

**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3班

**学生姓名** 蒲 尧

**指导教师** 向友君

**提交日期** 2019年 9月23 日

# 实验 1-3 非对称密码算法 RSA

## 一、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。

## 二、实验原理

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=M^e mod n ，得到密文C。

### 4.解密算法

对于密文C，由M=C^d 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。

## 三、实验环境

系统：Windows 10家庭中文版

IDE：Code::Blocks 17.12

Compiler：GNU GCC Compiler

## 四、实验内容及步骤

### 1.题目

1.为了加深对 RSA 算法，根据已知参数：p=3，q=11，M＝2，手工计算公私钥，并对明文进行加密，然后对密文进行解密。

2.光盘中给出了一个可以进行 RSA 加密和解密的对话框程序 RSATool，运行这个程序加密一段文字，了解 RSA 算法原理。尝试着加密一大段文字，记录程序的运行时间。使用DES 算法加密相同的文字，比较两种算法加密的速度。

### 2.解答

#### 1）利用RSA算法手动加密解密

两个素数是

**p=3,q=11**

**n=p×q=33, Φ(n)＝（p-1）(q-1)=20**

我从[0，Φ(n)-1]中选择一个与20互素的数e=3

e×d=1 modΦ(n)

算出：**e=3,d=7**

公开密钥 **Pk = {e,n} = {3,33}**

私密密钥 **Pk = {d,n} = {7,33}**

要加密的小于n=33正整数，即加密前的明文**M=2**

密文为：**C=M^e mod n = 8**

解密后的明文为：**M=C^d mod n = 2**

#### 2）比较两种算法加密的速度。

由于没有光盘中的程序，于是参考网上，我自己写了简单RSA加密解密的程序：

代码地址：

<https://github.com/Allenem/introduction-of-information-security/blob/master/RSA/main.cpp>

运行截图如下



## 五、实验总结

### 总结

通过本次实验，我了解了RSA算法的详细过程，感觉比DES算法简单，更加安全。但是大素数的判断，较大素数的选择，两数互质的判断是难点。下面是查找相关资料的结果：

### 回答两个问题

#### 问题一

编写一个程序，随机选择三个较大的数 x ，e ，n， 然后计算x^e 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。

##### x^e mod n代码如下

#include<iostream>

using namespace std;

//模幂函数

//返回X^YmodN

long PowMod(long x,long y,long n)

{

long s,t,u;

s=1;

t=x;

u=y;

while(u)

{

if(u&1)

s=(s\*t)%n;

u>>=1;

t=(t\*t)%n;

}

return s;

}

int main(){

unsigned long x,e,n,m;

cout<<"请输入3个数字x,e,n"<<endl;

cin>>x>>e>>n;

m = PowMod(x,e,n);

cout<<"m="<<m<<endl;

}

##### 大数可以这么表示

1000:1e3

#### 问题二

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。

##### 简单质数判断

#include <stdio.h>

int main()

{

int i,n;

scanf("%d",&n);

for(i=2;i\*i<=n;i++)

if(n%i==0)break;

printf("%s\n",n%i==0?"No":"Yes");

return 0;

}

##### 简单互质判断

int gcd(int x,int y)

{

int t;

while (y) t=x, x=y, y=t%y;

return x; // 最小公因数

}

##### 以下是查找的“判断大数是否为素数”的引用

###### 引用1

Rabin -Miller算法是典型的验证一个数字是否为素数的方法。判断素数的方法是Rabin-Miller概率测试，那么他具体的流程是什么呢。假设我们要判断n是不是素数，首先我们必须保证n 是个奇数，那么我们就可以把n 表示为 n = (2^r)\*s+1,注意s 也必须是一个奇数。然后我们就要选择一个随机的整数a (1<=a<=n-1)，接下来我们就是要判断 a^s=1 (mod n) 或a^((2^j)\*s)= -1（mod n）(0<=j如果任意一式成立，我们就说n通过了测试，但是有可能不是素数也能通过测试。所以我们通常要做多次这样的测试，以确保我们得到的是一个素数。（DDS的标准是要经过50次测试）

采用Rabin-Miller算法进行验算

首先选择一个代测的随机数p，计算b，b是2整除p-1的次数。然后计算m，使得n=1+(2^b)m。  
（1） 选择一个小于p的随机数a。  
（2） 设j=0且z=a^m mod p  
（3） 如果z=1或z=p-1，那麽p通过测试，可能使素数  
（4） 如果j>0且z=1, 那麽p不是素数  
（5） 设j=j+1。如果j且z<>p-1,设z=z^2 mod p，然后回到(4)。如果z=p-1，那麽p通过测试，可能为素数。  
（6） 如果j=b 且z<>p-1，不是素数

###### 引用2

Miller-Rabin算法是目前主流的基于概率的素数测试算法，在构建密码安全体系中占有重要的地位。通过比较各种素数测试算法和对Miller-Rabin算法进行的仔细研究，证明在计算机中构建密码安全体系时， Miller-Rain算法是完成素数测试的最佳选择。通过对Miller-Rabin 算 法底层运算的优化，可以取得较以往实现更好的性能。Miller-Rabin算法是Fermat算法的一个变形改进，它的理论基础是由Fermat定理引申而来。

Fermat 定理： n是一个奇素数，a是任何整数(1≤ a≤n-1) ，则 a^(n-1)≡1(mod n)。

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\*r的形式(r是奇 数)，a 是和n互素的任何整数， 那么ar≡1(mod n) 或者对某个j(0≤j ≤s －1， j∈Z) 等式 a2jr ≡－1(mod 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x2 ≡ 1 mod n只有±1两个解。

##### 下面是判断大数是否为素数的代码

<https://github.com/Allenem/introduction-of-information-security/blob/master/judgePrimeNumber/main.cpp>