RSA真的是困扰了我很久，看着非常简单，但是p,q两个素数的问题，还是比较复杂。

下面我们来捋一捋RSA算法的实现过程：

1. 首先要生成两个大素数 p, q (保密)
2. 计算 n= pq，f(n) = (p-1)\*(q-1).              【n公开，即N；    f(n)即欧拉函数值，需要保密 】
3. 随机选取正整数 1 < e < f(n),满足gcd( e,f(n) ) ==1 .  【e是公开的加密密钥，即E 】
4. 计算d,满足 d\*e \equiv 1（mod  f(n)）.      【d是保密的解密密钥】
5. 由此得到公钥（N,E），私钥（N,D）
6. 加密变换：对于明文 m\in Z_{n},密文为    c = m^{e} mod n.
7. 解密变换：对于密文 c \in Z_{n},明文为    m = c^{^{d}} mod n.

**1.大素数生成**

所以我们先来生成这两个素数，我们知道我们使用random库就可以生成一个随机数，但是我们怎么保证我们生成的数是素数呢？

于是就有了我们下面的内容：Miller-Rabin 素性检测

定义：设n>2是一个奇数，设n-1=2^{s}*m，其中s是非负整数，m>0是奇数。如果

b^{m} \equiv 1\left ( mod n\right )

或者存在一个r，0 <= r < s ，使得

b^{2^{r}m} \equiv -1\left ( mod n \right )= n-1\left ( mod n \right )

则称n通过以b为基的Miller-Rabin素性检测。

注意：1.这里通过素性检测只能证明该数可能是素数

           2.不能通过素性检测则证明该数一定不是素数

这里基于上面的定义有两个定理：（是素性测试的依据）

* 设p>2是一个素数。对任意整数b>0，如果gcd（b，p）=1，则p一定能通过以b为基的Miller-Rabin测试
* 如果n>2是一个奇合数，则至多有（n-1）/4 个b， 0 <b<n , 使得n通过以b为基的Miller-Rabin测试

定理一就是我们常规的判断最大公约数的方法，大家都应该知道，不过这个在数字很大的时候，即使你从2~\sqrt{n}之间逐个测试，在RSA算法的大数下，这也是一个不小的工作量。

下面我们来具体看看Miller-Rabin测试的具体描述：

设n>2是一个奇数，设n-1 = 2^{s}m，其中s是非负整数，m>0是奇数，Miller-Rabin素性测试算法如下：

1. 随机均匀的选取  b \in {1，2，...，n-1}
2. r <-- 0 ; z <-- b^{m}mod n.  如果z = 1 或者 z = n-1,则n通过测试；n可能是素数，结束；跳转第3步
3. 如果 r = s-1，则 s 是合数，结束；否则跳转第四步
4. r <--  r+1 ; z <--z^{^{2}}mod n.  如果 z = n - 1，则 n 通过了测试，n可能是素数，结束；否则跳转第3步

这样判断的正确性至少为75%，出错概率小于25%，b的个数足够多时，更准确。

下面我给出python的代码实现：（部分做了微调）

# 针对随机取得p，q两个数的素性检测

def miller\_rabin\_test(n): # n为要检验得数

p = n - 1

r = 0

# 寻找满足n-1 = 2^s \* m 的s,m两个数

# n -1 = 2^r \* p

while p % 2 == 0: # 最后得到为奇数的p(即m)

r += 1

p /= 2

b = random.randint(2, n - 2) # 随机取b=（0.n）

# 如果情况1 b得p次方 与1 同余 mod n

if fastExpMod(b, int(p), n) == 1:

return True # 通过测试,可能为素数

# 情况2 b得（2^r \*p）次方 与-1 (n-1) 同余 mod n

for i in range(0,7): # 检验六次

if fastExpMod(b, (2 \*\* i) \* p, n) == n - 1:

return True # 如果该数可能为素数，

return False # 不可能是素数

这里我们看到在测试的时候，使用到了大数的幂次取模，所以这里我们要先实现大数模的算法，如下：

# 模N大数的幂乘的快速算法

def fastExpMod(b, e, m): # 底数，幂，大数N

result = 1

e = int(e) #这里不转化的话，在python3下会出现type error

while e != 0:

if e % 2 != 0: # 按位与

e -= 1

result = (result \* b) % m

e >>= 1

b = (b \* b) % m

return result

 下面给出大素数生成的代码吧：（上面的重复内容不再写出）

# 生成大素数：

def create\_prime\_num(keylength): # 为了确保两素数乘积n 长度不会太长，使用keylength/2

while True:

# Select a random number n

# n = random.randint(0, 1<<int(halfkeyLength))

n = random.randint(0, keylength)

if n % 2 != 0:

found = True

# 如果经过10次素性检测，那么很大概率上，这个数就是素数

for i in range(0, 10):

if miller\_rabin\_test(n): #返回True 通过一轮测试

pass

else:

found = False #返回False则为合数，重新产生随机数

break

if found:

return n

我们得到生成的大素数之后，就比较简单了。

**2.生成密钥**

这里直接给出代码实现：

# 生成密钥（包括公钥和私钥）

def create\_keys(keylength):

p = create\_prime\_num(keylength / 2) 前面的生成大素数

q = create\_prime\_num(keylength / 2)

n = p \* q

# fn是euler函数值

fn = (p - 1)\*(q - 1)

e = selectE(fn, keylength / 2)

d = match\_d(e, fn)

return (n, e, d)

# 随机在（1，fn）选择一个E， 满足gcd（e,fn）=1

def selectE(fn, halfkeyLength):

while True:

# e and fn are relatively prime

e = random.randint(0, fn)

if math.gcd(e, fn) == 1:

return e

# 根据选择的e，匹配出唯一的d

def match\_d(e, fn):

d = 0

while True:

if (e \* d) % fn == 1:

return d

d += 1

至此我们得到了公钥和私钥。

**3，加解密实现**

#加密

def encrypt(M, e, n):

return fastExpMod(M, e, n)

#加密

def decrypt( C, d, m):

return fastExpMod(C, d, m)

4.测试截图：

可以完成读取文本的加密，注意标点符号如果是中文的标点，会出现乱码，但不会影响信息的读取。

​