**实践活动：用Python实现RSA公钥加密和解密算法**

说明： 需要先下载并安装Anaconda3-5.0.1-Windows-x86\_64.exe软件，安装库sympy。

以下是调用sympy库实现算法的关键代码，具体的请参见RSA\_Example.py文件。

sympy.crypto.crypto.rsa\_public\_key(*p*, *q*, *e*)

返回RSA公钥对(*n*, *e*), 其中*n*是两个素数的乘积，而*e*与ϕ(*n*)互素（其中ϕ(*n*)是Euler totient函数）。如果与任何假定相冲突则返回False。

>>> from sympy.crypto.crypto import rsa\_public\_key

>>> p, q, e = 3, 5, 7

>>> rsa\_public\_key(p, q, e)

(15, 7)

>>> rsa\_public\_key(p, q, 30)

False

sympy.crypto.crypto.rsa\_private\_key(*p*, *q*, *e*)

返回RSA私钥(*n*, *d*), 其中*n*是两个素数的乘积，而*d*是*e* (mod ϕ(*n*))的逆. 如果与任何假定相冲突则返回False。

>>> from sympy.crypto.crypto import rsa\_private\_key

>>> p, q, e = 3, 5, 7

>>> rsa\_private\_key(p, q, e)

(15, 7)

>>> rsa\_private\_key(p, q, 30)

False

sympy.crypto.crypto.encipher\_rsa(*i*, *key*)

通过计算*ie* (mod *n*)返回*i*的加密，此处*key*是公钥 (*n*, *e*)。

>>> from sympy.crypto.crypto import encipher\_rsa, rsa\_public\_key

>>> p, q, e = 3, 5, 7

>>> puk = rsa\_public\_key(p, q, e)

>>> msg = 12

>>> encipher\_rsa(msg, puk)

3

sympy.crypto.crypto.decipher\_rsa(*i*, *key*)

通过计算*id* (mod *n*)返回*i*的解密，此处*key*是私钥 (*n*, *d*)。

>>> from sympy.crypto.crypto import decipher\_rsa, rsa\_private\_key

>>> p, q, e = 3, 5, 7

>>> prk = rsa\_private\_key(p, q, e)

>>> msg = 3

>>> decipher\_rsa(msg, prk)

12

Kid RSA是RSA的一个用于中小学教学的版本，因为其不涉及到指数。

Alice想与Bob交互信息，Bob生成密钥如下。

密钥生成:

* 随机选择正整数*a*, *b*, *A*, *B*.
* 计算*M* = *a b* − 1, *e* = *A M* + *a*, *d* = *B M* + *b*, *n* = (*e d* − 1) / *M*.
* 公钥为(*n*, *e*), Bob将公钥发送给Alice。
* 私钥为(*n*, *d*), Bob将私钥保密存储。

加密:

若*p*是明文，则密文为*c* = *p e* (mod *n*).

解密:

若c是密文，则明文为*p* = *c d* (mod *n*).

sympy.crypto.crypto.kid\_rsa\_public\_key(*a*, *b*, *A*, *B*)

>>> from sympy.crypto.crypto import kid\_rsa\_public\_key

>>> a, b, A, B = 3, 4, 5, 6

>>> kid\_rsa\_public\_key(a, b, A, B)

(369, 58)

sympy.crypto.crypto.kid\_rsa\_private\_key(*a*, *b*, *A*, *B*)

计算*M* = *a b* − 1, *e* = *A M* + *a*, *d* = *B M* + *b*, *n* = (*e d* − 1) / *M*. 私钥为*d*, Bob将私钥保密存储。

>>> from sympy.crypto.crypto import kid\_rsa\_private\_key

>>> a, b, A, B = 3, 4, 5, 6

>>> kid\_rsa\_private\_key(a, b, A, B)

(369, 70)

sympy.crypto.crypto.encipher\_kid\_rsa(*msg*, *key*)

其中*msg*是明文，而*key*是公钥。

>>> from sympy.crypto.crypto import (encipher\_kid\_rsa, kid\_rsa\_public\_key)

>>> msg = 200

>>> a, b, A, B = 3, 4, 5, 6

>>> key = kid\_rsa\_public\_key(a, b, A, B)

>>> encipher\_kid\_rsa(msg, key)

161

sympy.crypto.crypto.decipher\_kid\_rsa(*msg*, *key*)

其中*msg*是密文，而*key*是私钥。

>>> from sympy.crypto.crypto import (kid\_rsa\_public\_key, kid\_rsa\_private\_key, decipher\_kid\_rsa, encipher\_kid\_rsa)

>>> a, b, A, B = 3, 4, 5, 6

>>> d = kid\_rsa\_private\_key(a, b, A, B)

>>> msg = 200

>>> pub = kid\_rsa\_public\_key(a, b, A, B)

>>> pri = kid\_rsa\_private\_key(a, b, A, B)

>>> ct = encipher\_kid\_rsa(msg, pub)

>>> decipher\_kid\_rsa(ct, pri)

200

# In[2]:

# p, q是两个不同的素数，e与(p-1)(q-1)互素（没有异于1和自身的公因子）

p, q, e = 3, 5, 7

# In[3]:

# 生成公钥(n,e)，其中n = pq.

rsa\_public\_key(p, q, e)

# In[4]:

# 若上面的任何假定不满足，则返回False。

rsa\_public\_key(p, q, 30)

# In[5]:

# 从SymPy中导入RSA私钥模块

from sympy.crypto.crypto import rsa\_private\_key

# In[6]:

# p, q是两个不同的素数，d与e模(p-1)(q-1)互逆，即de = 1 (mod (p-1)(q-1)).

p, q, e = 3, 5, 7

# In[7]:

# 生成公钥(n,d)，其中n = pq.

rsa\_private\_key(p, q, e)

# In[8]:

# 若上面的任何假定不满足，则返回False。

rsa\_private\_key(p, q, 30)

# In[14]:

from sympy.crypto.crypto import encipher\_rsa, rsa\_public\_key

# 准备公钥，选取p,q两个不同素数。

p, q, e = 13, 5, 7

# 生成公钥pub\_key

pub\_key = rsa\_public\_key(p, q, e)

# 待加密的明文消息

msg = 12

# 使用公钥pub\_key对明文进行加密

encipher\_rsa(msg, pub\_key)

# In[15]:

from sympy.crypto.crypto import decipher\_rsa, rsa\_private\_key

p, q, e = 13, 5, 7

# 生成私钥pri\_key

pri\_key = rsa\_private\_key(p, q, e)

# 待解密的密文消息

msg = 38

# 使用私钥pri\_key对密文进行解密

decipher\_rsa(msg, pri\_key)