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：

一． 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：$e_K(x)$=x^b mod n；

解密函数：$d_K(y)$=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）

二． 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。

三． 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]
  ]
```

四. 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）。

五. 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