**用Mathematica实现RSA密码体制**

**摘要：**

本文主要研究mathematica在RSA公钥密码体制中的应用，先讲述基本背景与实现步骤，随后将给出思路分析以及几个核心算法的mathematica代码，并举出实例加以验证与说明。最后给出仍存在的问题，安全性等几个相关问题的思考。文章结构十分简单：按照密码体制的步骤来讲述相应的mathematica实现思路与代码。这些操作包括：在mathematica环境下实现生成大素数，元素求逆，平方-乘等系列算法，并由此生成公钥和私钥。

**关键词：**

Mathematica 密码学 公钥密码体制 RSA 素性检测 Solovay-Strassen算法 **Zn**元素求逆 平方乘算法

**正文：**

1. RSA公钥密码体制简介

密码学的基本目的是使得两个在不安全信道中通信的人，通常称为Alice和Bob，以一种使他们的敌手Oscar不能明白和理解通信内容的方式进行通信。在经典密码学模型中，Alice和Bob秘密地选择密钥K，根据K得到加密规则e(K)和解密规则d(K)。d(K)可以从e(K)容易地导出，由于d(K)或e(K)的泄漏会导致系统的不安全。

由此我们引入公钥密码体制——使得由e(K)来求d(K)是计算上不可行的。加密规则e(K)是一个可以被公布的公钥。Alice利用公钥发一条加密的信息给Bob（无须共享秘密密钥的通信），Bob是唯一能够用私钥对密文进行解密的人。

我们将讨论的一个著名的公钥密码体制——RSA密码体制，于1977年由Rivest，Shamir和Adleman发明，其安全性基于分解大整数的困难性。

其定义**加密解密**操作如下：

n=pq，其中p，q为素数，在Zn中，ab modΦ(n)=1，

**加密函数：eK(x)=x^b mod n；**

**解密函数：dK(y)=y^a mod n；**x，y属于Zn。

值n和b组成公钥，值p，q，a组成私钥。

RSA密码体制实现步骤如下：

**Step1**：**生成两个大素数p,q，p≠q;**

**Step2: n=pq,且 Φ(n)=(p-1)(q-1); Φ(n)是小于n且与n互素的正整数个数)**

**Step3: 选择一个随机数b（1<b<Φ（n）），使得gcd(b, Φ(n))=1;**

**Step4: a=b^(-1) mod Φ(n)**

**Step5: 公钥为（n，b），私钥为（p，q，a）**

1. **用mathematica实现的思路**

下面针对以上算法的每一步给出用mathematica编程的思路。

首先考察如何快速随机生成两个大素数。

先介绍一般方法（非mathematica环境下）：

用Random函数取某个区间的随机数，若最低位为偶数, 则将它加1, 以确保该素数为奇数, 从而保证了平均节省一半的运算时间。检查以确保p不能被任何小素数整除, 如3, 5, 7, 11 等等目的是排除p 是合数的绝大部分可能性, 减少了下面步骤对p 进行素数测试的总次数, 从而大大节省了运算时间。接下来使Solovay-Strassen算法检测其是否为素数，最后返回两个大素数p和q。

Solovay-Strassen算法是一个偏是的Monte Carlo算法，并具有1/2的错误率。其意思就是：若程序返回合数（” composite”），则一定是正确的， 若返回素数（” prime”），则有至多1/2的概率它是一个合数。但是通过多次运算，错误概率可以降到任何所期望的值以下。

Solovay-Strassen算法是基于判断奇整数的Jacobi符号的算法。而计算Jacobi符号可以利用mathematica的内置函数JacobiSymbol很方便地直接实现。

其实在mathematica中，可以直接用PrimeQ函数判断一个数是否为素数，所以给定位数的随机数，可以经PrimeQ判别后直接输出两个大素数p，q。计算得知其速度比方法一更快。

选择一个随机数b(1<b<Φ(n))，使得gcd(b, Φ(n))=1，实现这个目的比较容易，只需要用Random函数取{1，Φ(n)}，中的随机数，检测其是否与Φ(n)互素，若是，则输出即可。

在上述步骤的Step4: “ a=b^(-1) mod Φ(n)” 中，要求整数b在模Φ(n)意义下的逆，这里要借助一些数论与抽象代数的知识，简单介绍如下：

在一个整环中求乘法逆元的思路基于Euclidean算法以及由它导出的扩展Euclidean算法：以两个整数作为输入，计算出整数r，s和t使得r=gcd（a，b）且sa+tb=r。扩展Euclidean算法立即得出数值b^-1 mod a（如果存在），因为乘法逆b^-1 mod a=t mod a，详细的证明不在本文给出，可阅读参考文献。

现在考虑模指数，即计算形如x^c mod n的函数。在RSA密码体制里，加密解密都要用到模指数，上述计算x^c mod n可以通过c-1次模乘来实现。然而如果c非常大，其效率会很低下。著名的平方-乘算法可以把计算x^c mod n所需模乘次数降低为最多2l次，l是c的二进制表示的比特数。于是x^c mod n可以在时间O（lk^2）内算出。

1. 几个核心算法及其代码

**1.生成大素数**

**方法一（任何环境都适用）：**

其中Solovay-Strassen算法语句,以及素性检测生成j位大素数的程序如下：

**PrimeTest1[n\_] := Module[{a, x, y},**

**a = Random[Integer, {1, n - 1}];**

**x = JacobiSymbol[n, a];**

**If[x == 0, Print["composite"]];**

**y = Mod[a^((n - 1)/2), n];**

**If[x != 0 && (Mod[x, n]) == (Mod[y, n]), Print["prime"], Print["composite"]]**

**]**

**MakePrime[n2\_,j\_] := Module[{d1, d2, k, x, a1, a2, i, p},**

**d1 = False;**

**d2 = False;**

**a2 = 0;**

**While[d2 == False,**

**x = Random[Integer, {10^j, 10^(j+1)}];**

**If[GCD[x, 2] == 2, x = x + 1];**

**While[d1 == False,**

**For[a1 = 0; k = 1, k <= 25, k++;**

**If[GCD[Prime[k], x] != 1, a1++]]**

**If[a1 == 0, Break[], d1 = False]]**

**For[p = 1; i = 1, i <= n2, i++;**

**If[PrimeTest1[x] == "composite", Break[], p++]**

**];**

**If[p == n2, Return[x], d2 = False]**

**]**

**Return[z]**

**]**

方法二（mathematica环境下适用）：

**Rub[n\_] := Module[{b, d},**

**d = 0;**

**While[d == 0,**

**b = Random[Integer, {10^n, 10^(n + 1)}];**

**If[PrimeQ[b] == True, Return[b]];**

**]]**

p=Rub[20]

q=Rub[20]

3.生成随机数b与互素

**Rub[n\_] := Module[{b, d},**

**d = 0;**

**While[d == 0,**

**b = Random[Integer, {2, n}];**

**If[GCD[b, n] == 1, Return[b]]**

**]]**

4.求**a∈Zn**在**Zn**下的逆

**InverseZn[n\_, a\_] := Module[{p, q, s, r, t1, t, temp},**

**p = n;**

**q = a;**

**t1 = 0;**

**t = 1;**

**s = Floor[p/q];**

**r = p - q\*s;**

**While[r > 0,**

**temp = Mod[t1 - s\*t, n];**

**t1 = t;**

**t = temp;**

**p = q;**

**q = r;**

**s = Floor[p/q];**

**r = p - q\*s;**

**]**

**If[q != 1, Print["没有逆元"]];**

**Return[t]**

**]**

5.平方-乘算法

**SquareMultiply[x1\_, c\_Integer, n0\_] := Module[{z, i, L, t = {}},**

**t = IntegerDigits[c, 2];**

**L = Length[t];**

**For[z = 1; i = 1, i <= L, i++;**

**z = Mod[z^2, n0];**

**If[c[[i]] == 1, z = Mod[z\*x1, n0]]**

**];**

**Return[z]**

**]**

1. 例子及其密码学意义解释

为了说明的简洁性，生成两个位数不太大的素数。即取j=5，得到p=3677，q=2671 。则n= ，Φ(n)=。生成得b=178331 。用求逆算法知：a=b^-1=2610491 。

至此我们得到公钥（n，b）=（9821267，178331），和私钥（p，q，a）=（3677，2671，2610491）。

假设Alice要传送消息（密文）GOHOME(Go home )，根据字母A～Z与数字0～25之间的一一对应，其对应数字为x=614714124，对其施加以上操作，用平方乘算法计算y=x^b mod n，得到y=，对应字母为DADEWD。即Alice用公开密钥（9821267，178331）加密明文x=614714124（GOHOME），得到密文y=3034223（DADEWD）并发送给Bob，Bob用私钥运行解密算法 x=y^a mod n得到明文x,并得知Alice的信息（Go home）。

1. 总结以及相关问题的思考

不难看出，基于mathematica的丰富功能与强大计算能力，运行RSA密码体制十分方便，下面讨论几个相关的问题。

首先，关于RSA的安全性，攻击RSA密码体制最明显的方式就是试图分解公开模数n，这方面的算法有：Pollard于1974年提出的p-1算法，Dixon的随机平方算法，还有在实际中最常用的二次筛法。另外，攻击者也可以试图得到Φ(n)。只要模数n不是足够大，这些攻击方法也可以用mathematica来实现。具体可以参考文献。

**参考文献**

①密码学原理与实践（第三版），[加] Douglas R.Stinson 著 冯登国 等译，电子工业出版社

②《数据加密算法与大素数的生成及运算》刘少涛, 凌捷著，广东工业大学学报

第18 卷第4 期2001 年12 月

③《抽象代数基本教程》（第七版），John B.Fraleigh