RSA实现报告

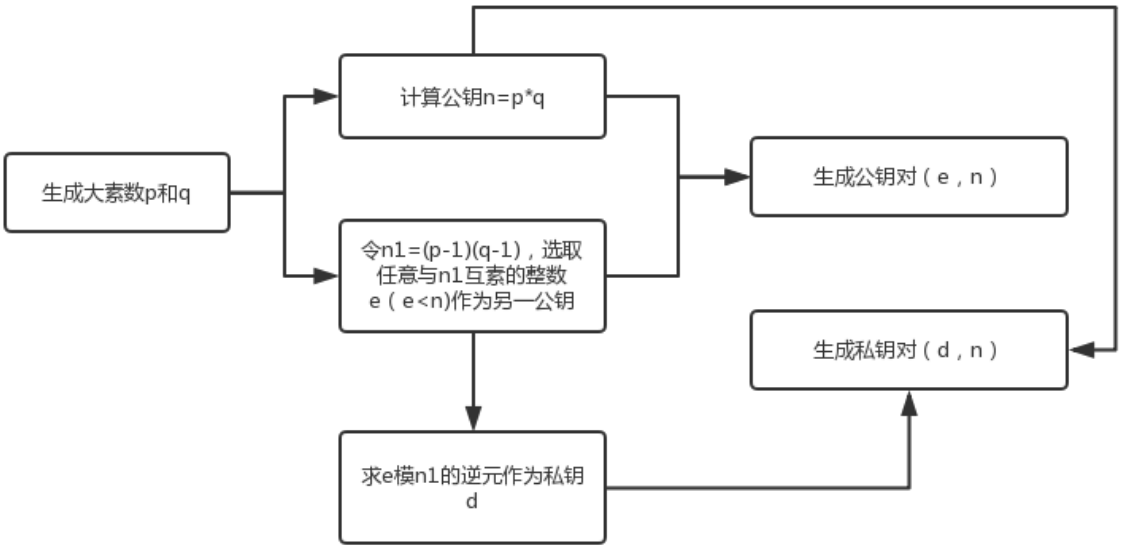
姓名	学号	联系方式
张家豪	16337303	13138478912

一、实验内容

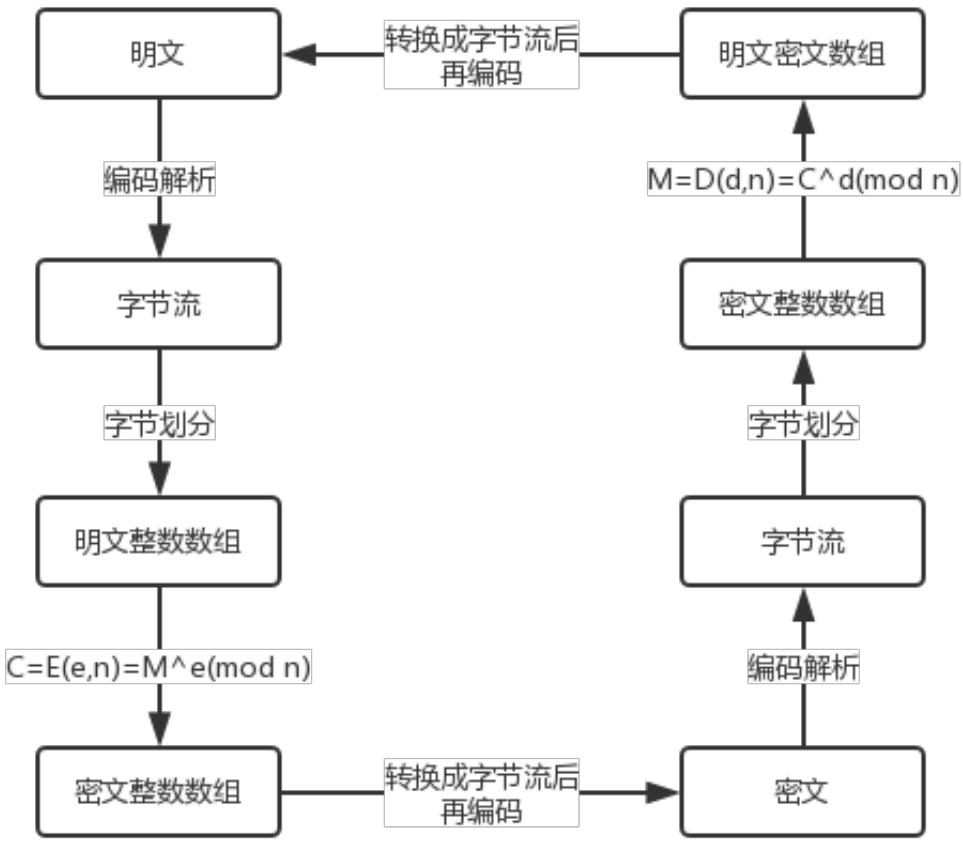
实现任意位数的RSA的加密和解密

二、RSA算法原理

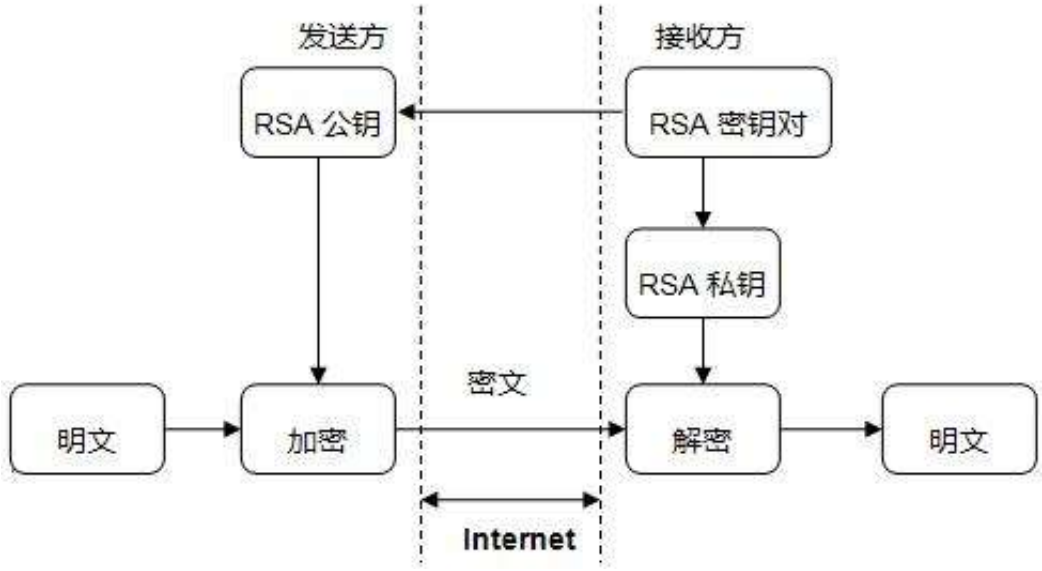
1. 公钥和私钥生成过程



2. RSA加密解密过程



3. RSA加密通信过程



二、代码模块介绍

1. 大整数生成：先生成01字符串，最高位必须为1，为了方便转换成整数，最低位也置为1:

```python
def proBin(w):  # w表示希望产生位数
    list = []
    list.append(1)  #最高位定为1
    for i in range(w - 2):
        c = numpy.random.randint(0, 2)
        list.append(c)
    list.append(1) #最低位也定为1
    ls2 = [str(j) for j in list]
    ls3 = ''.join(ls2)
    b = int(ls3[0])
    for i2 in range(len(ls3) - 1):
        b = b << 1
        b = b + int(ls3[i2 + 1])
    return b
```

2. Miller-Rabin素数检验：

   其检验的过程大致如下：给定奇整数 $n \geq 3$ 和安全参数 k 。写 $n-1=2^s t$ ，其中 t 为奇整数。

   （1） 随机选取整数 b ， $2 \leq b \leq n-2$.

   （2）计算 $r_0 \equiv b^t (mod\ n)$.

   （3）如果 $r_0 = 1$ 或 $r_0 = n-1$ ，则通过检验，可能为素数，回到（1）.继续选取另一个随机整数 b ， $2 \leq b \leq n-2$ ；否则，有 $r_0 \neq 1$ 以及 $r_0 \neq n-1$ ，计算 $r1 \equiv r_0^2 (mod\ n)$.

   （4）如果 $r_1 = n-1$ ，则通过检验，可能为素数，回到（1）.继续选取另一个随机整数 b ， $2 \leq b \leq n-2$ ；否则，有 $r_1 \neq n-1$ ，计算 $r2 \equiv r_1^2 (mod\ n)$.

   如此计算下去，

   （s+2）如果 $r_{s-1} = n-1$ ，则通过检验，可能为素数，回到（1）.继续选取另一个随机整数 b ， $2 \leq b \leq n-2$ ；否则，有 $r_{s-1} \neq n-1$ ，**n为合数**。

代码如下：

```python
    def miller_rabin(n, k=50): #素数检验，k=50保证出错的概率足够小
        if n < 6:
            return [False, False, True, True, False, True][n] #0-5的数直接给出，加快判断。
        elif n & 1 == 0:
            return False
        s = 0
        d = n - 1
        while d % 2 == 0:
            s = s + 1
            d = d >> 1
        for _ in range(k):
```

```
            a = numpy.random.randint(pow(2,63))
            x = pow(a, d, n)
            if x == 1 or x == n-1:
                continue
            for _ in range(s-1):
                x = pow(x, 2, n)
                if x == 1:
                    return False
                elif x == n - 1:
                    a = 0
                    break
            if a:
                return False
        return True
```

3. 求模的逆元：利用辗转相除法

```
def mod_reversi(x, y): #求模的逆元
    if x % y == 0:
        return 0, 0
    if(x % y == 1):
        return 1, -(x // y);
    t2 = -(x // y )
    t, t_ = mod_reversi(y, x % y)
    return t_, t+t2*t_
```

4. 生成公钥和私钥对：

```
def getKeys(p, q, bit): #获得公钥和私钥对
    e = getPrime(bit)
    n1 = (p-1)*(q-1)
    d,_ = mod_reversi(e,n1)
    d = d % n1
    return e, d
```

5. 字符串、字节流和整数数组之间的相互转换：

```

#.....密文的转换.....
def ints2bytes(ints, size):
    res = b''
    for i in ints:
        res += i.to_bytes(size,byteorder='little')
    return res

def bytes2str(bytes_):
```

```
    str = ''
    for i in bytes_:
        str += chr(i)
    return str

def str2bytes(str):
    res = b''
    for j in str:
        tmp = ord(j)
        res += bytes([tmp])
    return res

def bytes2ints(bytes_, size):
    res = []
    count = len(bytes_) // size
    for i in range(count):
        t = bytes_[i * size : (i+1) * size]
        tmp = int.from_bytes(t, byteorder='little')
        res.append(tmp)
    return res

def ints2str(ints, size):
    bytes_ = ints2bytes(ints, size)
    return bytes2str(bytes_)

def str2ints(str, size):
    byte_ = str2bytes(str)
    return bytes2ints(byte_, size)
#.....

#.....明文的转换.....
def ints2str_M(ints, bit):
    res = b''
    for i in ints:
        res += i.to_bytes(bit,byteorder= 'little').rstrip(b'\x00')
    return res.rstrip(b'\x00')



def str2ints_M(str, byte):
    bytes_ = str.encode()
    res = []
    for i in range((bytes_.__sizeof__()-33) // byte + 1):
        t = bytes_[i*byte : (i+1)*byte]
        tmp = int.from_bytes(t,byteorder='little')
        res.append(tmp)
    return res
#.....
```

6. RSA加密和解密

```python
#.....加密和解密.....
def encrypt(str, e, N, size):
    ints = str2ints_M(str, size)
    print("明文变成数字: ", ints)
    res = []
    for m in ints:
        res.append(pow(m,e,N))
        # print(res)
    print("密文数组",res)
    str = ints2str(res, 128)
    return str

def decrypt(str, d, N, size):
    ints =[]
    try:
        ints = str2ints(str, 128)
    except Exception:
        ints = str2ints(str, 512)
    print("密文变成数字", ints)
    res = []
    for m in ints:
        res.append(pow(m, d, N))
    print("解码后的数字:" , res)
    res = ints2str_M(res, size).decode()
    return res
#.....
```

四、实验结果展示

1、加、解密短文本


RSA加密
公钥 N
0x83b1fc49c879be4e67a7c29fcd6d15f10bb1ebc68444b1f376bbd865b(
E
0xdb61ce8b
私钥 D
0xa506781602ea82859489a277d5edae641615142f7969f9c2f89f07f0a3
随机生成
输入
希望密码学期末考对我好一点!
加密
解密
输出
0x21b02730411507382c76dd39e4bd10f35d903a2ad0c6f047bd8c41bab52cbc38063d146ba6
35e7dd9f6ddbbc5e817d83a44aaa9cc4a7603b2677c3a396448e03aa13542c7a017e15599aa89
d19fb67f26c5ce225441a7d414e78fec8b8aeda5360702856fa1a087b42d24592920271782a9d
965ede7f2f274adf6a5bd669dd75


RSA加密
公钥 N
0x83b1fc49c879be4e67a7c29fcd6d15f10bb1ebc68444b1f376bbd865b(
E
0xdb61ce8b
私钥 D
0xa506781602ea82859489a277d5edae641615142f7969f9c2f89f07f0a
随机生成
输入
0x21b02730411507382c76dd39e4bd10f35d903a2ad0c6f047bd8c41bab52cbc38063d146ba6
35e7dd9f6ddbbc5e817d83a44aaa9cc4a7603b2677c3a396448e03aa13542c7a017e15599aa89
d19fb67f26c5ce225441a7d414e78fec8b8aeda5360702856fa1a087b42d24592920271782a9d
965ede7f2f274adf6a5bd669dd75
加密
解密
输出
希望密码学期末考对我好一点!

2、加、解密长文本


RSA加密
公钥 N
0x91fd730e0a729acc11c9f111da8a0f36768fe8e54bce0b59bfbf7c8941b
E
0xdbcfdfbd
私钥 D
0x3b0ec27d4c407793e4fa67635ca3a00039b9cb9f83ad64cef81e5a636
随机生成
输入
密码学真是一门有意思的学科！！！叶广智是一个弟弟！！！密码学真是一门有意思的学科！！！叶广智是一个弟弟！！！密码学真是一门有意思的学科！！！叶广智是一个弟弟！！！密码学真是一门有意思的学科！！！叶广智是一个弟弟！！！密码学真是一门有意思的学科！！！叶广智是一个弟弟！！！密码学真是一门有意思的学科！！！叶广智是一个弟弟！！！
加密
解密
输出
ec166700b85fc95b844944f9036a593d30c799cc188ea179a81b0f46293376e3ded0x678ac2ccee4cfb00db213fb981c0f32d0c88ca2886c27b6e13c8372f2a3b43098fc2cd71014b3b9a12f48715dbea1f0326ca230a1e58ef268879d8583a7d02fc25258c508829cb9491771804c008d833363ca5697c398d928aa49a86645b405a2a97a1752a64b1aefdc80609e5e867a64d0565e04d2a86d08ea1d333f93e438b0x8822cafdee970ef8b40154533053636f7e5166453009240a9ff92a867e55b4c5dc41b3fc432d469824757d3d30545fdcea321ced7a2f64490621654c2a4d0369774c0902f5f401635a76ed380d421e0d76d1e59f97cf095f1f66a4c92228adb68656ce645df086e24eed03f20d7900a7e5d914f5d10558b4b8be340e6151cc970x341318d3c9bb7073899e337a0fd5d4c40c35aab71af32f1b9b36e078259b9badb5cbabbb274a59cd2711b9425f87acb5b623784339777e66cdcdbedd958a25b4ca0806d03988e0722adbdd0b15cb958626fad5f9c8321d5f485498b492d

五、实验思考

实验中遇到比较棘手的问题是当确定了加密时块的位数之后，解密完的位数是不确定的，这时候有分割和填充两个方法，我选择的是用某些特定的标识符分割，统一用“0x”进行分割。

附录

1. GUI

```python
from PyQt5 import QtCore, QtGui, QtWidgets
import numpy
from RSA import *

class Ui_MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super(Ui_MainWindow, self).__init__()
        self.setupUi(self)
        self.retranslateUi(self)
```

```
10
11          self.p = 0
12          self.q = 0
13          self.n = 0
14          self.n1 = 0
15          self.e = 0
16          self.d = 0
17          # self.work = False
18
19      def setupUi(self, MainWindow):
20          MainWindow.setObjectName("MainWindow")
21          MainWindow.resize(718, 556)
22          self.centralWidget = QtWidgets.QWidget(MainWindow)
23          self.centralWidget.setObjectName("centralWidget")
24          self.public_key_2 = QtWidgets.QLineEdit(self.centralWidget)
25          self.public_key_2.setGeometry(QtCore.QRect(70, 20, 471, 21))
26          self.public_key_2.setObjectName("public_key_2")
27          self.privat_key = QtWidgets.QLineEdit(self.centralWidget)
28          self.privat_key.setGeometry(QtCore.QRect(70, 50, 471, 21))
29          self.privat_key.setMouseTracking(False)
30          self.privat_key.setObjectName("privat_key")
31          self.public_key = QtWidgets.QLabel(self.centralWidget)
32          self.public_key.setGeometry(QtCore.QRect(20, 20, 26, 19))
33          self.public_key.setObjectName("public_key")
34          self.private_key = QtWidgets.QLabel(self.centralWidget)
35          self.private_key.setGeometry(QtCore.QRect(20, 80, 31, 16))
36          self.private_key.setObjectName("private_key")
37          self.gen_key = QtWidgets.QPushButton(self.centralWidget)
38          self.gen_key.setGeometry(QtCore.QRect(560, 20, 141, 81))
39          self.gen_key.setObjectName("gen_key")
40          self.inputText = QtWidgets.QPlainTextEdit(self.centralWidget)
41          self.inputText.setGeometry(QtCore.QRect(70, 120, 631, 161))
42          self.inputText.setObjectName("inputText")
43          self.outputText = QtWidgets.QPlainTextEdit(self.centralWidget)
44          self.outputText.setGeometry(QtCore.QRect(70, 320, 631, 161))
45          self.outputText.setObjectName("outputText")
46          self.input = QtWidgets.QLabel(self.centralWidget)
47          self.input.setGeometry(QtCore.QRect(20, 120, 31, 16))
48          self.input.setObjectName("input")
49          self.output = QtWidgets.QLabel(self.centralWidget)
50          self.output.setGeometry(QtCore.QRect(20, 320, 31, 16))
51          self.output.setObjectName("output")
52          self.encrypt = QtWidgets.QPushButton(self.centralWidget)
53          self.encrypt.setGeometry(QtCore.QRect(190, 285, 114, 32))
54          self.encrypt.setObjectName("encrypt")
55          self.decrypt = QtWidgets.QPushButton(self.centralWidget)
56          self.decrypt.setGeometry(QtCore.QRect(420, 285, 114, 32))
57          self.decrypt.setObjectName("decrypt")
58          self.public_key_3 = QtWidgets.QLabel(self.centralWidget)
```

```
        self.public_key_3.setGeometry(QtCore.QRect(55, 20, 16, 19))
        self.public_key_3.setObjectName("public_key_3")
        self.public_key_4 = QtWidgets.QLabel(self.centralWidget)
        self.public_key_4.setGeometry(QtCore.QRect(54, 50, 16, 19))
        self.public_key_4.setObjectName("public_key_4")
        self.privat_key_2 = QtWidgets.QLineEdit(self.centralWidget)
        self.privat_key_2.setGeometry(QtCore.QRect(70, 80, 471, 21))
        self.privat_key_2.setMouseTracking(False)
        self.privat_key_2.setObjectName("privat_key_2")
        self.public_key_5 = QtWidgets.QLabel(self.centralWidget)
        self.public_key_5.setGeometry(QtCore.QRect(55, 80, 16, 19))
        self.public_key_5.setObjectName("public_key_5")
        MainWindow.setCentralWidget(self.centralWidget)
        self.menuBar = QtWidgets.QMenuBar(MainWindow)
        self.menuBar.setGeometry(QtCore.QRect(0, 0, 718, 22))
        self.menuBar.setObjectName("menuBar")
        MainWindow.setMenuBar(self.menuBar)
        self.mainToolBar = QtWidgets.QToolBar(MainWindow)
        self.mainToolBar.setObjectName("mainToolBar")
        MainWindow.addToolBar(QtCore.Qt.TopToolBarArea,
self.mainToolBar)
        self.statusBar = QtWidgets.QStatusBar(MainWindow)
        self.statusBar.setObjectName("statusBar")
        MainWindow.setStatusBar(self.statusBar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.clickEvents()

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
"MainWindow"))
        self.public_key.setText(_translate("MainWindow", "公钥"))
        self.private_key.setText(_translate("MainWindow", "私钥"))
        self.gen_key.setText(_translate("MainWindow", "随机生成"))
        self.input.setText(_translate("MainWindow", "输入"))
        self.output.setText(_translate("MainWindow", "输出"))
        self.encrypt.setText(_translate("MainWindow", "加密"))
        self.decrypt.setText(_translate("MainWindow", "解密"))
        self.public_key_3.setText(_translate("MainWindow", "N"))
        self.public_key_4.setText(_translate("MainWindow", "E"))
        self.public_key_5.setText(_translate("MainWindow", "D"))

    def Gen_keys(self):
        bit = 512
        self.p = getPrime(bit)
        self.q = getPrime(bit)
```

```
106
107         print("p:", self.p)
108         print("q:", self.q)
109         self.n = self.p*self.q
110         print("n:", self.n)
111
112         self.e, self.d = getKeys(self.p, self.q, 32)
113         self.public_key_2.setText(hex(self.n))
114         # self.public_key_2.displayText()
115
116         self.n1 = (self.p - 1) * (self.q - 1)
117
118         self.privat_key.setText(hex(self.e))
119         # self.privat_key.displayText()
120         self.privat_key_2.setText(hex(self.d))
121         # self.privat_key_2.displayText()
122         print("e:", self.e)
123         print("d:", self.d)
124
125
126     def E(self):
127         self.e = bytes2ints_(self.privat_key.text())[0]
128         self.n = bytes2ints_(self.public_key_2.text())[0]
129
130         M = self.inputText.toPlainText()
131         print("M" , M)
132         P = encrypt(M, self.e, self.n, 64)
133         self.outputText.setPlainText(P)
134         print(P)
135         # M_ = decrypt(P,d,n,8)
136         # print(M_)
137
138
139     def D(self):
140         self.d = bytes2ints_(self.privat_key_2.text())[0]
141         self.n = bytes2ints_(self.public_key_2.text())[0]
142
143         S = self.inputText.toPlainText()
144         print("S", S)
145         # print("lol", decrypt(S, self.d, self.n, 512))
146         self.outputText.setPlainText(decrypt(S, self.d, self.n, 64))
147
148     def clickEvents(self):
149         self.gen_key.clicked.connect(self.Gen_keys)
150         self.encrypt.clicked.connect(self.E)
151         self.decrypt.clicked.connect(self.D)
```