**RSA算法**

**一、基础数论**  
  
1、**互质关系**  
  
如果两个正整数，除了1以外，没有其他公因子，我们就称这两个数是互质关系（coprime）。比如，15和32没有公因子，所以它们是互质关系。这说明，不是质数也可以构成互质关系。  
  
2、**欧拉函数**  
  
定义：任意给定正整数n，请问在**小于等于n的正整数之中**，**有多少个与n构成互质关系**？（比如，在1到8之中，有多少个数与8构成互质关系？），计算这个值的方法就叫做**欧拉函数，以φ(n)**表示。  
  
欧拉函数求法及性质：  
  
1.对于素数p, φ(p)=p-1，**对于对两个素数p,q， //φ（pq）=pq-1**，

欧拉函数是积性函数,但不是完全积性函数.  
  
2.对于一个正整数N的素数幂分解N=P1^q1\*P2^q2\*...\*Pn^qn，则

φ(N)=N\*(1-1/P1)\*(1-1/P2)\*…\*(1-1/Pn).  
  
3.**除了N=2,φ(N)都是偶数**.  
  
4.如果n**可以分解成两个互质的整数之积**，n = p1 × p2，

则φ(n) = φ(p1p2) = φ(p1)φ(p2)

**二、RSA加密**  
  
第一步，随机选择两个不相等的质数p和q。  
  
A选择了61和53。（**实际应用中，这两个质数越大，就越难破解**。）  
  
第二步，计算p和q的乘积n。  
  
A把61和53相乘。  
  
　　n = 61×53 = 3233  
  
**n的长度就是密钥长度**。3233**写成二进制**是110010100001，

一共有12位，所以这个密钥就是12位。

实际应用中，RSA密钥一般是1024位，重要场合则为2048位。  
  
第三步，计算n的欧拉函数φ(n)。  
  
根据公式：  
  
　　φ(n) = (p-1)(q-1)  
  
 A算出φ(3233)等于60×52，即3120。  
  
第四步，**随机选择一个整数e，条件是1< e < φ(n)，且e与φ(n) 互质**。  
  
A就在1到3120之间，随机选择了17。（//实际应用中，常常选择65537。）  
  
第五步，计算e对于φ(n)的模反元素d。  
  
所谓"模反元素"**就是指有一个整数d，可以使得ed被φ(n)除的余数为1**。  
  
　　ed ≡ 1 (mod φ(n))  
  
这个式子**等价于**

ed - 1 = k\*φ(n)  
于是，找到模反元素d，实质上就是对下面这个**二元一次方程求解**。

ex + φ(n)\*k = 1  
  
已知 e=17, φ(n)=3120，  
  
　　17x + 3120y = 1  
  
这个方程可以用"**扩展欧几里得算法**"求解，此处省略具体过程。

总之，A算出一组整数解为 (x,y)=(2753,-15)，即 d=2753。  
  
至此所有计算完成。  
  
第六步，**将n和e封装成公钥，n和d封装成私钥**。  
  
在A的例子中，n=3233，e=17，d=2753，

所以公钥就是 (3233,17)，私钥就是（3233, 2753）。  
  
实际应用中，公钥和私钥的数据都采用ASN.1格式表达（实例）。

**RSA算法的可靠性**  
  
回顾上面的密钥生成步骤，一共出现六个数字：  
  
　　p  
　　q  
　　n  
　　φ(n)  
　　e  
　　d  
  
这六个数字之中，公钥用到了两个（n和e），其余四个数字都是不公开的。其中最关键的是d，因为n和d组成了私钥，一旦d泄漏，就等于私钥泄漏。

那么，有无可能在已知n和e的情况下，推导出d？

（1）ed≡1 (mod φ(n))。只有知道e和φ(n)，才能算出d。  
（2）φ(n)=(p-1)(q-1)。只有知道p和q，才能算出φ(n)。  
（3）n=pq。只有将n因数分解，才能算出p和q。  
  
结论：**如果n可以被因数分解，d就可以算出**，也就意味着私钥被破解。  
  
可是，**大整数的因数分解，是一件非常困难的事情**。目前，除了暴力破解，还没有发现别的有效方法。维基百科这样写道：  
  
　　"对极大整数做因数分解的难度决定了RSA算法的可靠性。换言之，**对一极大整数做因数分解愈困难，RSA算法愈可靠**。  
  
　　假如有人找到一种快速因数分解的算法，那么RSA的可靠性就会极度下降。但找到这样的算法的可能性是非常小的。今天只有短的RSA密钥才可能被暴力破解。到2008年为止，世界上还没有任何可靠的攻击RSA算法的方式。  
  
　　只要密钥长度足够长，用RSA加密的信息实际上是不能被解破的。"  
  
举例来说，你可以对3233进行因数分解（61×53），但是你没法对下面这个整数进行因数分解。  
  
　　12301866845301177551304949  
　　58384962720772853569595334  
　　79219732245215172640050726  
　　36575187452021997864693899  
　　56474942774063845925192557  
　　32630345373154826850791702  
　　61221429134616704292143116  
　　02221240479274737794080665  
　　351419597459856902143413

它等于这样两个质数的乘积：  
  
　　33478071698956898786044169  
　　84821269081770479498371376  
　　85689124313889828837938780  
　　02287614711652531743087737  
　　814467999489  
　　　　×  
　　36746043666799590428244633  
　　79962795263227915816434308  
　　76426760322838157396665112  
　　79233373417143396810270092  
　　798736308917  
  
事实上，这大概是人类已经分解的最大整数（232个十进制位，768个二进制位）。比它更大的因数分解，还没有被报道过，因此目前被破解的最长RSA密钥就是768位。  
  
**加密和解密**  
  
**有了公钥和密钥，就能进行加密和解密了**。  
  
（1）**加密要用公钥 (n,e)**  
  
假设B要向A发送加密信息m，他就要**用A的公钥 (n,e) 对m进行加密**。

这里需要注意，**m必须是整数（字符串可以取ascii值或unicode值），**

且**m必须小于n**。  
  
所谓"加密"，就是算出下式的c：  
  
　　 ≡ c (mod n)  
  
A的公钥是 (3233, 17)，B的m假设是65，那么可以算出下面的等式：  
  
 ≡ 2790 (mod 3233)  
  
于是，c等于2790，B就把2790发给了A。  
（2）解密要用私钥(n,d)  
  
A拿到B发来的2790以后，就用自己的私钥(3233, 2753) 进行解密。可以证明，下面的等式一定成立：  
  
　　 ≡ m (mod n)  
  
也就是说，c的d次方除以n的余数为m。现在，c等于2790，私钥是(3233, 2753)，那么，A算出  
  
　　 ≡ 65 (mod 3233)  
  
因此，A知道了B加密前的原文就是65。  
  
至此，"加密--解密"的整个过程全部完成。  
  
我们可以看到，如果不知道d，就没有办法从c求出m。而前面已经说过，**要知道d就必须分解n，这是极难做到的**，所以RSA算法保证了通信安全。  
  
你可能会问，公钥(n,e) 只能加密小于n的整数m，**那么如果要加密大于n的整数，该怎么办**？有两种解决方法：

**一种是把长信息分割成若干段短消息，每段分别加密**；

另一种是**先选择一种"对称性加密算法"**（比如DES），

用这种算法的密钥加密信息，**再用RSA公钥加密DES密钥**。

**私钥解密的证明**  
  
最后，我们来证明，为什么用私钥解密，一定可以正确地得到m。也就是证明下面这个式子：  
  
　　 ≡ m (mod n)  
  
因为，根据加密规则  
  
　　 ≡ c (mod n)  
于是，c可以写成下面的形式：  
　　 c = - kn  
  
将c代入要我们要证明的那个解密规则：

≡ m (mod n)  
  
它等同于求证  
  
　　 ≡ m (mod n)  
  
由于  
  
　　ed ≡ 1 (mod φ(n))  
  
所以  
  
　　ed = hφ(n)+1  
  
将ed代入：  
  
　　 ≡ m (mod n)  
-------------------------------------------------------------

接下来，分成两种情况证明上面这个式子。  
  
（1）m与n互质。  
  
根据欧拉定理，此时  
  
　　 ≡ 1 (mod n)  
  
得到  
  
　　 × m ≡ m (mod n)  
  
原式得到证明。  
（2）m与n不是互质关系。  
  
此时，由于n等于质数p和q的乘积，所以m必然等于kp或kq。  
  
以 m = kp为例，考虑到这时k与q必然互质，则根据欧拉定理，下面的式子成立：  
  
　　 ≡ 1 (mod q)  
  
进一步得到  
  
　　 × kp ≡ kp (mod q)  
  
即  
　　 ≡ kp (mod q)  
  
将它改写成下面的等式  
  
　　 = tq + kp  
  
这时t必然能被p整除，**即 t=t'p**  
  
　　= t'pq + kp  
  
因为 m=kp，n=pq，所以  
  
　　 ≡ m (mod n)  
  
原式得到证明。

RSA数字签名

一 用RSA生成签名

在RSA中，被签名的消息、密钥以及最终生成的签名都是以数字形式表示的。

在对文本进行签名时，需要事先对文本编码成数字。

签名 = mod N （用RSA生成签名）

这里所使用的D和N就是签名者的私钥。

签名就是对消息的D次方求mod N的结果，也就是说将消息和自己相乘D次，然后再除以N求余数，最后求得的余数就是签名。

生成签名后，发送者就可以将消息和签名发送给接收者了。

二 用RSA验证签名

由签名求得的消息 = mod N （用RSA验证签名）

这里所使用的E和N就是签名者的公钥。

接收者计算签名的E次方并求mod N，

**得到“由签名求得的消息”，并将其与发送过来的“消息”内容进行对比**，

如果两者一致，则签名验证成功，否则签名验证失败。

**数字签名的作用:保证数据完整性,机密性和发送方角色的不可抵赖性  
加密与签字结合时，两套公私钥是不同的。**

为了方面演示，手动生成一个密钥对

（项目中的秘钥对由开发来生成，会直接给到我们）

生成秘钥对的时候，可以指定生成秘钥的长度，

一般推荐使用1024bit, 1024bit的rsa公钥，加密数据时，最多只能加密117byte的数据），数据量超过这个数，则需要对数据进行分段加密。

但是目前1024bit长度的秘钥已经被证明了不够安全，尽量使用2048bit长度的秘钥。2048bit长度的秘钥，最多245byte长度的数据。

上面生成秘钥的时候提到过在我们加密的时候，如果数据长度超过了当前秘钥的所能处理最大长度，则需要进行分段加密，

分段加密：通俗易懂的讲就是把原来一长串的数据，分割成多段，每段的大小控制在秘钥的最大加密数量之内，加密完了之后再把数据进行拼接。

分段解密：经过分段加密的数据，在进行解密的时候我们也要将它进行分成多段，然后解密之后再进行拼接就能得到原来的数据内容。