|  |  |
| --- | --- |
| **计算机网络安全实验报告** | |
| 点到点加密通讯 | |
| 图片包含 标牌  已生成极高可信度的说明 | |
| **姓名** | 田丰瑞 |
| **班级** | 软件73班 |
| **学号** | 2172213528 |
| **电话** | 18744296191 |
| **Email** | [tianfr@stu.xjtu.edu.cn](mailto:tianfr@stu.xjtu.edu.cn?subject=软件73%20田丰瑞) |
| **日期** | 2020-5-14 |

# 声明

网络安全实验的所有代码已经全部开源于我的GitHub（仅开源代码部分），项目地址：

<https://github.com/tianfr/Internet-Security-ExpCode/>

# 点到点加密通讯实验

## 实验要求

使用非对称加密算法设计一个点到点加密通讯软件，要求实现完整性和鉴别。

## 设计思路

首先设计一个点到点通讯软件，然后在软件通讯时利用一系列加密算法加密。

### 点到点通讯软件

#### 功能性需求

该软件需实现包括但不限于以下功能：

**用户注册：**用户通过点击注册按钮进入注册界面，通过输入一次用户名两次用户密码实现用户注册。

**用户登录：**用户通过输入合法用户名和密码实现用户登录，并进入软件主页面。

**实时查询在线用户：**登录用户在软件界面可实时刷新查看在线用户人员。

**选择聊天用户：**登录用户可在软件界面选择想聊天的用户人员。

**向指定用户发送消息：**用户可在软件主界面向用户发送消息。

**向指定用户发送文件：**用户可在软件主界面向指定用户发送本地文件。

**实时接收用户发送消息：**软件主界面实时显示其他用户发送的消息。

**实时接收用户发送文件：当其他用户向本用户发送文件时，软件主界面实时提示查收文件。**

**消息保存：用户可以保存聊天记录。**

#### **加密通讯需求**

保证消息完整性、机密性、防重放

加密关键技术：RSA非对称加密，Hash算法完整性检查，DES对称加密算法，挑战应答机制

CA认证机构，数字信封，公钥证书

在通讯过程中，服务器充当一个CA，起到分发公钥证书和密钥的过程。

公钥证书制作流程

通讯整体流程如下：

**前提条件：每个客户端有CA的公钥，CA维护所有用户的公钥，每个客户端有自己的公私钥。**

##### 连接建立开始——公钥传递

假设客户端A想与A通讯，首先客户端A向服务器发出请求申请公钥证书请求包括一段随机数作为挑战应答的内容，服务器返回一个用A的公钥加密好B的公钥证书，A用自己的私钥解密并校验解密后的挑战应答结果，无误后读取解密内容获得B的公钥，A再将自己的公钥证书和随机数作为挑战应答整合一起用hash创建校验码，最后用B的公钥加密后发给B，B得到A发过来的包后先用自己的私钥解密，然后校验哈希函数，准确无误后从数据中获得A的公钥，并用A的公钥加密挑战应答结果后发给A，A收到后用自己的私钥解开比对挑战应答结果，无误证明连接建立。



图1：用户A与用户B的连接建立

##### 消息传输——a与b之间即时消息传递

A将发送的消息用自己的私钥签名后创建hash校验码，将校验码加到消息后面再用B的公钥加密后发送给B，B收到消息后用自己的私钥解密后，用hash函数校验完整性，无误后用A的公钥验证签名，签名无误后读取A的消息，AB之间消息传递完成。

表1：即时消息报文传输协议

|  |  |  |  |
| --- | --- | --- | --- |
| **即时消息报文协议** | | | |
| **数据内容** | **数据处理** | | |
| HASH校验码 |  |  | 接收方公钥加密 |
| 接收方用户地址 | HASH256 |  |
| 发送方用户地址 |
| 发送时间 |
| 即时消息内容 | 用发送方私钥加密后压缩 |



图2：即时消息传输示例图

##### 消息传输——a与b之间文件传递

A首先创建一个随机的DES密钥，然后将文件内容及信息，时间戳和A的签名用zip压缩后再用DES密钥加密文件内容，并用B的公钥加密DES密钥创建数字信封，将加密文件信息和数字信封整合到一起利用socket发送给B，B用自己的私钥解开DES密钥，然后解压缩后用DES密钥解密，验证签名后读取文件信息，至此，文件传输完成。

表2：传输文件协议

|  |  |  |
| --- | --- | --- |
| **传输文件协议** | | |
| **数据内容** | **数据处理** | |
| HASH校验码 | HASH256 | |
| 接收方用户地址 |
| 发送方用户地址 |
| DES密钥 | 发送方私钥签名，接收方公钥加密 |  |
| 发送时间 |  |
| 文件内容 | zip压缩后用随机DES密钥加密 |



图3：文件传输示意图

## 原理简介

### **非对称算法加密通讯**

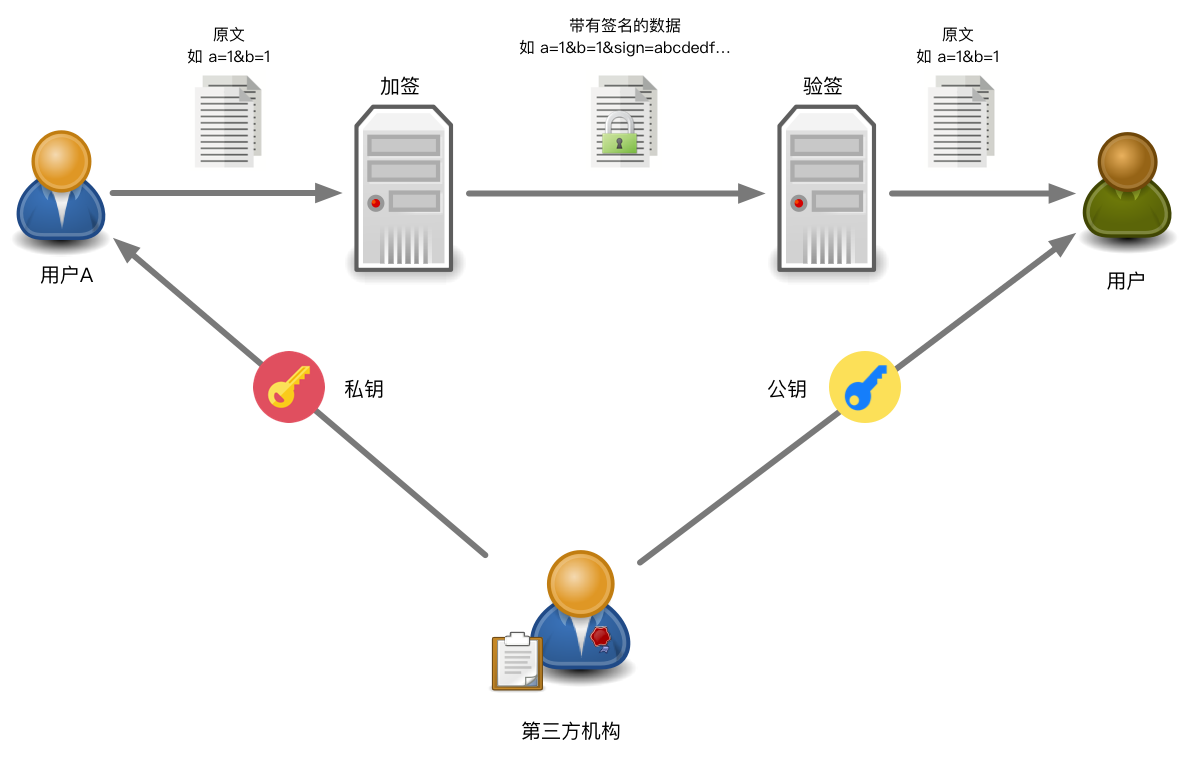
非对称加密算法是一种密钥的保密方法。

非对称加密算法需要两个密钥：**公开密钥（publickey:简称公钥）和私有密钥（privatekey:简称私钥）**。公钥与私钥是一对，如果用公钥对数据进行加密，只有用对应的私钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。 非对称加密算法实现机密信息交换的基本过程是：甲方生成一对密钥并将公钥公开，需要向甲方发送信息的其他角色(乙方)使用该密钥(甲方的公钥)对机密信息进行加密后再发送给甲方；甲方再用自己私钥对加密后的信息进行解密。甲方想要回复乙方时正好相反，使用乙方的公钥对数据进行加密，同理，乙方使用自己的私钥来进行解密。

另一方面，甲方可以使用自己的私钥对机密信息进行签名后再发送给乙方；乙方再用甲方的公钥对甲方发送回来的数据进行验签。

甲方只能用其私钥解密由其公钥加密后的任何信息。 非对称加密算法的保密性比较好，它消除了最终用户交换密钥的需要。

非对称密码体制的特点：算法强度复杂、安全性依赖于算法与密钥但是由于其算法复杂，而使得加密解密速度没有对称加密解密的速度快。对称密码体制中只有一种密钥，并且是非公开的，如果要解密就得让对方知道密钥。所以保证其安全性就是保证密钥的安全，而非对称密钥体制有两种密钥，其中一个是公开的，这样就可以不需要像对称密码那样传输对方的密钥了。这样安全性就大了很多。



### 点到点通讯原理

在电信中，点到点连接是指两个节点或端点之间的通信连接。

一个例子是电话呼叫，其中一个电话相互连接，一个呼叫者所说的只能被另一个呼叫者听到。 这与点对多点或广播连接形成对比，其中许多节点可以接收由一个节点发送的信息。 点对点通信链路的其他示例是租用线路，微波中继链路和双向无线电。

该术语还用于计算机网络和计算机体系结构中，以指代仅链接两个计算机或电路的线路或其它连接，而不是诸如可连接许多通信设备的总线或交叉开关的其它网络拓扑。

点对点有时缩写为P2P。 P2P的这种使用不同于指向用于文件共享网络的对等的P2P。

### Python socket编程

Python 提供了两个级别访问的网络服务。：

* 低级别的网络服务支持基本的 Socket，它提供了标准的 BSD Sockets API，可以访问底层操作系统Socket接口的全部方法。
* 高级别的网络服务模块 SocketServer， 它提供了服务器中心类，可以简化网络服务器的开发。

#### 什么是 Socket?

Socket又称"套接字"，应用程序通常通过"套接字"向网络发出请求或者应答网络请求，使主机间或者一台计算机上的进程间可以通讯。

#### socket()函数

Python 中，我们用 socket（）函数来创建套接字，语法格式如下：

socket.socket([family[, type[, proto]]])

#### 参数

* family: 套接字家族可以使AF\_UNIX或者AF\_INET
* type: 套接字类型可以根据是面向连接的还是非连接分为SOCK\_STREAM或SOCK\_DGRAM
* protocol: 一般不填默认为0.

#### Socket 对象(内建)方法

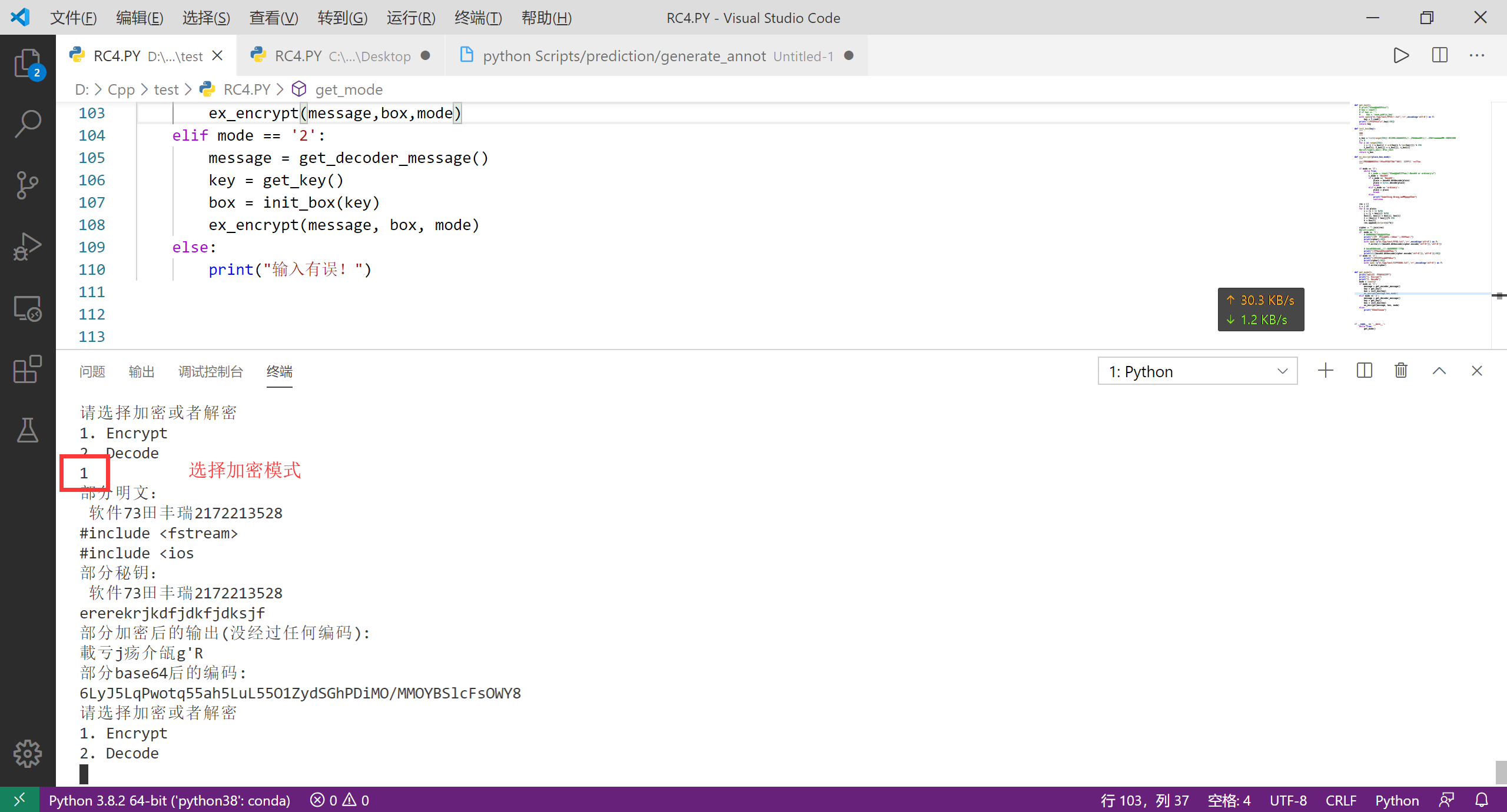
| **函数** | **描述** |
| --- | --- |
| 服务器端套接字 | |
| s.bind() | 绑定地址（host,port）到套接字， 在AF\_INET下,以元组（host,port）的形式表示地址。 |
| s.listen() | 开始TCP监听。backlog指定在拒绝连接之前，操作系统可以挂起的最大连接数量。该值至少为1，大部分应用程序设为5就可以了。 |
| s.accept() | 被动接受TCP客户端连接,(阻塞式)等待连接的到来 |
| 客户端套接字 | |
| s.connect() | 主动初始化TCP服务器连接，。一般address的格式为元组（hostname,port），如果连接出错，返回socket.error错误。 |
| s.connect\_ex() | connect()函数的扩展版本,出错时返回出错码,而不是抛出异常 |
| 公共用途的套接字函数 | |
| s.recv() | 接收TCP数据，数据以字符串形式返回，bufsize指定要接收的最大数据量。flag提供有关消息的其他信息，通常可以忽略。 |
| s.send() | 发送TCP数据，将string中的数据发送到连接的套接字。返回值是要发送的字节数量，该数量可能小于string的字节大小。 |
| s.sendall() | 完整发送TCP数据，完整发送TCP数据。将string中的数据发送到连接的套接字，但在返回之前会尝试发送所有数据。成功返回None，失败则抛出异常。 |
| s.recvfrom() | 接收UDP数据，与recv()类似，但返回值是（data,address）。其中data是包含接收数据的字符串，address是发送数据的套接字地址。 |
| s.sendto() | 发送UDP数据，将数据发送到套接字，address是形式为（ipaddr，port）的元组，指定远程地址。返回值是发送的字节数。 |
| s.close() | 关闭套接字 |
| s.getpeername() | 返回连接套接字的远程地址。返回值通常是元组（ipaddr,port）。 |
| s.getsockname() | 返回套接字自己的地址。通常是一个元组(ipaddr,port) |
| s.setsockopt(level,optname,value) | 设置给定套接字选项的值。 |
| s.getsockopt(level,optname[.buflen]) | 返回套接字选项的值。 |
| s.settimeout(timeout) | 设置套接字操作的超时期，timeout是一个浮点数，单位是秒。值为None表示没有超时期。一般，超时期应该在刚创建套接字时设置，因为它们可能用于连接的操作（如connect()） |
| s.gettimeout() | 返回当前超时期的值，单位是秒，如果没有设置超时期，则返回None。 |
| s.fileno() | 返回套接字的文件描述符。 |
| s.setblocking(flag) | 如果flag为0，则将套接字设为非阻塞模式，否则将套接字设为阻塞模式（默认值）。非阻塞模式下，如果调用recv()没有发现任何数据，或send()调用无法立即发送数据，那么将引起socket.error异常。 |
| s.makefile() | 创建一个与该套接字相关连的文件 |

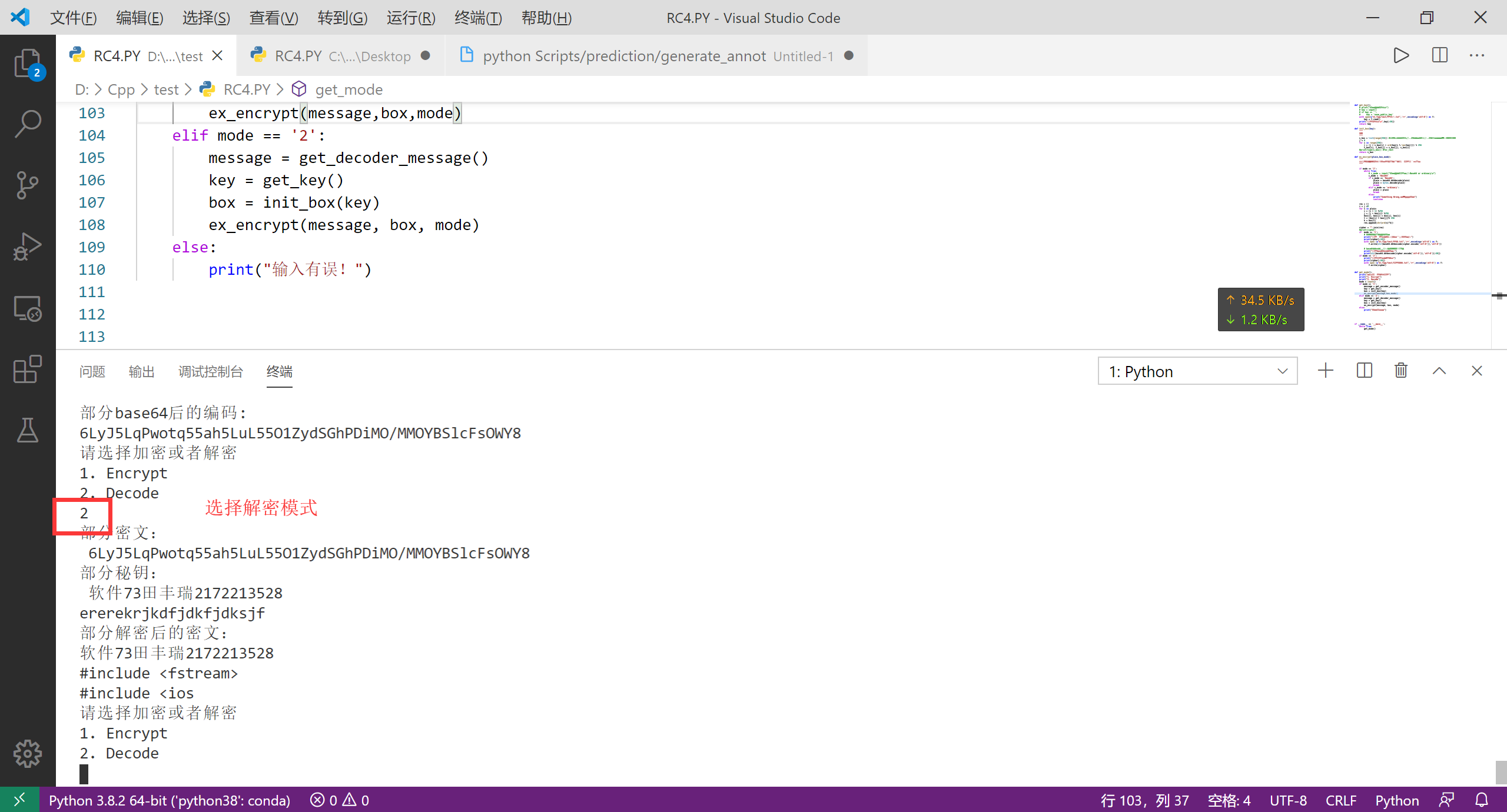
### DES加密简介

# 运行结果

本次使用的明文为我之前写过的一段程序，明文开头有我的姓名学号标识。

### 程序执行结果

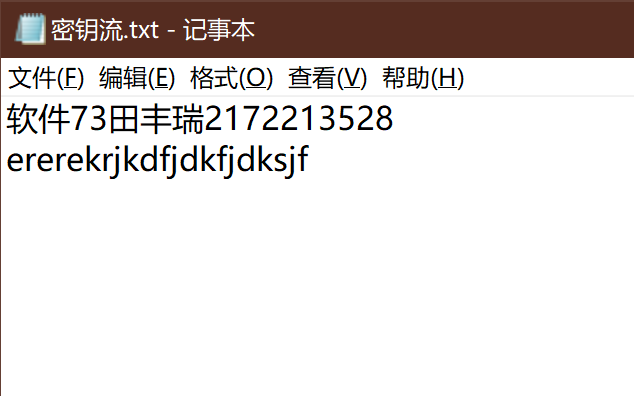




### 明文

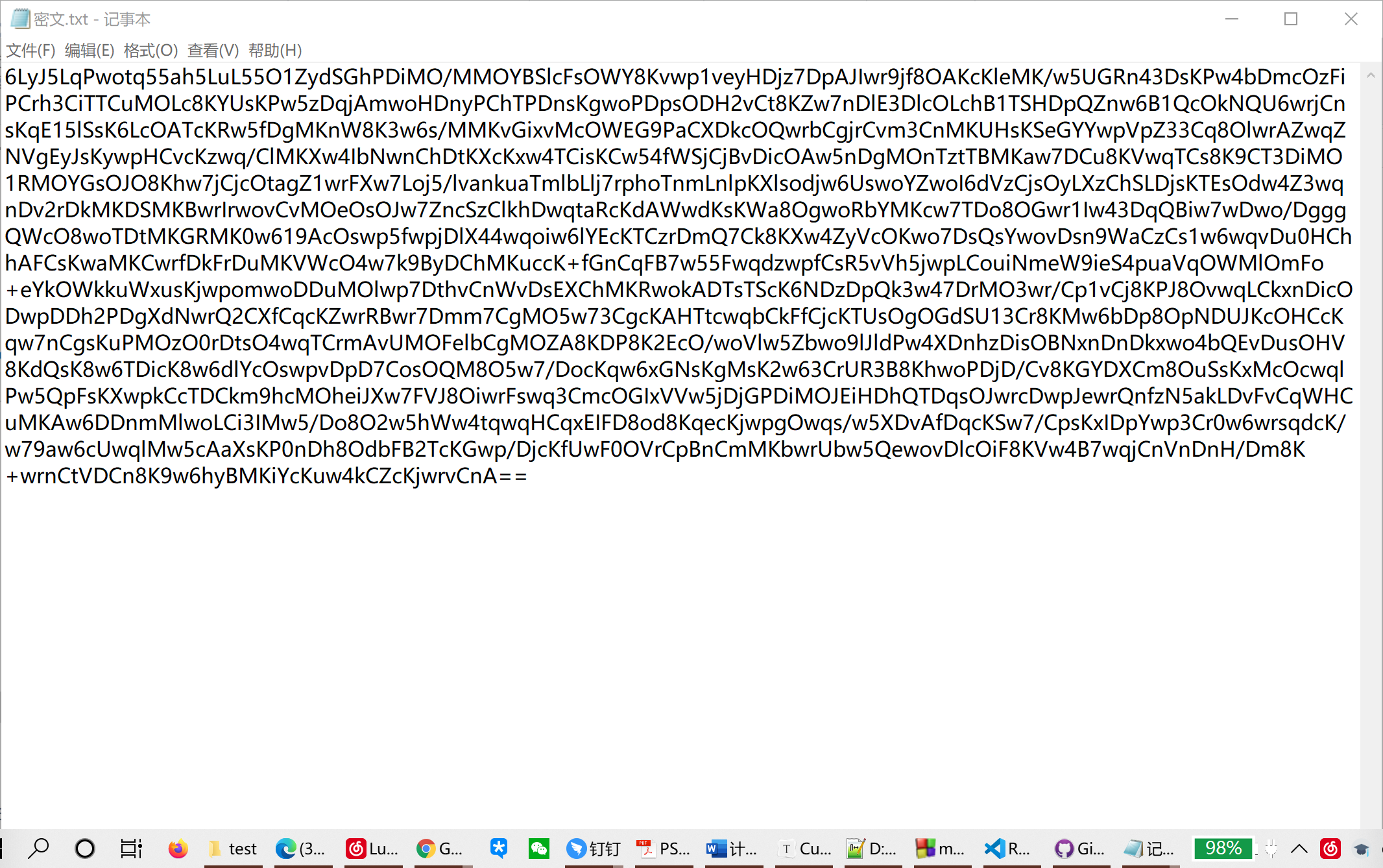


### 密钥流



### 密文

密文已经转换为base64编码格式保证可读性。



### 解密文件



可见已经解密成功。

# 总结

本次实验加深了我对RC4的理解。我个人认为RC4最有意思的地方在于其可通过简单的交换实现加密和解密，其没有用到很深的数学知识。本次实验是我在网络安全课上做的第一个实验，让我对网络安全领域有一个初步的认识。在实现RC4时候由于我搞清楚了RC4的整个流程，因此我并没有用网络现成的代码来实现，而是自己写了一个RC4的函数来实现其算法过程。