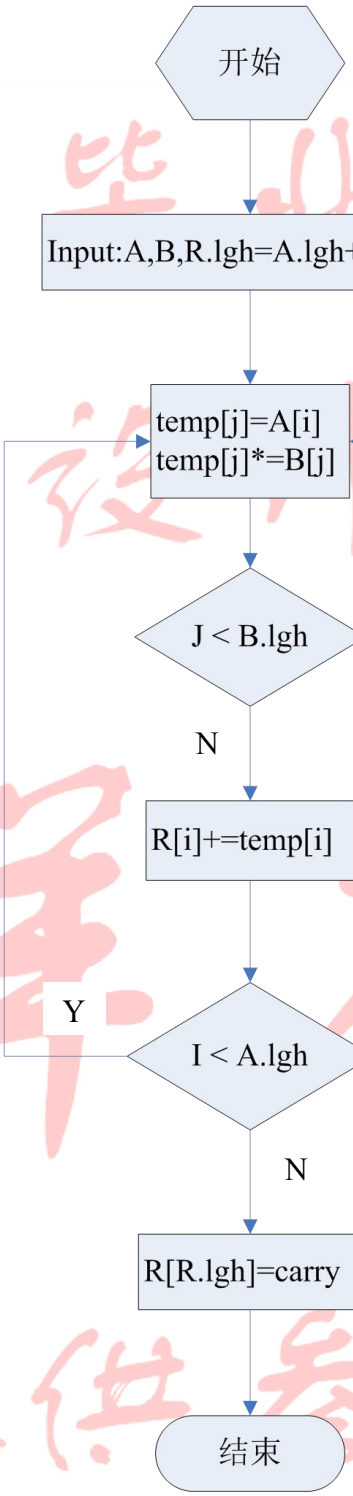

图 5 乘法流程图

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

除法：

大数的除法可使用减法来实现。1.比较被除数 A 和除数 B 的大小。如果被除数 A 小于除数 B，则返回 0。因为该大数除法实现为整除。例如，1 / 2 的结果为 0。而非 0.5。又或，5 / 2 的结果为 2。而非 2.5。2.将除数 B 移位直到最高位与被除数 A 最高位对齐。低位为空的位置用 0 填补。如被除数为：123，456，789，除数为：123，456。除数移位后为 123，456，000。3.比较移位后的除数 C 是否小于被除数 A。若，除数 C 大于被除数 A。则，向右移位直至小于被除数 A 为止。4.被除数 A 减除数 C，将所得值赋予 A，记录 A 减 C 的次数为 D。比较被除数 A 与除数 C 的大小。若除数 C 大于被除数 A，循环执行 2 和 3 直到 B 大于 A 为止。否则，继续执行 4。商为 D。除法的流程图如图 6。以下为除法的伪算法：

```
BigInt BigInt::division( BigInt& Num)
{
    BigInt result;
    BigInt copyNum;
    BigInt sum;
    BigInt count;
    BigInt one(1);
    result.assign(*this);
    copyNum.assign(Num);

    if (0 > result.compare(Num))
    {
        result.assign(BigInt(0));
        return result;
    }

    if (0 == result.compare(Num))
    {
        result.assign(one);
        return result;
    }

    //结束除法的条件为:余数小于除数
    while (0 < result.compare(Num))
    {
        //余数与除数比较有效数字的位数,得出该移多少位
        int length = result.m_length - copyNum.m_length;

        //若位数不为 0,进行移位
        if (0 != length)
        {
            count.assign(one);
            //移位操作
            toMove(copyNum, length);
            //移位后比较余数与除数的大小,除数大,则向右移位,直到小于余数
            while (0 > result.compare(copyNum))
```

```
            {
                --length;
                toMove(copyNum, -1);
            }
            toMove(count, length);
        }

        int count2 = 0;
        //余数循环减移位后的除数,直到余数小于移位后的除数
        while (0 <= result.compare(copyNum))
        {
            result.assign(result.subduction(copyNum));
            ++count2;
        }
        if (length)
        {
            count.assign(count.multiplication(BigInt(count2)));
            sum.assign(sum.addition(count));
        }
        else
        {
            sum.assign(sum.addition(BigInt(count2)));
        }
        copyNum.assign(Num);
    }

    result.assign(sum);
    for (int index = result.m_length - 1; index > 0; --index)
    {
        if ('0' != result.m_bigInt[index])
        {
            break;
        }
        --result.m_length;
    }
    return result;
}
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

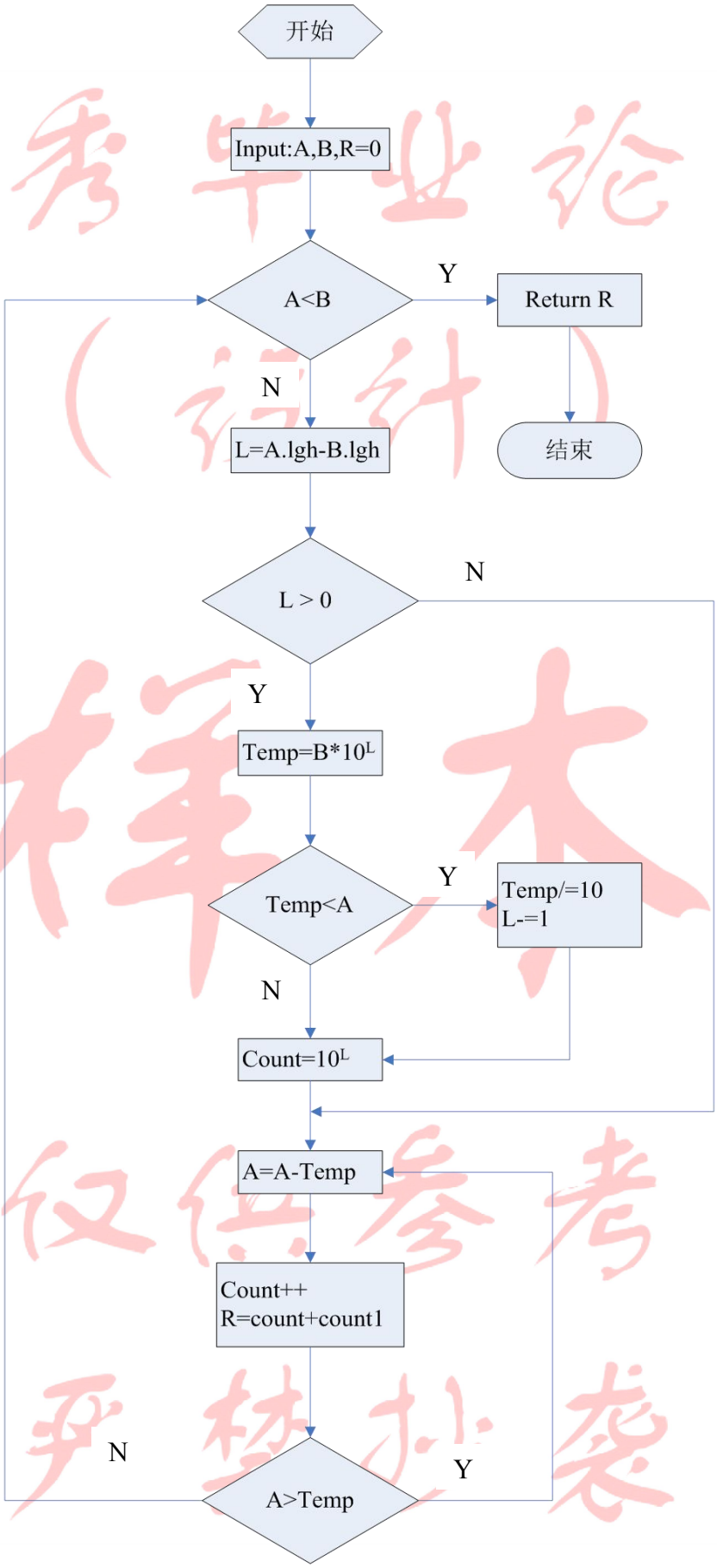
开始
Input:A,B,R=0
A<B
Y
Return R
N
结束
L=A.lgh-B.lgh
L > 0
N
Y
Temp=B*10^L
Temp<A
Y
Temp/=10
L-=1
N
Count=10^L
A=A-Temp
Count++
R=count+count1
N
A>Temp
Y

图 6 除法流程图

3.2.2 模运算

模运算：

一般的模运算大多以加减乘三个算法实现。但是，若使用这三个算法实现，模运算的效率十分的地下。因此，本算法使用循环减去减数的做法。具体做法为：1、取被减数 A 和减数 B，计算被减数 A 与减数 B 相差的位数 length。2、若 length 不为 0，则减数 B 向左移 length 位。移位后比较 B1 与 A 的大小。若 B1 大于 A，则 B1 向右移一位。3、用被减数 A 循环减 B1，将减得的结果赋予 A。比较 A 与 B1 的大小。若 A 小于 B1，则执行 1。否则，继续执行 3。除法流程图如图 7。以下为模运算的伪代码：

```
BigInt BigInt::mod( BigInt& Num )
{
    BigInt result;
    BigInt copyNum;
    result.assign(*this);
    copyNum.assign(Num);
    if (0 > result.compare(Num))
    {
        return result;
    }
    if (0 == result.compare(Num))
    {
        result.assign(BigInt(0));
        return result;
    }
    //结束除法的条件为:余数小于除数
    while (0 < result.compare(Num))
    {
        //余数与除数比较有效数字的位数,得出该移多少位
        int length = result.m_length - copyNum.m_length;
        //若位数不为 0,进行移位
        if (0 != length)
        {
            //移位操作
            toMove(copyNum, length);
            //移位后比较余数与除数的大小,除数大,则向右移位,直到小于余数
            while (0 > result.compare(copyNum))
            {
                toMove(copyNum, -1);
            }
        }
        //余数循环减移位后的除数,直到余数小于移位后的除数
        while (0 <= result.compare(copyNum))
        {
            result.assign(result.subduction(copyNum));
        }
        copyNum.assign(Num);
    }
    return result;
```

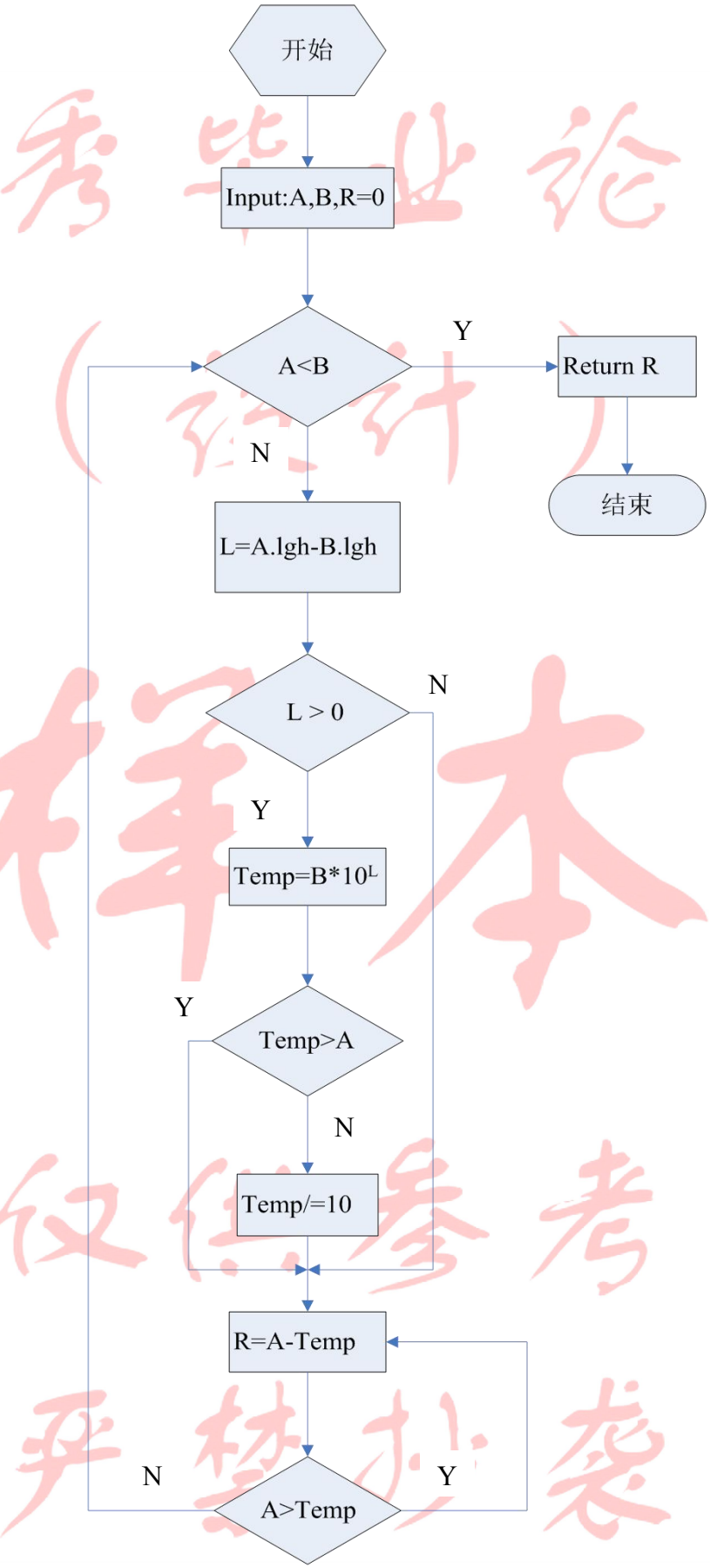
开始
Input:A,B,R=0
A<B
Y
Return R
结束
N
L=A.lgh-B.lgh
L > 0
N
Y
Temp=B*10^L
Y
Temp>A
N
Temp/=10
R=A-Temp
N
A>Temp
Y

图 7 模运算流程图

3.2.3 素数判定算法的实现

测试一个随机数是否为素数，有各式各样的算法。本文选用拉宾-米勒算法结合一个小素数表测试随机生成的大数是否为素数。拉宾-米勒算法[1]是典型的验证一个数是否为素数的算法。该算法是一个判断素性的多项式时间概率算法。由于米勒-拉宾法的非确定性，当我们对判定素数的要求仅为需要偏“是”的判定法时，用米勒-拉宾法比较容易，但是如果我们需要确定解时仍然要依靠速度较慢的其他确定性素性判定法。测试素数的方法是拉宾-米勒概率测试。一个随机数通过一次拉宾-米勒算法的测试，其为素数的概率为 75%。因此，对一个随机数进行多次的素数测试,可保证其素数性达 99%。

其过程如下：
Miller-Rabin(n)
把 n-1 写成 n-1=2^k *m ，其中 m 是一个奇数
选取随机整数 a，使得 1<=a<=n-1
将 a^m(mod n)赋值给 b,
if b=1(mod n)
then return(“n is prime”)
for 将 0 赋值给 i to k-1
do { if b=-1(mod n)
then return ("n is prime")
else 将 b^2(mod n)赋值给 b
}
return(“n is composite”)
若 n 通过一次测试，则 n 不是素数的概率为 25%。以下为拉宾米勒的代码。

```
int BigInt::Rab()
{
    int pass = 0;
    BigInt temp1;
    BigInt zero(0);
    //使用小素数表筛选大数
    for (int index = 0; index < TABLE_SIZE; ++index)
    {
        temp1.assign(mod(BigInt(PrimeTable[index])));
        if (0 == temp1.compare(zero))
        {
            return pass;
        }
    }
    BigInt S, A, I, K;
    BigInt two(2);
    K.assign(*this);
    K.assign(K.subduction(BigInt(1)));
    srand(time(NULL));
    int randnum = rand();
```

[1] (美)(圣丹斯 Denis)(T.S.)著.尹浩琼等译.BigNum Math：加密多精度算法的理论与实现.中国水利水电出版社,2008

```
        A.assign(BigInt(randnum));
        S.assign(K);
        while(1 != S.m_length || '0' != S.m_bigInt[0])
        {
            S.assign(S.division(two));
            I.assign(A.RsaTrans(S, *this));
            if(I.compare(K) == 0)
            {
                pass = 1;
                break;
            }
        }
        if((I.m_length == 1) && (I.m_bigInt[0] == '1'))
        {
            pass = 1;
        }
        if(pass == 0)
        {
            return pass;
        }

        return pass;
    }
```

3.2.4 RSA 算法的实现

RSA 类的实现是基于前述的 BigInt 类，主要实现 RSA 加密算法对明文的加密与解密。RSA 类中，随机大素数的产生与大数的计算均由前述的 BigInt 提供。在类的构造函数中，生成一对密钥对，用于对明文的加解密。密钥对可即时生成新的，也可加载以往生成后置于文件中的旧密钥对。RSA 类对外只提供加密和解密的两个接口。输入输出均为字符串。以下为伪代码：

```
        BigInt P;
        BigInt Q;
        BigInt N;
        BigInt E1(11);
        BigInt E2;
        BigInt MW;
        BigInt result1;
        BigInt result2;

        P.getPrime(1024);
        Q.getPrime(1024);

        N.assign(P.multiplication(Q));

        --P.m_bigInt[0];
        --Q.m_bigInt[0];
        E2.assign(E1.Euc(P.multiplication(Q)));
```

```
cout << "E: ";
for (int index = E2.m_length - 1; index >= 0; --index)
{
    cout << E2.m_bigInt[index];
}
cout << endl;
cout << endl;

string str = "&#$";
MW.strToBigInt(str);
MW.encrypt(E1, N);

result2.assign(BigInt::decrypt(E2, N));

char temp[20] = {'0'};
result2.BigIntToStr(temp, str.length());
```

优秀毕业论文（设计）
样本
仅供参考
严禁抄袭

4. 软件整体测试与分析改进

4.1 生成随机大素数测试

RSA 加密算法实现上，最耗时间的是产生密钥的部分。寻找一个 1024 位的大素数是一个项十分复杂的工作。其速度可能受以下几点影响：大数的幂模运算、大数的素性判定、计算不定方程解等。其中最具影响力的是大数的素性判断。以下为对不同长度随机大素数生成时间的统计。

表 1 生成随机素数耗时表

长度	测试五次随机大素数消耗的时间（秒）					消耗时间的平均值
	第一次	第二次	第三次	第四次	第五次	
64	0.156	0.140	0.124	0.156	0.140	0.1432
128	0.936	1.030	0.920	0.998	0.936	0.964
256	9.345	7.676	8.377	9.719	8.830	8.7894
512	91.962	90.325	90.870	88.592	89.076	90.135
1024	1083.56	1156.23	1050.45	1023.85	1123.5	1087.518

表 1 从实验的角度直观的看出，生成随机大素数的时间随着位数增长。但是，同长度的大素数每一次耗时是不确定的。由于素数分布规律非常奥妙，加上测试运算需要的时间颇长，这里很难给出对于一个具体位数的密钥生成所需时间的统计模型。

4.2 生成素数的准确性测试

素数的测试算法如本文 3.2.3 节所述，选用拉宾米勒素数测试法。使用该算法测试随机大数能达到高效率。但是，使用该算法测试一次只能保证准确率为 75%[1]。以下为五次测试结果：

表 2 生成随机素数准确率表

长度	测试五次随机大素数的准确性（Y/N）					准确率
	第一次	第二次	第三次	第四次	第五次	
64	Y	Y	Y	Y	Y	100%
128	Y	Y	Y	N	Y	80%
256	Y	N	Y	Y	Y	80%
512	Y	Y	Y	Y	Y	100%
1024	Y	Y	Y	Y	N	80%

表 2 从实验的角度可看出，素数的测试通过率为 80%左右。但是，这只是小次数的测试。从工程的角度来说是可取的。但是，从解决问题的角度来说是不严谨的。

[1]Douglas R.Stinson，冯登国著.密码学原理与实践.电子工业出版社，2009

4.3 加密效率测试

进行对任意信息串加密效率的测试。这里给出几组不同长度的密钥对进行加密效率测试。测试结果如表 3 所示。

表 3 加密效率的测试

密钥长度	测试五次加密消耗的时间（秒）					消耗时间的平均值
	第一次	第二次	第三次	第四次	第五次	
64	0.356	0.264	0.324	0.256	0.340	0.308
128	1.136	1.239	1.520	1.498	1.236	1.326
256	11.541	9.694	10.393	11.759	10.836	10.845
512	101.922	100.925	105.879	99.992	101.976	102.139
1024	1283.66	1356.93	1250.75	1123.85	1323.5	1267.738

4.4 加密与解密的测试

```
string str = "&#$*&*&%^$#%@#$%#^&^&@!)(";
MW.strToBigInt(str);
```

图 8 明文图

```
明文转换为大数: 038035036042038042038037094036035037064035036037035094038094038064033041040
```

图 9 明文转大数

```
解密所得大数: 040140330460830490830490530730630530460730530630490730830240830240630530830
```

图 10 解密所得结果

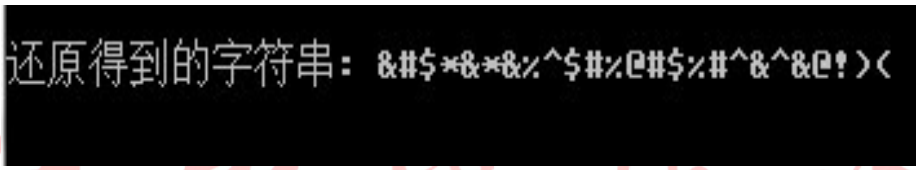

图 11 大数还原为字符串

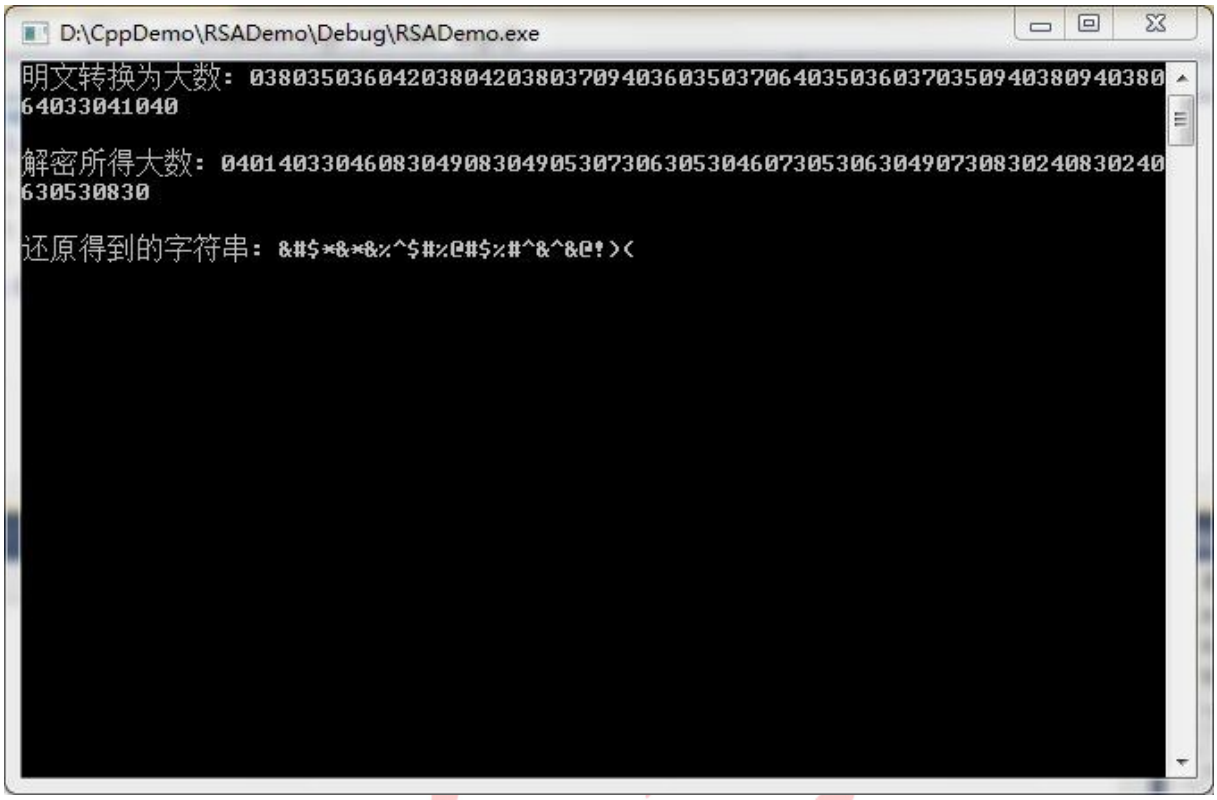

图 12 总体结果

优秀毕业论文
仅供参考
严禁抄袭

结语

RSA 加密算法可用于文件加密，适合于小型文件。将任意文件中内容以非对称密钥加密成密文文本可以增加其安全性，有广阔的开发前景。本项目应用的设计模式兼顾执行效率和可复用性。

毕业论文（设计）
样本
仅供参考
严禁抄袭

参考文献

[1] (美)(圣丹斯 Denis)(T.S.)著.尹浩琼等译.BigNum Math：加密多精度算法的理论与实现.中国水利水电出版社,2008
[2] (美)斯托林斯著.白国强等译.网络安全基础.清华大学出版社,2011
[3] (美)Steve Burnett(美)Stephen Paine 著冯登国等译.密码工程实践指南.清华大学出版社，2001
[4] 段钢著.加密与解密.电子工业出版社，2008
[5] (美)卡茨,(以色列)耶胡达·林德尔著.任伟译.现代密码学：原理与协议.国防工业出版社,2012
[6] BruceSchneier，吴世忠等著.应用密码学：协议算法与 C 源程序.机械工业出版社，2000
[7] Douglas R.Stinson，冯登国著.密码学原理与实践.电子工业出版社，2009
[8] (美)帕尔(Christof Pear)(美)佩尔茨尔(Jan Pelzl)著马小婷译.深入浅出密码学：常用加密技术原理与应用.清华大学出版社，2012
[9] (美)康海涛著唐明译.计算机安全与密码学.电子工业出版社，2010
[10] 林东岱，曹天杰著.应用密码学.科学出版社，2009
[11] 夏娜等著.信息编码与加密实践.合肥工业大学出版社，2008
[12] 郑东,李祥学,黄征著.密码算法与协议.电子工业出版社,2009
[13] 董秀则等著.路而红编.现代密码算法工程.清华大学出版社,2012
[14] (德尔夫斯 Delfs)(H.)(克内贝尔 Knebl()H.)著.密码学导引：原理与应用.清华大学出版社,2007
[15] 威廉·斯托林斯(WilliamStallings)著.王张宜译.密码编码学与网络安全：原理与实践(第 5 版).电子工业出版社,2011

仅供参考
严禁抄袭

致　谢

历时将近五个月的时间终于将把论文写完，在论文的写作过程中遇到了各种各样的困难和障碍，大多在朋友和老师的指导下度过了。尤其要强烈感谢我的论文指导老师—陈红顺老师，他给了我许多无私的指导和帮助，不厌其烦的帮助我进行论文的修改和改进。感谢这篇论文所涉及到的各位学者。本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正。

（设计）
样本
仅供参考
严禁抄袭

附 录

大数类 BigInt 的实现：

BigInt.h

```
#pragma once
#define MAXSIZE 280
class BigInt
{
public:
	BigInt(void);
	BigInt(int num);
	~BigInt(void);
public:
	void getPrime(int);
	BigInt addition(BigInt& num);
	BigInt subduction(BigInt& num);
	BigInt multiplication(BigInt& num);
	BigInt division(BigInt& num);
	BigInt mod(BigInt& num);
	void assign(BigInt& num);
	int compare(BigInt &num);
	BigInt RsaTrans(BigInt& A, BigInt& B);
	void toMove(BigInt &num, int length);
	int Rab();
	BigInt Euc(BigInt& A);
	void strToBigInt(string messages);
	void BigIntToStr(char result[], int count);
	void encrypt(BigInt &E1, BigInt &N);
	static BigInt decrypt(BigInt &E2, BigInt &N);

public:
	int m_length;
	char m_bigInt[MAXSIZE];
};
```

BigInt.cpp

```
#include "StdAfx.h"
#include "BigInt.h"
#include <time.h>
#include <string.h>
#include <Windows.h>
#include <fstream>
#define TABLE_SIZE 550
#define ZERO 48
#define NINE 57
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
const static int PrimeTable[TABLE_SIZE] = {
    3,    5,    7,    11,   13,   17,   19,   23,   29,   31,
    37,   41,   43,   47,   53,   59,   61,   67,   71,   73,
    79,   83,   89,   97,   101,  103,  107,  109,  113,  127,
    131,  137,  139,  149,  151,  157,  163,  167,  173,  179,
    181,  191,  193,  197,  199,  211,  223,  227,  229,  233,
    239,  241,  251,  257,  263,  269,  271,  277,  281,  283,
    293,  307,  311,  313,  317,  331,  337,  347,  349,  353,
    359,  367,  373,  379,  383,  389,  397,  401,  409,  419,
    421,  431,  433,  439,  443,  449,  457,  461,  463,  467,
    479,  487,  491,  499,  503,  509,  521,  523,  541,  547,
    557,  563,  569,  571,  577,  587,  593,  599,  601,  607,
    613,  617,  619,  631,  641,  643,  647,  653,  659,  661,
    673,  677,  683,  691,  701,  709,  719,  727,  733,  739,
    743,  751,  757,  761,  769,  773,  787,  797,  809,  811,
    821,  823,  827,  829,  839,  853,  857,  859,  863,  877,
    881,  883,  887,  907,  911,  919,  929,  937,  941,  947,
    953,  967,  971,  977,  983,  991,  997,  1009, 1013, 1019,
    1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087,
    1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153,
    1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229,
    1231, 1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297,
    1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381,
    1399, 1409, 1423, 1427, 1429, 1433, 1439, 1447, 1451, 1453,
    1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 1523,
    1531, 1543, 1549, 1553, 1559, 1567, 1571, 1579, 1583, 1597,
    1601, 1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663,
    1667, 1669, 1693, 1697, 1699, 1709, 1721, 1723, 1733, 1741,
    1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811, 1823,
    1831, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901,
    1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993,
    1997, 1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063,
    2069, 2081, 2083, 2087, 2089, 2099, 2111, 2113, 2129, 2131,
    2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221,
    2237, 2239, 2243, 2251, 2267, 2269, 2273, 2281, 2287, 2293,
    2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371,
    2377, 2381, 2383, 2389, 2393, 2399, 2411, 2417, 2423, 2437,
    2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539,
    2543, 2549, 2551, 2557, 2579, 2591, 2593, 2609, 2617, 2621,
    2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687, 2689,
    2693, 2699, 2707, 2711, 2713, 2719, 2729, 2731, 2741, 2749,
    2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803, 2819, 2833,
    2837, 2843, 2851, 2857, 2861, 2879, 2887, 2897, 2903, 2909,
    2917, 2927, 2939, 2953, 2957, 2963, 2969, 2971, 2999, 3001,
    3011, 3019, 3023, 3037, 3041, 3049, 3061, 3067, 3079, 3083,
    3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169, 3181, 3187,
    3191, 3203, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259,
    3271, 3299, 3301, 3307, 3313, 3319, 3323, 3329, 3331, 3343,
    3347, 3359, 3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433,
    3449, 3457, 3461, 3463, 3467, 3469, 3491, 3499, 3511, 3517,
```

```
    3527, 3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581,
    3583, 3593, 3607, 3613, 3617, 3623, 3631, 3637, 3643, 3659,
    3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733,
    3739, 3761, 3767, 3769, 3779, 3793, 3797, 3803, 3821, 3823,
    3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911,
    3917, 3919, 3923, 3929, 3931, 3943, 3947, 3967, 3989, 4001
};

BigInt::BigInt(void)
{
    m_length = 1;
    for (int index = 0; index < MAXSIZE; ++index)
    {
        m_bigInt[index] = '0';
    }
}

BigInt::BigInt(int num)
{
    for (int index = 0; index < MAXSIZE; ++index)
    {
        m_bigInt[index] = '0';
    }

    if (0 == num)
    {
        m_length = 1;
    }
    else
    {
        char temp[128] = {0};
        itoa(num, temp, 10);

        m_length = strlen(temp);
        int ths_index = m_length - 1;
        for (int index = 0; index < m_length; ++index)
        {
            m_bigInt[ths_index--] = temp[index];
        }
    }
}

BigInt::~BigInt(void)
{
}

BigInt BigInt::addition( BigInt& num )
{
    BigInt result;
    int sum = 0;
```

纸质毕业论文（设计）
样本
仅供参考
严禁抄袭

```
    int carry = 0;

    if (m_length < num.m_length)
    {
        result.m_length = num.m_length;
    }
    else
    {
        result.m_length = m_length;
    }

    for (int index = 0; index < result.m_length; ++index)
    {
        sum = (m_bigInt[index] + num.m_bigInt[index]) % 96 + carry;
        if (10 <= sum)
        {
            carry = 1;
            sum = sum % 10;
        }
        else
        {
            carry = 0;
        }
        result.m_bigInt[index] = (char)(sum + ZERO);
    }

    if (carry)
    {
        ++result.m_length;
        result.m_bigInt[result.m_length - 1] = (char)(carry + ZERO);
    }
    return result;
}

BigInt BigInt::subduction( BigInt& num )
{
    BigInt result;
    if (0 >= this->compare(num))
    {
        result.m_length = 1;
        result.assign(BigInt(0));
        return result;
    }
    result.m_length = m_length;
    int carry = 0;
    int temp = 0;

    for(int index = 0; index < m_length; ++index)
    {
        if (m_bigInt[index] < num.m_bigInt[index]
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
            || (m_bigInt[index] == num.m_bigInt[index] && carry != 0))
        {
            temp = m_bigInt[index] - num.m_bigInt[index] + 10 - carry;
            carry = 1;
        }
        else
        {
            temp = m_bigInt[index] - num.m_bigInt[index] - carry;
            carry = 0;
        }

        result.m_bigInt[index] = (char)(temp + 48);
    }

    for (int index = result.m_length - 1; index > 0; --index)
    {
        if ('0' != result.m_bigInt[index])
        {
            break;
        }
        --result.m_length;
    }

    return result;
}

BigInt BigInt::multiplication( BigInt& num )
{
    BigInt result;
    BigInt temp;
    char str = NULL;
    int A = 0;
    int B = 0;
    int sum = 0;
    int carry = 0;

    result.m_length = m_length + num.m_length;
    for (int index_n = 0; index_n < num.m_length; ++index_n)
    {
        str = num.m_bigInt[index_n];
        A = atoi(&str);
        int index_r = 0;
        temp.assign(BigInt(0));
        while ( index_r < m_length )
        {
            str = m_bigInt[index_r];
            B = atoi(&str);
            sum = A * B + carry;
            carry = sum / 10;
            sum = sum % 10;
```

优秀毕业论文（设计）样本 仅供参考 未经抄袭

```
            temp.m_bigInt[index_r] = (char)(sum + ZERO);
            ++index_r;
        }

        if (carry)
        {
            temp.m_bigInt[index_r] = (char)(carry + ZERO);
            carry = 0;
        }

        int index = index_n;
        int index_t = 0;
        while ( index < result.m_length )
        {
            sum = (result.m_bigInt[index] + temp.m_bigInt[index_t++]) % 96 + carry;
            result.m_bigInt[index] = (char)(sum % 10 + ZERO);
            carry = sum / 10;
            ++index;
        }

        if (carry)
        {
            result.m_bigInt[index] = (char)(carry + ZERO);
            carry = 0;
        }
    }

    for (int index = result.m_length - 1; index > 0; --index)
    {
        if ('0' != result.m_bigInt[index])
        {
            break;
        }
        --result.m_length;
    }

    return result;
}

BigInt BigInt::division( BigInt& Num)
{
    BigInt result;
    BigInt copyNum;
    BigInt sum;
    BigInt count;
    BigInt one(1);
    result.assign(*this);
    copyNum.assign(Num);

    if (0 > result.compare(Num))
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
{
    result.assign(BigInt(0));
    return result;
}
if (0 == result.compare(Num))
{
    result.assign(one);
    return result;
}

while (0 < result.compare(Num))
{
    int length = result.m_length - copyNum.m_length;

    if (0 != length)
    {
        count.assign(one);
        toMove(copyNum, length);
        while (0 > result.compare(copyNum))
        {
            --length;
            toMove(copyNum, -1);
        }

        toMove(count, length);
    }

    int count2 = 0;
    while (0 <= result.compare(copyNum))
    {
        result.assign(result.subduction(copyNum));
        ++count2;
    }
    if (length)
    {
        count.assign(count.multiplication(BigInt(count2)));
        sum.assign(sum.addition(count));
    }
    else
    {
        sum.assign(sum.addition(BigInt(count2)));
    }
    copyNum.assign(Num);
}

result.assign(sum);
for (int index = result.m_length - 1; index > 0; --index)
{
    if ('0' != result.m_bigInt[index])
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
            {
                break;
            }
            --result.m_length;
        }
        return result;
}

BigInt BigInt::mod( BigInt& Num )
{
    BigInt result;
    BigInt copyNum;
    result.assign(*this);
    copyNum.assign(Num);

    if (0 > result.compare(Num))
    {
        return result;
    }
    if (0 == result.compare(Num))
    {
        result.assign(BigInt(0));
        return result;
    }
    while (0 < result.compare(Num))
    {
        int length = result.m_length - copyNum.m_length;

        if (0 != length)
        {
            toMove(copyNum, length);
            while (0 > result.compare(copyNum))
            {
                toMove(copyNum, -1);
            }
        }

        while (0 <= result.compare(copyNum))
        {
            result.assign(result.subduction(copyNum));
        }

        copyNum.assign(Num);
    }
    return result;
}

int BigInt::compare( BigInt &num )
{
    if (m_length < num.m_length)
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
    {
        return    -1;
    }
    if (m_length > num.m_length)
    {
        return 1;
    }
    for (int index = m_length - 1; index >= 0; --index)
    {
        if (m_bigInt[index] < num.m_bigInt[index])
        {
            return -1;
        }
        else if (m_bigInt[index] > num.m_bigInt[index])
        {
            return 1;
        }
    }
    return 0;
}

void BigInt::assign( BigInt& num )
{
    for (int index = 0; index < MAXSIZE; ++index)
    {
        m_bigInt[index] = '0';
    }

    m_length = num.m_length;
    for (int index = m_length - 1; index >= 0; --index)
    {
        m_bigInt[index] = num.m_bigInt[index];
    }
    return;
}

void BigInt::getPrime(int length)
{
    m_length = length / 8;
    srand(time(NULL));
begin:
    for (int index = 0; index < m_length; ++index)
    {
        int temp    = 9 * rand() / (RAND_MAX + 1);
        m_bigInt[index] = (char)(temp + ZERO);
    }

    while ('0' == m_bigInt[m_length - 1])
    {
        int temp    = 1 + 9 * rand() / (RAND_MAX + 1);
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
            m_bigInt[m_length - 1] = (char)(temp + ZERO);
        }

        m_bigInt[0] = m_bigInt[0] | 1;

        if (!Rab())
        {
            goto begin;
        }

        for (int index = m_length - 1; index >= 0; --index)
        {
            cout << m_bigInt[index];
        }
        cout << endl;
}

BigInt BigInt::RsaTrans( BigInt& B1, BigInt& N )
{
        BigInt result(1);
        BigInt T;
        T.assign(*this);
        BigInt temp;
        BigInt B;
        B.assign(B1);
        BigInt zero(0);
        BigInt two(2);
        BigInt one(1);

        while (0 < B.compare(zero))
        {
            temp.assign(B.mod(two));
            if (0 == temp.compare(one))
            {
                result.assign(result.multiplication(T));
                result.assign(result.mod(N));
            }
            B.assign(B.division(two));
            T.assign(T.multiplication(T));
            T.assign(T.mod(N));
        }
        return result;
}

void BigInt::toMove( BigInt &num, int length )
{
        BigInt result;
        result.m_length = num.m_length + length;
        int index_result = result.m_length - 1;
        int index_num = num.m_length - 1;
```

优秀毕业论文（设计）样本仅供参考严禁抄袭

```
    int count = 0;
    if (num.m_length < result.m_length)
    {
        count = num.m_length - 1;
    }
    else
    {
        count = result.m_length - 1;
    }

    for (int index = count; index >= 0; --index)
    {
        result.m_bigInt[index_result--] = num.m_bigInt[index_num--];
    }
    num.assign(result);
}

int BigInt::Rab()
{
    int pass = 0;
    BigInt temp1;
    BigInt zero(0);
    for (int index = 0; index < TABLE_SIZE; ++index)
    {
        temp1.assign(mod(BigInt(PrimeTable[index])));
        if (0 == temp1.compare(zero))
        {
            return pass;
        }
    }
    BigInt S, A, I, K;
    BigInt two(2);
    K.assign(*this);
    K.assign(K.subduction(BigInt(1)));
    srand(time(NULL));

    int randnum = rand();
    A.assign(BigInt(randnum));
    S.assign(K);
    while(1 != S.m_length || '0' != S.m_bigInt[0])
    {
        S.assign(S.division(two));
        I.assign(A.RsaTrans(S, *this));
        if(I.compare(K) == 0)
        {
            pass = 1;
            break;
        }
    }
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
    if((I.m_length == 1) && (I.m_bigInt[0] == '1'))
    {
        pass = 1;
    }
    if(pass == 0)
    {
        return pass;
    }

    return pass;
}

BigInt BigInt::Euc(BigInt& A)
{
    BigInt M, E, X, Y, I, J;
    int x, y;
    M.assign(A);
    E.assign(*this);
    X.assign(BigInt(0));
    Y.assign(BigInt(1));
    x = y = 1;
    while((E.m_length != 1)||(E.m_bigInt[0] != '0'))
    {
        I.assign(M.division(E));
        J.assign(M.mod(E));
        M.assign(E);
        E.assign(J);
        J.assign(Y);
        Y.assign(Y.multiplication(I));
        if(x == y)
        {
            if(X.compare(Y) >= 0)
            {
                Y.assign(X.subduction(Y));
            }
            else
            {
                Y.assign(Y.subduction(X));
                y = 0;
            }
        }
        else
        {
            Y.assign(X.addition(Y));
            x = 1 - x;
            y = 1 - y;
        }
        X.assign(J);
    }
    if(x == 0)
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
    {
        X.assign(A.subduction(X));
    }
    return X;
}
void BigInt::strToBigInt( string messages )
{
    int index_bgInt = 0;
    m_length = messages.length() * 3;
    for (int index = 0; index < messages.length(); ++index)
    {
        int pMessage = (int)messages[index];
        m_bigInt[index_bgInt++] = (char)(pMessage / 100 + ZERO);
        m_bigInt[index_bgInt++] = (char)((pMessage % 100) / 10 + ZERO);
        m_bigInt[index_bgInt++] = (char)(pMessage % 10 + ZERO);
    }
}

void BigInt::BigIntToStr( char result[], int count)
{
    int index_bgInt = 0;
    for (int index = 0; index < count; ++index)
    {
        int temp = 0;
        temp = (m_bigInt[index_bgInt++] - ZERO) * 100;
        temp += (m_bigInt[index_bgInt++] - ZERO) * 10;
        temp += m_bigInt[index_bgInt++] - ZERO;
        result[index] = (char)temp;
    }
}

void BigInt::encrypt( BigInt &E1, BigInt &N )
{
    BigInt temp1;
    ofstream fout("enResult");

    int index_r = 0;
    for (int index = 0; index < m_length; ++index)
    {
        BigInt temp(m_bigInt[index] - ZERO);
        temp1.assign(temp.RsaTrans(E1, N));

        for (int index_t = temp1.m_length - 1; index_t >= 0; --index_t)
        {
            fout << temp1.m_bigInt[index_t];
        }
        fout << "\n";
    }
    fout.close();
```

```
}

BigInt BigInt::decrypt(BigInt &E2, BigInt &N)
{
    BigInt result;
    BigInt temp1;
    ifstream fin("enResult");
    ofstream fout("result");
    int index_r = 0;

    int num = 0;
    fin >> num;
    while (!fin.eof())
    {
        BigInt temp(num);
        temp1.assign(temp.RsaTrans(E2, N));

        for (int index_t = temp1.m_length - 1; index_t >= 0; --index_t)
        {
            fout << temp1.m_bigInt[index_t];
        }
        fout << "\n";
        for (int index = 0; index < temp1.m_length; ++index)
        {
            result.m_bigInt[index_r++] = temp1.m_bigInt[index];
        }
        fin >> num;
    }
    fin.close();
    fout.close();
    result.m_length = index_r;
    return result;
}



// RSADemo.cpp : 定义控制台应用程序的入口点。
//
#include "stdafx.h"
#include "BigInt.h"
//#include <Windows.h>

int _tmain(int argc, _TCHAR* argv[])
{
    BigInt P;
    BigInt Q;
    BigInt N;
    BigInt E1(11);
    BigInt E2;
    BigInt MW;
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

```
    BigInt result1;
    BigInt result2;

    P.getPrime(40);
    Q.getPrime(40);

    N.assign(P.multiplication(Q));

    --P.m_bigInt[0];
    --Q.m_bigInt[0];
    E2.assign(E1.Euc(P.multiplication(Q)));

    string str = "&#$*&*&%^$#%@#$%#^&^&@!)(";
    cout << "明文串:";
    cout << str.c_str() << endl;
    cout << endl;
    MW.strToBigInt(str);
    cout << "明文转换为大数: ";
    for (int index = 0; index < MW.m_length; ++index)
    {
        cout << MW.m_bigInt[index];
    }
    cout << endl;
    cout << endl;

    MW.encrypt(E1, N);

    result2.assign(BigInt::decrypt(E2, N));
    cout << "解密所得大数: ";
    for (int index = result2.m_length - 1; index >= 0; --index)
    {
        cout << result2.m_bigInt[index];
    }
    cout << endl;
    cout << endl;

    char temp[20] = {'0'};
    result2.BigIntToStr(temp, str.length());
    cout << "还原得到的字符串: " << temp << endl;
    cout << endl;
    cout << endl;

    int gameOver = 0;
    while (!gameOver)
    {
        cin >> gameOver;
    }

    return 0;
}
```

优秀毕业论文（设计）样本 仅供参考 严禁抄袭

北京师范大学珠海分校本科生毕业论文评定表

信息技术 **学院（系）** 2009 **级** 软件工程 **专业** **姓名** 叶泽鸿 **学号** 0901030140

题目	
指导教师意见	（论文评语及给出初步成绩） 指导教师签章　　201　年　月　日
答辩小组意见	（论文、答辩评语，成绩及是否推荐院级优秀论文） 成绩（百分制）_____ （四级分制）_____； 推荐申报院级优秀论文投票：赞成____人，反对____人，弃权____人。 组长签章　　201　年　月　日
院级评优意见	（是否同意评为院级优秀论文及推荐校级优秀论文） 推荐申报院级优秀论文投票：赞成____人，反对_____人，弃权_____人。 教学院长（主任）签章　　院系章　　201　年　月　日
校级评优意见	 教务处长（签章）　　201　年　月　日

注：（1）此表一式四份，一份存入学校档案，一份装入学生档案、一份装入论文封底，一份学院存档、长期保存。填写时务必字迹工整，签章俱全。（2）*如系两位教师合作指导，应同时签名。

优秀毕业论文（设计）样本 仅供参考 严禁抄袭