

**信息安全概论实验报告**

实验1-3 非对称密码算法 RSA

**学 院**  电子与信息学院

**班 级** 信息工程7班

**学生姓名** 李文剑

**指导教师** 向友君

**提交日期** 2016年 1月7 日

1. 实验目的

通过实际编程了解非对称密码算法RSA的加密和解密过程，加深对非对称密码算法的认识。

1. 实验原理

前面讲的对称密码算法要求通信双方通过交换密钥实现使用同一个密钥，这在密钥管理、发布和安全性方面存在很多问题，而非对称密码算法解决了这个问题。

非对称密码算法是指一个加密系统的加密密钥和解密密钥是不相同，或者说不能从其中一个推导出另一个。在非对称密码算法的两个密钥中，一个是用于加密的密钥，它是可以公开的称为公钥；另一个是用于解密的密钥，是保密的，称为私钥。非对称密码算法解决了对称密码体制中密钥管理的难题，并提供了对信息发送人的身份进行验证的手段，是现代密码学的最重要的发明和进展。

RSA密码体制是目前为止最为成功的非对称密码算法，是在1977年由Rivest、Shamir和Adleman提出的第一个比较完善的非对称密码算法。它的安全性是建立在“大数分解和素性检测”这个数论难题的基础上，即将两个大素数相乘在计算上容易实现，而将该乘积分解为两个大素数因子的计算量相当大。虽然的安全性还未能得到理论证明，但经过20多年的密码分析和攻击，迄今仍然被实践证明是安全的。

RSA算法描述如下：

1. 公钥：

选择两个互异的大素数p和q，n是二者的乘积，即n = pq，使Φ(n)＝（p-1）(q-1)，Φ(n)为欧拉函数。随机选取正整数e，使其满足gcd(e, Φ(n))=1，即e和Φ(n)互质，则将(n,e)作为公钥。

2. 私钥:求出正数d，使其满足e×d=1 modΦ(n)，则将(n,d)作为私钥。

3. 加密算法：对于明文M，由C=Me mod n ，得到密文C。

4．解密算法: 对于密文C，由M=Cd mod n，得到明文M。

如果窃密者获得了n，e和密文C，为了破解密文他必须计算出私钥d，为此需要先分解n为p和q。为了提高破解难度，达到更高的安全性，一般商业应用要求n的长度不小于1024bit,更重要的场合不小于2048bit。

1. 实验环境

运行windows或linux操作系统的PC机，具有gcc（linux）、VC（windows）等C语言编译环境。

1. 实验内容和步骤

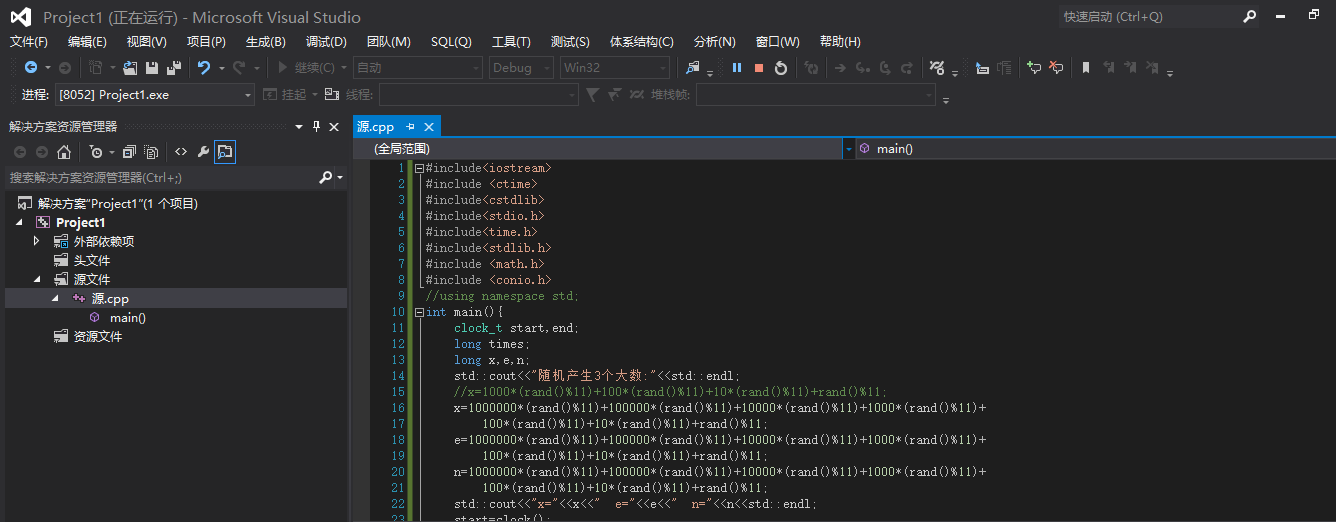
1．编写一个程序，随机选择三个较大的数x，，然后计算xe mod n，记录程序运行时间。实际中应用的素数为512bit，也就为1024bit。这样的大数在计算机上如何表示，如何进行运算，查阅资料给出简单说明。n

2．计算机在生成一个随机数时，并不就是素数，因此要进行素性检测。是否有有确定的方法判定一个大数是素数？查阅资料，找出目前实际可行的素数判定法则，并且比较各自的优缺点。

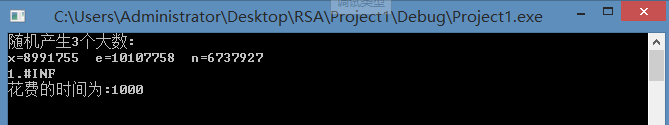
3．光盘附录中给出了一个密码算法库，其中包括各种对称加密算法、非对称加密算法、HASH算法和数字签名算法。找出其中关于RSA的部分，并且基于标准输入输出写一段用RSA加密文件的程序。

1. 实验过程
2. 编写一个程序，随机选择三个较大的数x，，然后计算xe mod n，记录程序运行时间。实际中应用的素数为512bit，也就为1024bit。这样的大数在计算机上如何表示，如何进行运算，查阅资料给出简单说明。

程序如下：



运行结果为：



资料：

1028位，就是64字节，但现在大多数人用的计算机都是32位，远达不到。网上搜索得到下面资料： RSA依赖大数运算，目前主流RSA算法都建立在512位到1024位的大数运算之上，所以我们在现阶段首先需要掌握1024位的大数运算原理。 大多数的编译器只能支持到64位的整数运算，即我们在运算中所使用的整数必须小于等于64位，即：0xffffffffffffffff，也就是18446744073709551615，这远远达不到RSA的需要，于是需要专门建立大数运算库来解决这一问题。

最简单的办法是将大数当作字符串进行处理，也就是将大数用10进制字符数组进行表示，然后模拟人们手工进行“竖式计算”的过程编写其加减乘除函数。但是这样做效率很低，因为1024位的大数其10进制数字个数就有数百个，对于任何一种运算，都需要在两个有数百个元素的数组空间上做多重循环，还需要许多额外的空间存放计算的进位退位标志及中间结果。当然其优点是算法符合人们的日常习惯，易于理解。

另一种思路是将大数当作一个二进制流进行处理，使用各种移位和逻辑操作来进行加减乘除运算，但是这样做代码设计非常复杂，可读性很低，难以理解也难以调试。

于是分析可能是一种介于两者之间的思路：

将大数看作一个n进制数组，对于目前的32位系统而言n可以取值为2的32次方，即0x10000000，假如将一个1024位的大数转化成0x10000000进制，它就变成了32位，而每一位的取值范围就不是0-1或0-9，而是0-0xffffffff。我们正好可以用一个无符号长整数来表示这一数值。所以1024位的大数就是一个有32个元素的unsigned long数组。而且0x100000000进制的数组排列与2进制流对于计算机来说，实际上是一回事，但是我们完全

可以针对unsigned long数组进行“竖式计算”，而循环规模被降低到了32次之内，并且算法很容易理解。

例如大数18446744073709551615，等于“ffffffff ffffffff”，它就相当于10进制的“99”：有两位，每位都是ffffffff。而大数18446744073709551616，等于“00000001 0000000000000000”，它就相当于10进制的“100”：有三位，第一位是1，其它两位是0。如果我们要计算18446744073709551616-18446744073709551615，就类似于100-99：

1. 找出目前实际可行的素数判定法则，并且比较各自的优缺点。

所谓素数，是指除了能被1和它本身整除而不能被其他任何数整除的数。根据素数的定义，只需用2到N-1去除N，如果都除不尽则N是素数，结束知其循环。由此得算法1。

（1） flay=0,i=2. /\*flay为标志，其初值为0，只要有一个数除尽，其值变为1.

（2） If n mod i=0 then flay=l else i=i+1/\* n mod i是n除以i的余数.

（3） If flay=0 and I<=n-1 then(2) else go (4)

（4） If flay=0 then write“n是素数。”else write“不是素数”

最坏的情形下，即N是素数时，算法1需要执行N-2次除法，时间复杂性太大。

假设N桶分解成iXj(i,j是小于N的整数)，则必存在一个因子（1<=i<=int(√n)）,这样只需用2到int(√n)去除N即可，于是循环次数可以大减小，由此得出算法2

算法2（改进算法）

（1） flag=0,i=2

（2） if n mod i then flag=1else i=i+1

（3） if flag=0 and i<=int(√n) then go(2) else go(4)

（4） if flah=0 then write”n是素数”else write “n不是素数“。

最坏的情形下，即当N是纱数时1需要执行int(√n)-1次除法。

虽然算法2比算法1确是快了不小，但有重复计算，如果用2去除N时若不尽则用2的倍数去除N也除不尽，于是只要2除不尽，2的倍数就不用去除，这样可以减少除法次数，由此得出算法3（1）for(i=2;int(√n);i++)mark[i]=0/\*mark是标记其初值为0，只要它的因子除不尽其值变为1。

（2）i=2,flag=0

(3)while(flag=0and i<=int(√n)

{If mark[i]=0

Then

{ If n mod i=0

Then flag=1

Else

S=i+i

While s<int(√n)

{Mark[s]=1

S=s+i}

}

}i=i+1

}

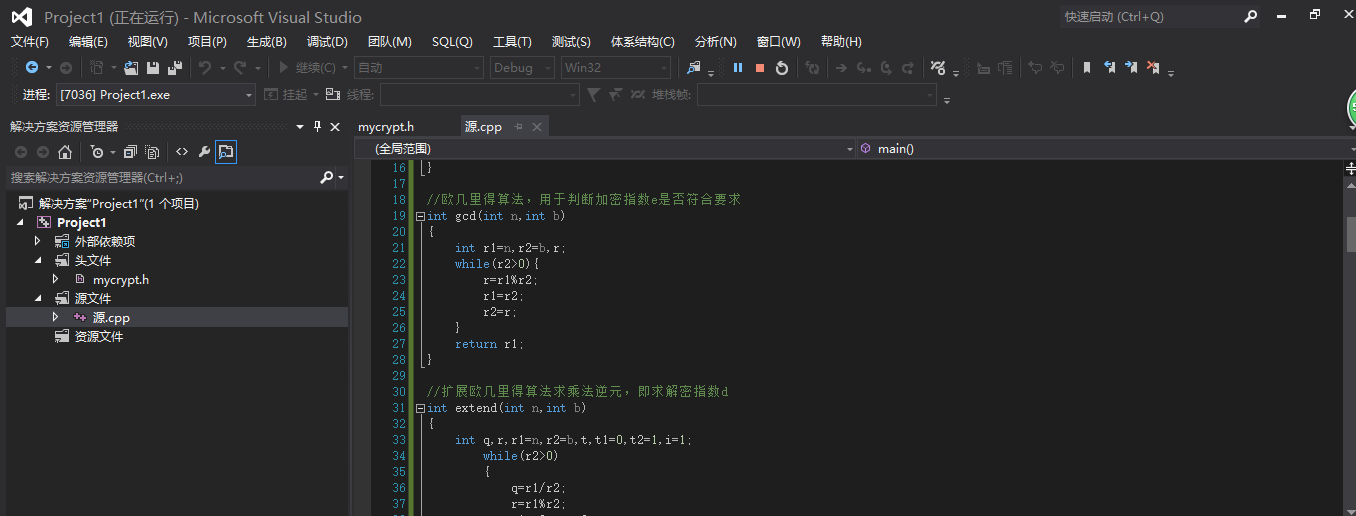
(4)if flag=0 then write”n是素数。Else write “N不是素数。“

该算法迭代重复次数仍为int(√n)次，但是筛法算法在执行过程中，每次都筛去许多数，使下一次迭代时不用再除法而只做比较，比较比除法运算要简单得多，因此实际执行速度比算法1，算法2要得多。

当然，对于素数的判定还有别的方法，如用概率算法求素数，但其理论太难，一般情况下很少用。关于素数的判定问题还有待发现简单更高效的方法。

1. 找出其中关于RSA的部分，并且基于标准输入输出写一段用RSA加密文件的程序。

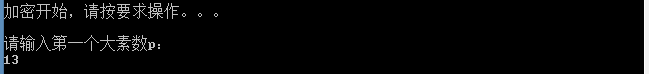
界面如下：



步骤一：输入明文



步骤二：输入第一个素数



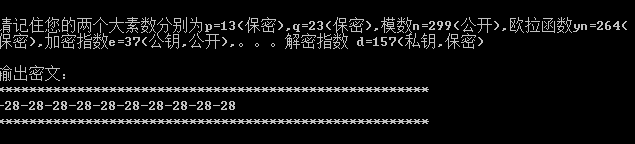
步骤三：输入第二个素数



步骤四：输入加密指数



步骤五：得到密文



步骤六：输入密钥157得到明文



1. 结论与心得

对非对称密码算法RSA的理解加深了，会运用一些现成的算法进行编程，对一些比较复杂的算法开始基本认识并深刻的掌握。在以后所涉及这方面的知识将会有全新的了解和掌握。

代码附录：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

//将十进制数转换成二进制，用于检验大素数p和q

int zhuan\_huan(int b,int a[],int k)

{ int t,temp=-1;

while(b>0){

t=b%2;

temp++;

a[temp]=t;

b=b/2;

}

return temp;

}

//欧几里得算法，用于判断加密指数e是否符合要求

int gcd(int n,int b)

{

int r1=n,r2=b,r;

while(r2>0){

r=r1%r2;

r1=r2;

r2=r;

}

return r1;

}

//扩展欧几里得算法求乘法逆元，即求解密指数d

int extend(int n,int b)

{

int q,r,r1=n,r2=b,t,t1=0,t2=1,i=1;

while(r2>0)

{

q=r1/r2;

r=r1%r2;

r1=r2; r2=r;

t=t1-q\*t2;

t1=t2;

t2=t;

}

if(t1>=0) return t1%n;

else{

while((t1+i\*n)<0)

i++;

return t1+i\*n;

}

}

//检验大素数，符合要求返回1，否则返回0

int Witness(int a,int n)

{

int d=1,k,r=n-1,i,x,b[1000];

k=zhuan\_huan(r,b,1000);

for(i=k;i>=0;i--){

x=d;

d=(d\*d)%n;

if((d==1)&&(x!=1)&&(x!=n-1)) return 0;

if(b[i]==1) d=(d\*a)%n;

}

if(d!=1) return 0;

else return 1;

}

//快速计算模指数

int js\_mod(int a,int b,int n)

{

int x=0,y=1,k,i,s[1000];

k=zhuan\_huan(b,s,1000);

for(i=k;i>=0;i--){

x=2\*x;

y=(y\*y)%n;

if(s[i]==1){

x++;

y=(y\*a)%n;

}

}

return y;

}

//主函数。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

void main()

{

int p,q,e,d,n,yn,m[1000],c[10000];//c[10000]存放加密后的数字密文，m[1000]存放解密后的数字明文，即英文明文在zimu\_biao[69]中的下标。

int i,j; //i,j用于循环遍历数组

int mi\_yue; //用户输入的密钥

int count=1; //统计输入密钥的次数，count>3时将不允许用户再输入。

char min\_wen[1000],re\_min\_wen[1000];//分别为用户输入的明文、密文，解密后的明文。

//密钥生成

char zimu\_biao[69]="abcdefghijklmnopqrstuvwxyz,ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789'.?!";

printf("请输入您要发送的明文文件（小写英文表示）:\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

gets(min\_wen);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n加密开始，请按要求操作。。。\n\n");

printf("请输入第一个大素数p：\n");

while(1){

scanf("%d",&p);

if(Witness(2,p)==1){

printf("您输入的第一个大素数 %d 符合要求\n",p);

break;

}

else

printf("您输入的 %d 不是素数,请重新输入：\n",p);

}

printf("请输入第二个大素数q：\n");

while(1){

scanf("%d",&q);

if(Witness(2,q)){

printf("您输入的第二个大素数 %d 符合要求\n",q);

break;

}

else

printf("您输入的 %d 不是素数,请重新输入：\n",q);

}

n=p\*q; yn=(p-1)\*(q-1);

printf("请输入加密指数（整数）e,且0<e<%d\n",yn);//下面由用户设定加密指数

while(1){

scanf("%d",&e);

if(gcd(yn,e)==1){

printf("您输入加密指数 %d 与 %d 互素,符合要求\n",e,yn);

break;

}

else

printf("您输入加密指数 %d 与 %d 不互素,请重新输入。。。\n",e,yn);

}

d=extend(yn,e); //求解密指数d

printf("\n\n请记住您的两个大素数分别为p=%d(保密),q=%d(保密),模数n=%d(公开),欧拉函数yn=%d(保密),加密指数e=%d(公钥,公开),。。。解密指数 d=%d(私钥,保密)\n\n",p,q,n,yn,e,d);

//明文转换过程

/\* scanf("%s",min\_wen);

printf("%s",min\_wen); \*/

for(i=0;i<strlen(min\_wen);i++)

for(j=0;j<68;j++) //for(j=0;j<26;j++)

if(min\_wen[i]==zimu\_biao[j])

m[i]=j;//将字符串明文换成数字，并存到整型数组m里面，即明文的另一种表示方法

//加密过程

for(i=0;i<strlen(min\_wen);i++)

c[i]=js\_mod(m[i],e,n);

printf("输出密文：\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

for(i=0;i<strlen(min\_wen);i++)

printf("%d",c[i]);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

//解密过程

for(i=0;i<strlen(min\_wen);i++)

m[i]=js\_mod(c[i],d,n);

for(i=0;i<strlen(min\_wen);i++)

re\_min\_wen[i]=zimu\_biao[m[i]];

//提示用户解密

printf("\n\n您有3次输入密钥的机会，密钥正确后将进行解密显示明文，3次输入错误解密将终止，请注意。。。\n\n");

while(1){

scanf("%d",&mi\_yue);

if(mi\_yue==d){

printf("密钥输入正确，您得到的明文为：\n\n");

for(i=0;i<strlen(min\_wen);i++)

printf("%c",re\_min\_wen[i]);

printf("\n\n");

break;

}

else{

printf("您第%d次输入的密钥错误，请重新输入。。。\n",count);

count++;

if(count>3){

printf("\n您已%d次输入的密钥错误，将不允许继续输入\n",count-1);

break;

}

}

}

}