# 信息安全作业一：RSA算法实现

## RSA算法原理

RSA加密算法是一种非对称加密算法，所谓非对称，就是指该算法加密和解密使用不同的密钥，即使用加密密钥进行加密、解密密钥进行解密。在RAS算法中，加密密钥（即公开密钥）PK是公开信息，而解密密钥（即秘密密钥）SK是需要保密的。加密算法E和解密算法D也都是公开的。虽然解密密钥SK是由公开密钥PK决定的，由于无法计算出大数n的欧拉函数phi(N)，所以不能根据PK计算出SK。

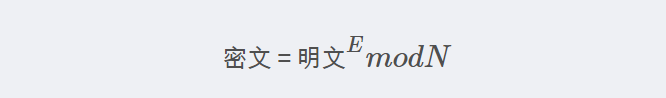
也就是说，对极大整数做因数分解的难度决定了RSA算法的可靠性。理论上，只要其钥匙的长度n足够长，用RSA加密的信息实际上是不能被解破的。

RSA算法通常是先生成一对RSA密钥，其中之一是保密密钥，由用户保存；另一个为公开密钥，可对外公开。为提高保密强度，RSA密钥至少为500位长，一般推荐使用1024位。这就使加密的计算量很大。为减少计算量，在传送信息时，常采用传统加密方法与公开密钥加密方法相结合的方式，即信息采用改进的DES或IDEA密钥加密，然后使用RSA密钥加密对话密钥和信息摘要。对方收到信息后，用不同的密钥解密并可核对信息摘要。

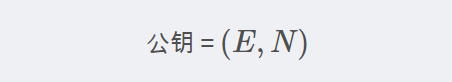
## RSA过程

### RSA加密过程

RSA的加密过程可以使用一个通式来表达：

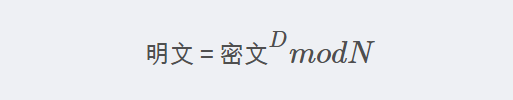


也就是说RSA加密是对明文的E次方后除以N后求余数的过程。从通式可知，只要知道E和N任何人都可以进行RSA加密了，所以说E、N是RSA加密的密钥，也就是说E和N的组合就是公钥，我们用(E,N)来表示公钥：

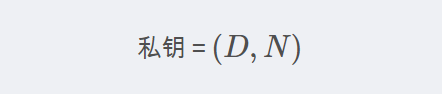


### RSA解密过程

RSA的解密同样可以使用一个通式来表达：



也就是说对密文进行D次方后除以N的余数就是明文，这就是RSA解密过程。知道D和N就能进行解密密文了，所以D和N的组合就是私钥：



从上述可以看出RSA的加密方式和解密方式是相同的，加密是求“E次方的mod N”;解密是求“D次方的mod N”。此处D是解密（Decryption）的首字母；N是数字（Number）的首字母。

小结：

|  |  |
| --- | --- |
| 公钥 | （E，N） |
| 私钥 | （D，N） |
| 密钥对 | （E，D，N） |
| 加密 | 密文＝明文EmodN密文＝明文EmodN |
| 解密 | 明文＝密文DmodN明文＝密文DmodN |

### 密钥生成

既然公钥是（E，N），私钥是（D，N），所以密钥对即为（E，D，N），但密钥对是怎样生成的？步骤如下：

* 求N
* 求L（L为中间过程的中间数）
* 求E
* 求D

#### 3.1 求N：

准备两个质数p，q。这两个数不能太小，太小则会容易破解，p、q之积就是N。如果互质数p和q足够大，那么根据目前的计算机技术和其他工具，至今也没能从N分解出p和q。换句话说，只要密钥长度N足够大（一般1024足矣），基本上不可能从公钥信息推出私钥信息。

#### 3.2 求L：

L 是(p－1)和(q－1)的最小公倍数，可用如下表达式表示

#### 3.3 求E：

E必须满足两个条件：1.E是一个比1大比L小的数;2.E和L的最大公约数为1；

用gcd(X,Y)来表示X，Y的最大公约数则E条件如下：

之所以需要E和L的最大公约数为1，是为了保证一定存在解密时需要使用的数D。现在我们已经求出了E和N，也就是说我们已经生成了密钥对中的公钥了。

#### 3.4 求D：

数D是由数E计算出来的，D是E关于L的乘法逆元。数D必须保证足够大。D、E和L之间必须满足以下关系：

只要D满足上述2个条件，则通过E和N进行加密的密文就可以用D和N进行解密。简单地说条件2是为了保证密文解密后的数据就是明文。

现在私钥也已经生成了，密钥对也就自然生成了。

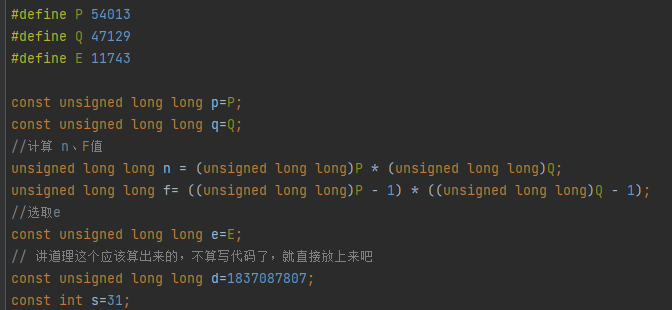
#### 3.5小结：

1. 求N＝p＊q ；p，q为质数
2. 求L＝lcm（p－1，q－1）；L为p－1、q－1的最小公倍数
3. 求E 1 < E < L，gcd（E，L）=1；E，L最大公约数为1（E和L互质）
4. 求D 1 < D < L，E＊D mod L＝1

## RSA程序实现

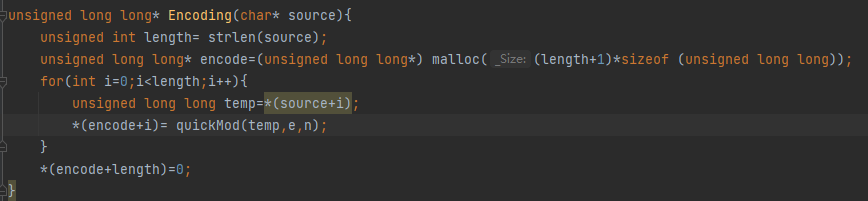
### 参数设置部分：

此部分用于设置RSA密钥的各种参数，设置方法如同上文所叙。



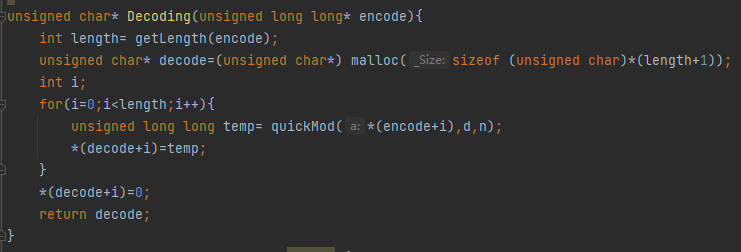
一般而言，乘法逆元D应该是通过计算得到，但是这里为了节省时间就直接使用指导书中的例子了。

### 加密部分：



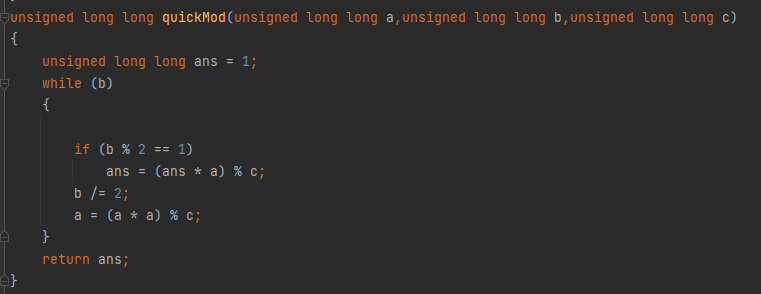
对传入的字符串进行加密运算。

### 解密运算：



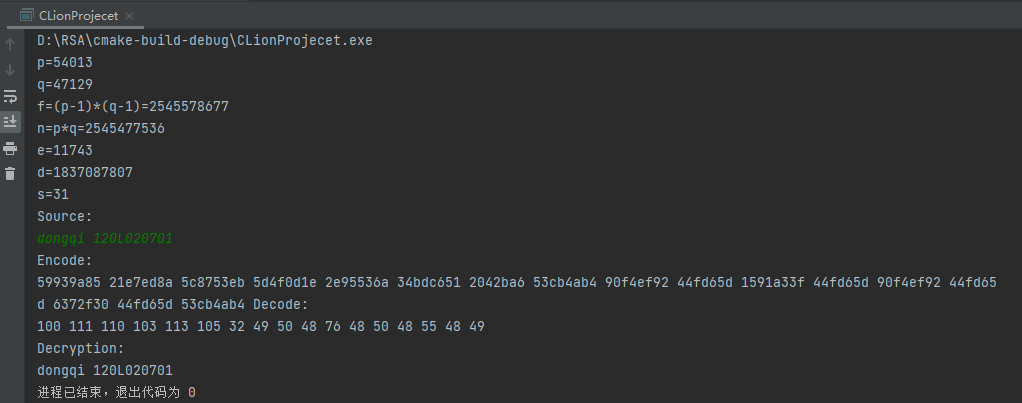
对传入的字符串进行解密运算。

### 快速幂取余：



用于加快幂取余运算，在decode和encode俩个函数中复用。

### 程序演示：



得到期望结果。