

# 计算机网络安全实 验报告

---

点到点加密通讯



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)

---

GitHub Github.com/tianfr

---

日期 2020-5-14

---

# 1 目录

1	目录 .....	2
2	声明 .....	4
3	点到点加密通讯实验内容 .....	4
3.1	实验要求 .....	4
3.2	设计思路 .....	4
3.2.1	点到点通讯软件 .....	4
3.2.1.1	功能性需求 .....	4
3.2.1.2	加密通讯需求 .....	5
3.2.1.3	通讯整体流程 .....	5
3.3	原理简介 .....	8
3.3.1	非对称算法加密通讯 .....	8
3.3.2	点到点通讯原理 .....	9
3.3.3	Python socket 编程 .....	9
3.3.3.1	什么是 Socket? .....	9
3.3.3.2	socket()函数 .....	9
3.3.3.3	参数 .....	9
3.3.3.4	Socket 对象(内建)方法 .....	9
3.3.4	DES 加密简介 .....	12
3.3.4.1	DES 算法简介 .....	12
3.3.4.2	IP 置换 .....	13
3.3.4.3	密钥置换 .....	14
3.3.4.4	E 扩展置换 .....	14
3.3.4.5	S 盒代替 .....	15
3.3.4.6	P 盒置换 .....	18
3.3.4.7	7.IP <sup>-1</sup> 末置换 .....	18
4	点到点加密软件安全性分析 .....	19
4.1	AES 公私钥加密安全性分析 .....	19
4.1.1	RSA 算法加密强度 .....	19

4.1.2	大数因子分解的难度 .....	19
4.2	DES 算法传输文件安全性分析 .....	19
4.2.1	密钥分发 .....	19
4.2.2	DES 算法漏洞 .....	20
5	运行结果 .....	20
5.1	服务器启动 .....	20
5.2	用户注册与登录 .....	20
5.3	消息的加密传输 .....	23
5.4	文件的加密传输 .....	25
6	代码展示 .....	26
6.1	代码逻辑结构 .....	26
6.2	部分代码展示 .....	31
6.2.1	数据库设计 .....	31
6.2.2	Mysql 文件夹代码展示 .....	32
6.2.2.1	Connect.py .....	32
6.2.2.2	connetMysql.py .....	33
6.2.2.3	database.py .....	39
6.2.3	object 文件夹代码展示 .....	39
6.2.3.1	client.py .....	39
6.2.3.2	message.py .....	43
6.2.3.3	myDES.py .....	46
6.2.3.4	myRSA.py .....	47
6.2.3.5	Server.py .....	53
6.2.4	ui 文件夹代码展示 .....	58
6.2.4.1	chat.py .....	58
6.2.4.2	chatWindow.py .....	64
6.2.4.3	chatWindow.ui .....	67
6.2.4.4	Login.py .....	74
6.2.4.5	loginWindow.py .....	75
6.2.4.6	loginWindow.ui .....	77

6.2.4.7	signUp.py .....	81
6.2.4.8	signUpWindow.py .....	82
6.2.4.9	signUpWindw.ui .....	84
6.2.5	客户端主程序 .....	89
6.2.5.1	Main.py 与 main2.py .....	89
7	实验总结 .....	90

## 2 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/>

## 3 点到点加密通讯实验内容

### 3.1 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。

### 3.2 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。

#### 3.2.1 点到点通讯软件

##### 3.2.1.1 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。

### 3.2.1.2 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。

### 3.2.1.3 通讯整体流程

**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。

#### 3.2.1.3.1 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。

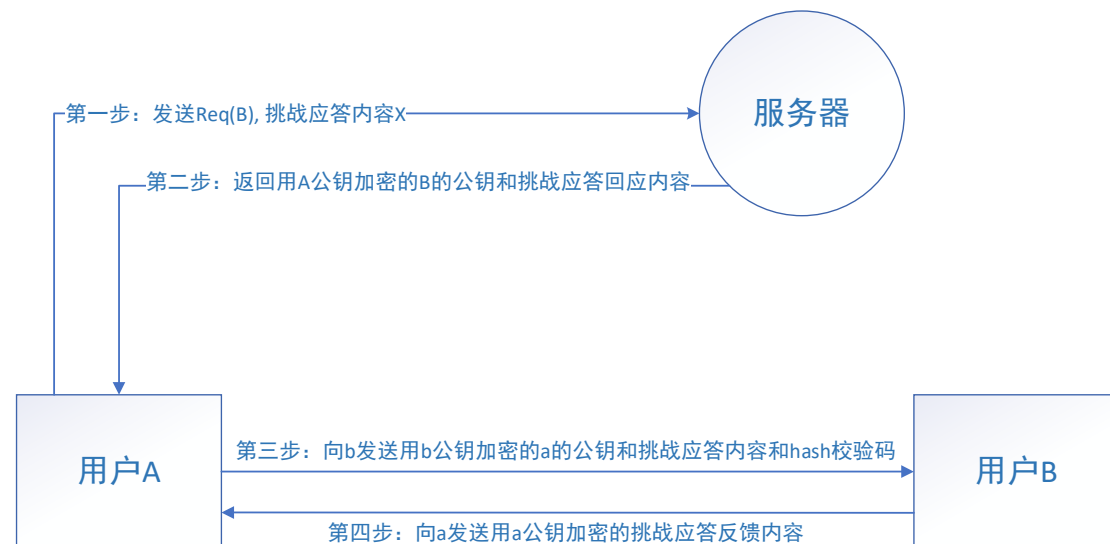


图 1：用户 A 与用户 B 的连接建立

#### 3.2.1.3.2 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	HASH256	
接收方用户地址		
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	接收方公钥加密

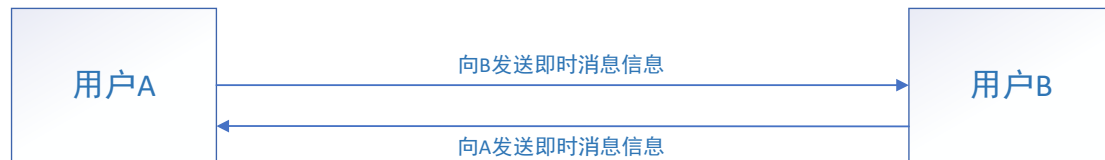


图 2：即时消息传输示例图

### 3.2.1.3.3 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

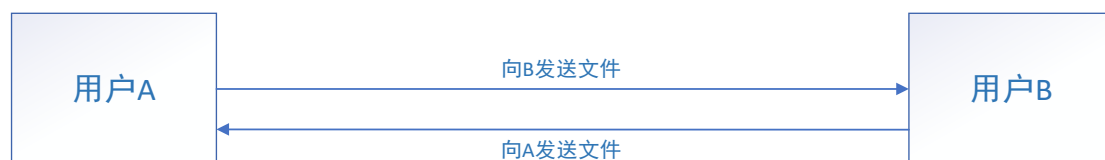


图 3：文件传输示意图

### 3.3 原理简介

#### 3.3.1 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。

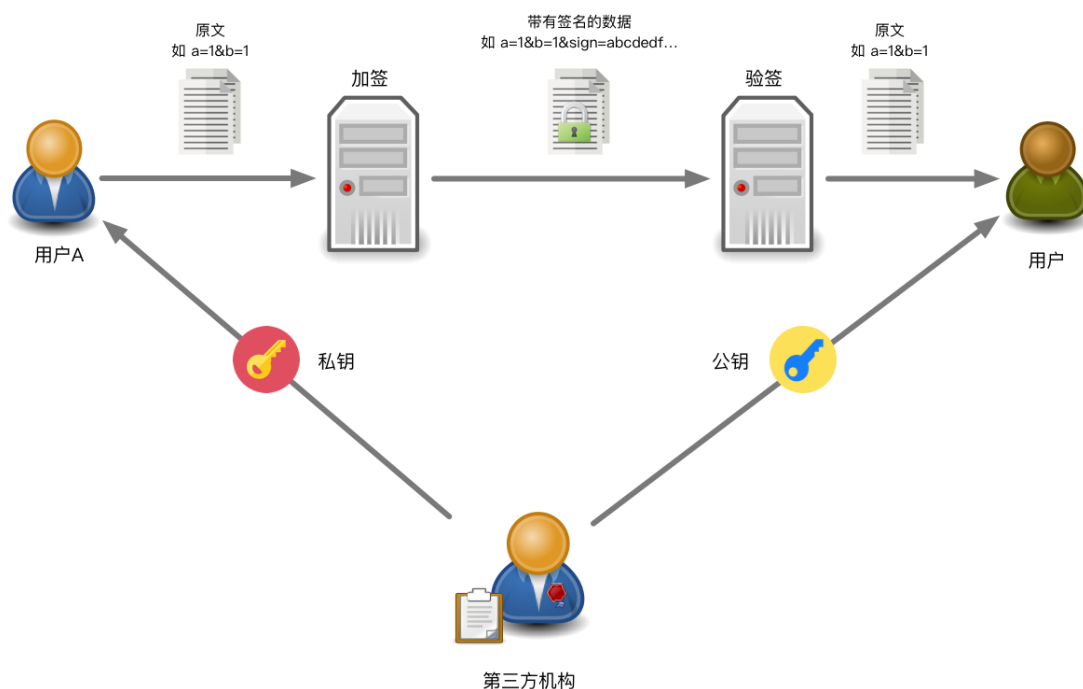


图 4: RSA 加密体制



### 3.3.2 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。

### 3.3.3 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。

#### 3.3.3.1 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。

#### 3.3.3.2 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]]])
```

#### 3.3.3.3 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.

#### 3.3.3.4 Socket 对象(内建)方法

表 3: socket 构建表

函数	描述
	服务器端套接字

函数	描述
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。

函数	描述
s.sendall()	完整发送 TCP 数据，完整发送 TCP 数据。将 <b>string</b> 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 <b>None</b> ，失败则抛出异常。
s.recvfrom()	接收 UDP 数据，与 <b>recv()</b> 类似，但返回值是（ <b>data,address</b> ）。其中 <b>data</b> 是包含接收数据的字符串， <b>address</b> 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， <b>address</b> 是形式为（ <b>ipaddr, port</b> ）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（ <b>ipaddr,port</b> ）。
s.getsockname()	返回套接字自己的地址。通常是一个元组（ <b>ipaddr,port</b> ）
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， <b>timeout</b> 是一个浮点数，单位是秒。值为 <b>None</b> 表示没有超时期。一

函数	描述
	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 <code>connect()</code> )
<code>s.gettimeout()</code>	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 <code>None</code> 。
<code>s.fileno()</code>	返回套接字的文件描述符。
<code>s.setblocking(flag)</code>	如果 <code>flag</code> 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 <code>recv()</code> 没有发现任何数据,或 <code>send()</code> 调用无法立即发送数据,那么将引起 <code>socket.error</code> 异常。
<code>s.makefile()</code>	创建一个与该套接字相关连的文件

### 3.3.4 DES 加密简介

#### 3.3.4.1 DES 算法简介

DES 算法为密码体制中的**对称密码体制**,又被称为美国数据加密标准。

DES 是一个**分组**加密算法,典型的 DES 以 **64 位为分组对数据加密**,加密和解密用的是**同一个算法**。

密钥长 64 位,密钥事实上是 **56 位参与 DES 运算**(第 8、16、24、32、40、48、56、64 位是**校验位**,使得每个密钥都有奇数个 1),分组后的明文组和 56 位的密钥按位替代或交换的方法形成密文组。

DES 算法的主要流程如下图所示,本文按照流程依次介绍每个模块。

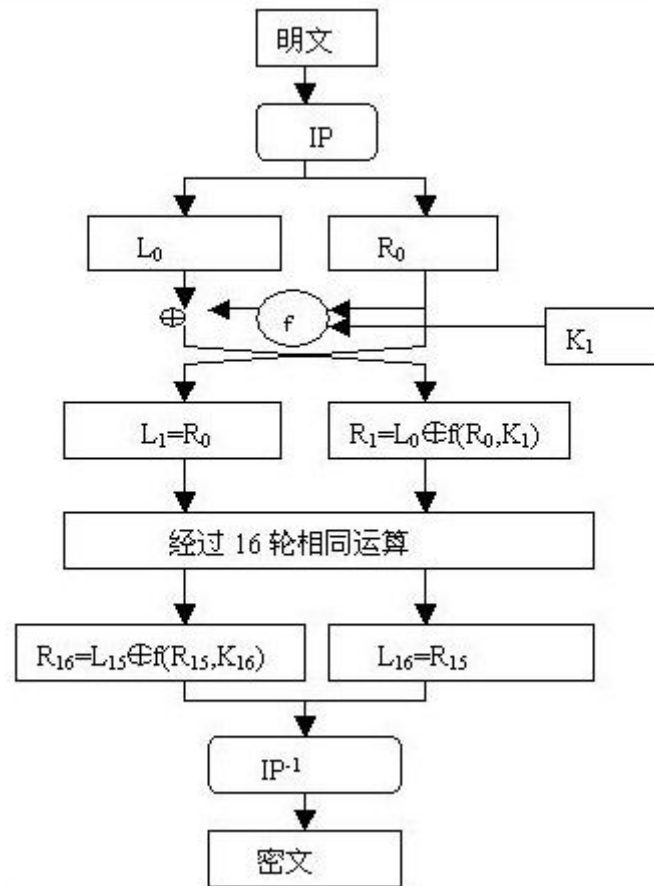


图 5: DES 加密流程

### 3.3.4.2 IP 置换

IP 置换目的是将输入的 64 位数据块按位重新组合，并把输出分为 L0、R0 两部分，每部分各长 32 位。

置换规则如下表所示：

表 4: 置换规则

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

表中的数字代表新数据中此位置的数据在原数据中的位置，即原数据块的第 58 位放到新数据的第 1 位，第 50 位放到第 2 位，……依此类推，第 7 位放到第 64 位。置换后的数据分为 L0 和 R0 两部分，L0 为新数据的左 32 位，R0 为新数据的右 32 位。

要注意一点，位数是从左边开始数的，即最 0x0000 0080 0000 0002 最左边的位为 1，最右边的位为 64。

### 3.3.4.3 密钥置换

不考虑每个字节的第 8 位，DES 的密钥由 64 位减至 56 位，每个字节的第 8 位作为奇偶校验位。产生的 56 位密钥由下表生成（注意表中没有 8,16,24, 32,40,48,56 和 64 这 8 位）：

表 5：56 位密钥生成表

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

在 DES 的每一轮中，从 56 位密钥产生出不同的 48 位子密钥，确定这些子密钥的方式如下：

- 1).将 56 位的密钥分成两部分，每部分 28 位。
- 2).根据轮数，这两部分分别循环左移 1 位或 2 位。每轮移动的位数如下表：

表 6：56 位密钥移动表

轮数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

移动后，从 56 位中选出 48 位。这个过程中，既置换了每位的顺序，又选择了子密钥，因此称为压缩置换。压缩置换规则如下表（注意表中没有 9, 18, 22, 25, 35, 38, 43 和 54 这 8 位）：

表 7：压缩置换规则表

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

置换方法同上，此处省略。

### 3.3.4.4 E 扩展置换

扩展置换目标是 IP 置换后获得的右半部分 R0，将 32 位输入扩展为 48 位(分为 4 位×8 组)输出。

扩展置换目的有两个：生成与密钥相同长度的数据以进行异或运算；提供更长的结果，在后续的替代运算中可以进行压缩。

扩展置换原理如下表：

表 8：扩展置换表

32	1	2	3	4	5
----	---	---	---	---	---

4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

表中的数字代表位，两列黄色数据是扩展的数据，可以看出，扩展的数据是从相邻两组分别取靠近的一位，4位变为6位。靠近32位的位为1，靠近1位的位为32。表中第二行的4取自上组中的末位，9取自下组中的首位。

我们举个例子看一下(虽然扩展置换针对的是上步IP置换中的R0，但为便于观察扩展，这里不取R0举例)：

输入数据 0x1081 1001，转换为二进制就是 0001 0000 1000 0001B，按照上表扩展得下表：

表 9：扩展置换表

1	0	0	0	1	0
1	0	0	0	0	1
0	1	0	0	0	0
0	0	0	0	1	0
1	0	0	0	1	0
1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	1	0

表中的黄色数据是从临近的上下组取得的，二进制为 1000 1010 0001 0100 0000 0010 1000 1010 0000 0000 0000 0010B，转换为十六进制 0x8A14 028A 0002。

扩展置换之后，右半部分数据 R0 变为 48 位，与密钥置换得到的轮密钥进行异或。

### 3.3.4.5 S 盒代替

压缩后的密钥与扩展分组异或以后得到 48 位的数据，将这个数据送入 S 盒，进行替代运算。替代由 8 个不同的 S 盒完成，每个 S 盒有 6 位输入 4 位输出。48 位输入分为 8 个 6 位的分组，一个分组对应一个 S 盒，对应的 S 盒对各组进行代替操作。

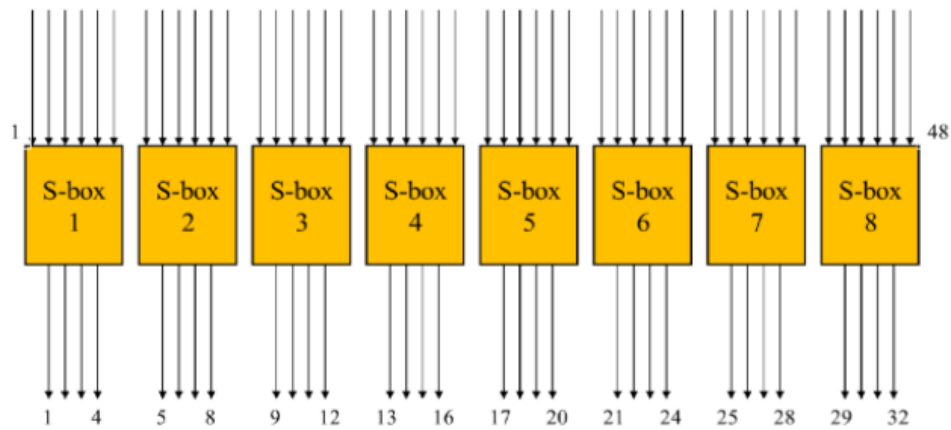


图 6: S 盒

一个 S 盒就是一个 4 行 16 列的表，盒中的每一项都是一个 4 位的数。S 盒的 6 个输入确定了其对应的输出在哪一行哪一列，输入的高低两位做为行数 H，中间四位做为列数 L，在 S-BOX 中查找第 H 行 L 列对应的数据(<32)。

8 个 S 盒如下：

S 盒 1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S 盒 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S 盒 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S 盒 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
---	----	----	---	---	---	---	----	---	---	---	---	----	----	---	----



13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	19
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S 盒 5

2	12	4	1	7	10	11	6	5	8	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	13	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S 盒 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S 盒 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S 盒 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

例如，假设 S 盒 8 的输入为 110011，第 1 位和第 6 位组合为 11，对应于 S 盒 8 的第 3 行；第 2 位到第 5 位为 1001，对应于 S 盒 8 的第 9 列。S 盒 8 的第 3 行第 9 列的数字为 12，因此用 1100 来代替 110011。注意，S 盒的行列计数都是从 0 开始。

代替过程产生 8 个 4 位的分组，组合在一起形成 32 位数据。

S 盒代替时 DES 算法的关键步骤，所有的其他的运算都是线性的，易于分析，而 S 盒是非线性的，相比于其他步骤，提供了更好安全性。

### 3.3.4.6 P 盒置换

S 盒代替运算的 32 位输出按照 P 盒进行置换。该置换把输入的每位映射到输出位，任何一位不能被映射两次，也不能被略去，映射规则如下表：

表 10：映射规则表

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

表中的数字代表原数据中此位置的数据在新数据中的位置，即原数据块的第 16 位放到新数据的第 1 位，第 7 位放到第 2 位，……依此类推，第 25 位放到第 32 位。

例如 0x10A1 0001 进行 P 盒置换后变为 0x8000 0886。

0x10A1 0001 表现为表的形式（第一位位于左上角）原来为

0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1

经 P 盒变换后为

1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
1	0	0	0	0	1	1	0

即 1000 0000 0000 0000 0000 1000 1000 0110B，十六进制为 0x8000 0886。

最后，P 盒置换的结果与最初的 64 位分组左半部分 L0 异或，然后左、右半部分交换，接着开始另一轮。

### 3.3.4.7 7.IP<sup>-1</sup> 末置换

末置换是初始置换的逆过程，DES 最后一轮后，左、右两半部分并未进行交换，而是两部分合并形成一个分组做为末置换的输入。末置换规则如下表：

表 11：末置换规则表

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26

33	1	41	9	49	17	57	25
----	---	----	---	----	----	----	----

置换方法同上，此处省略。

经过以上步骤，就可以得到密文了。

## 4 点到点加密软件安全性分析

### 4.1 AES 公私钥加密安全性分析

#### 4.1.1 RSA 算法加密强度

目前密码的破译主要有 2 种方法。方法之一是密钥的穷尽搜索，其破译方法是尝试所有可能的密钥组合。虽然大多数的密钥尝试都是失败的，但最终有一个密钥让破译者得到原文，这个过程称为密钥的穷尽搜索。方法之二是密码分析。由于 RSA 算法在加密和解密过程都是用指数计算，其计算工作量巨大，用穷尽搜索法进行破译是根本不可能的。因此要对 RSA 算法加密后的信息进行破译只能采用密码分析法，用密码分析法攻击 RSA 密码系统，途径之一是直接计算“规的  $e$  次方根”，但目前还没有解决这一问题的算法，这个问题是现实不可计算的问题；途径之二[4]是想办法计算出  $d$ ，欲得到  $d$ ，可考虑从以下 3 个方面入手。

#### 4.1.2 大数因子分解的难度

著名数学家费马(1601--1665)和勒让德(1752--1833)都研究过分解因子的算法，现代某些更好的算法是勒让德方法的扩展。其中，R. Schroeppe1 算法是好算法中的一类，用此法分解因子仍然需要大约  $e^{\sqrt{\ln N \ln \ln N}}$  次运算，其中  $\ln$  表示自然对数，可见分解行所需的运算次数与密钥的长度有关，随着密钥长度的增加，分解所需的时间会成指数倍增加。对于不同长度的十进制数  $n$ ，Schroeppe1 算法分解  $n$  的因子时所需的运算次数如表 1 所示。

表 12：用 Schroeppe1 分解因子算法的运算次数表

数 $n$ 的十进制位数	50	100	200	300	400
运算次数	$1.4 \times 10^{10}$	$2.3 \times 10^{15}$	$1.2 \times 10^{23}$	$1.5 \times 10^{29}$	$2.7 \times 10^{34}$

就目前的计算机水平用 1024 位二进制(约 340 位十进制)的密钥是安全的，2048 位是绝对安全的。

### 4.2 DES 算法传输文件安全性分析

#### 4.2.1 密钥分发

DES 是对称的分组密码算法，对称的分组密码算法最主要的问题是：由于加解密双方都要使用相同的密钥，因此在发送、接收数据之前，必须完成密钥的分发，因而密钥的分发便成了该加密体系中的最薄弱风险最大的环节。

而在本次实验中，作者使用 RSA 非对称加密算法来分发密钥，因此如果想直接破译密钥的难度转成了破译 RSA 加密算法的难度。关于 RSA 加密算法的破译难度可以参考上

节。由于 RSA 密钥破译的困难性，本次用 DES 加密的文件传输最薄弱的环节——密钥分发得以保障。

#### 4.2.2 DES 算法漏洞

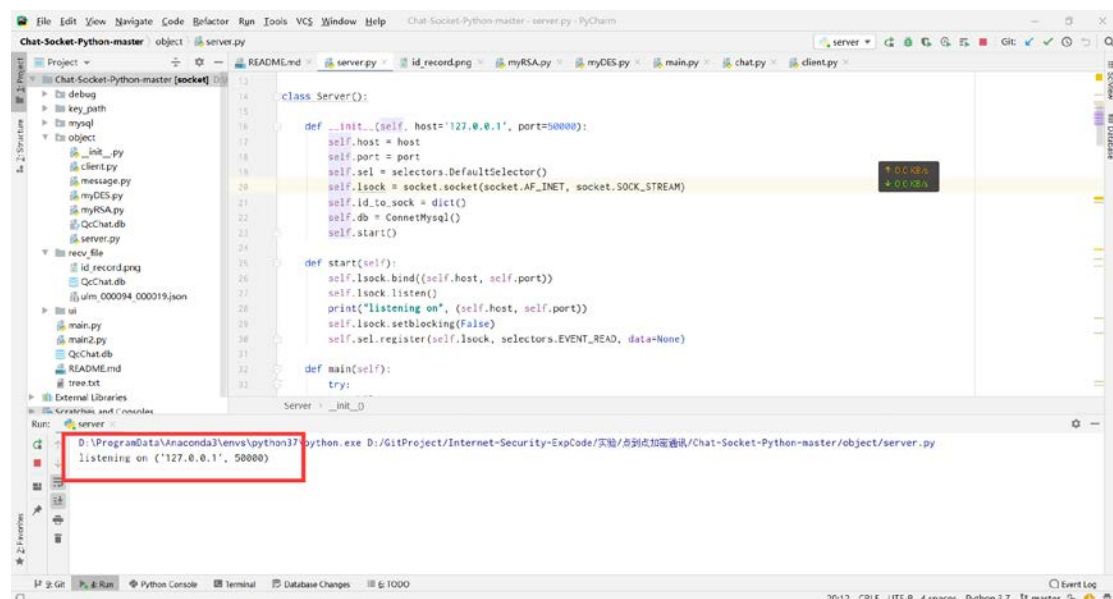
由 DES 算法我们可以看到，DES 算法中只用到 64 位密钥中的其中 65 位而第 8，16，24.....64 位 8 个位并未参与 DES 运算。这一点向我们提出了一个应用上的要求即 DES 的安全性是基于除了 8，16，24.....64 位外的其余 56 位的组合变化才得以保证的。因此，在实际应用中我们应避开使用第 8，16，24.....64 位作为 DES 密钥的有效数据位，而使用其它的 56 位作为有效数据位。只有这样才能保证 DES 算法安全可靠地发挥作用。如果不了解这一点把密钥 8，16，24.....64 位作为有效数据位使用，将不能保证 DES 加密数据的安全性，对运用 DES 来达到保密作用的系统将产生数据被破译的危险。

基于以上的问题，**我们就不能用汉字作为密钥**。因为汉字由两个字节组成，每个字节的 ASCII 码 都大于 127，转换成二进制就是 8 位，不能加奇偶校验位，而且如果去掉第 8，16，24.....64 位就会丢失密钥，而且不同的汉字可能实际上是相同的密钥。

## 5 运行结果

### 5.1 服务器启动

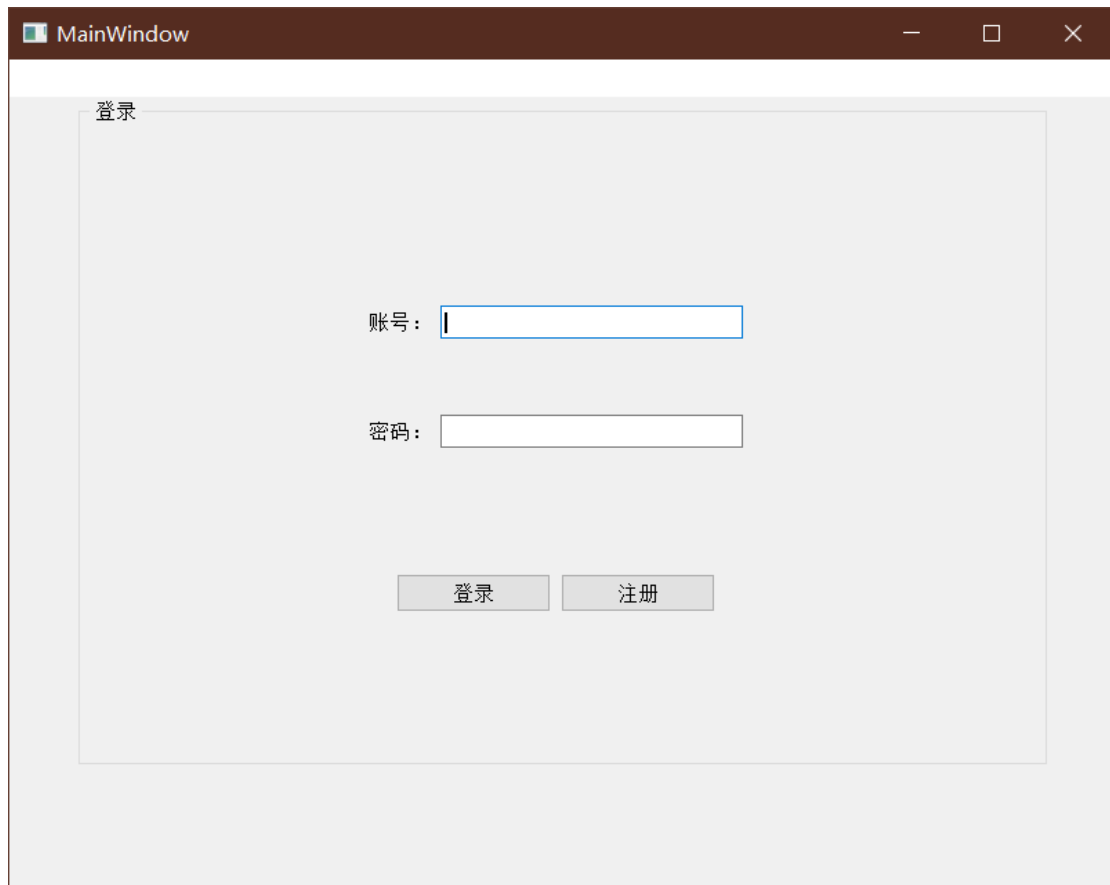
首先启动服务器。



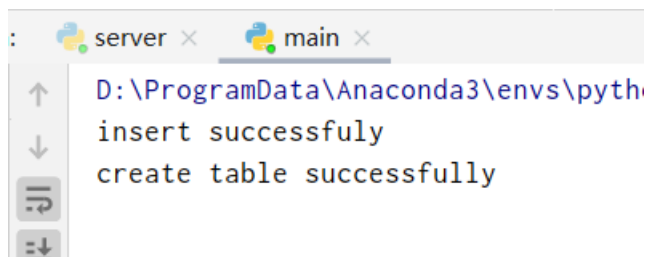
服务器启动，监听端口 50000；

### 5.2 用户注册与登录

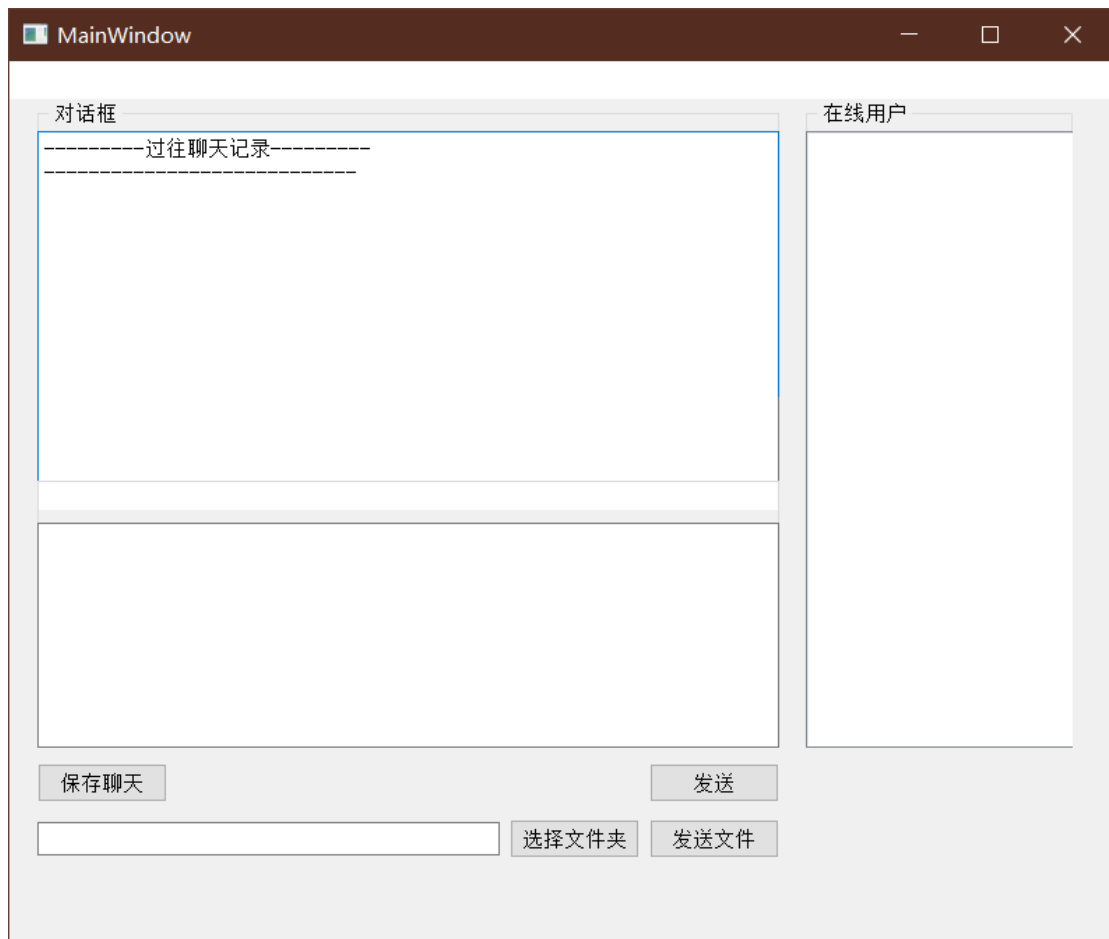
然后启动客户端



第一次使用点击注册按钮注册一个用户名为 user1， 密码为 user1 的用户

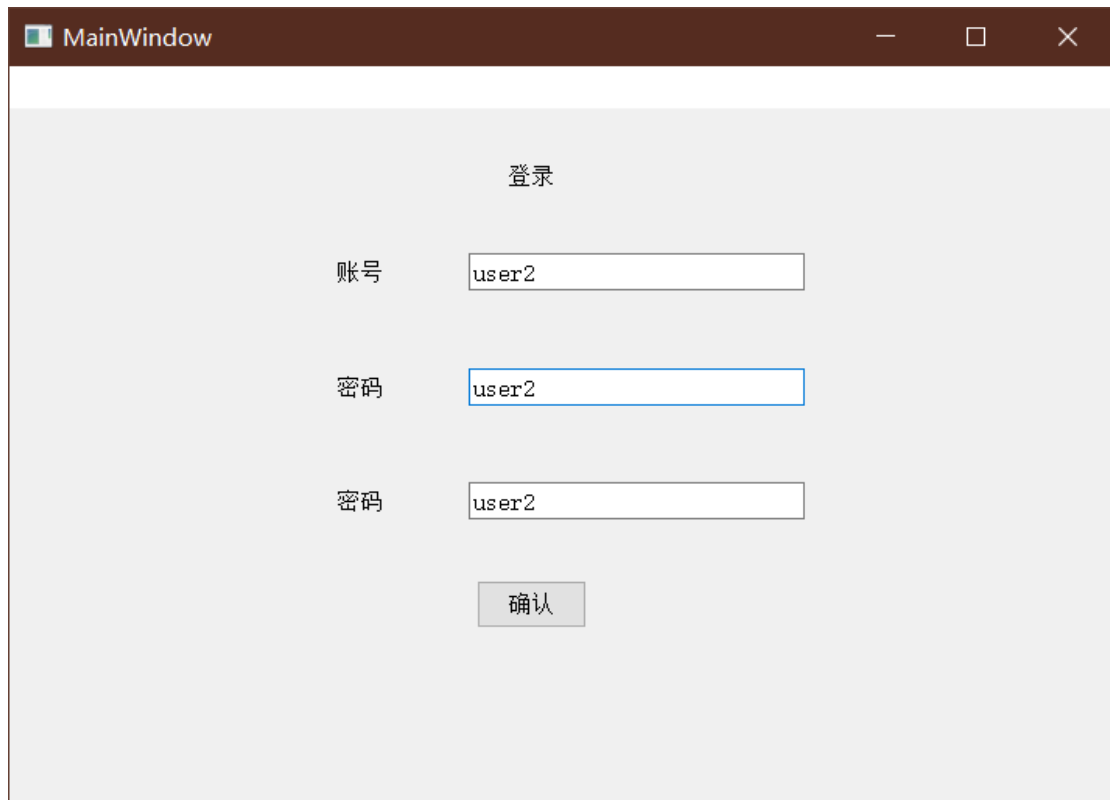


注册成功后，用 user1 用户登录

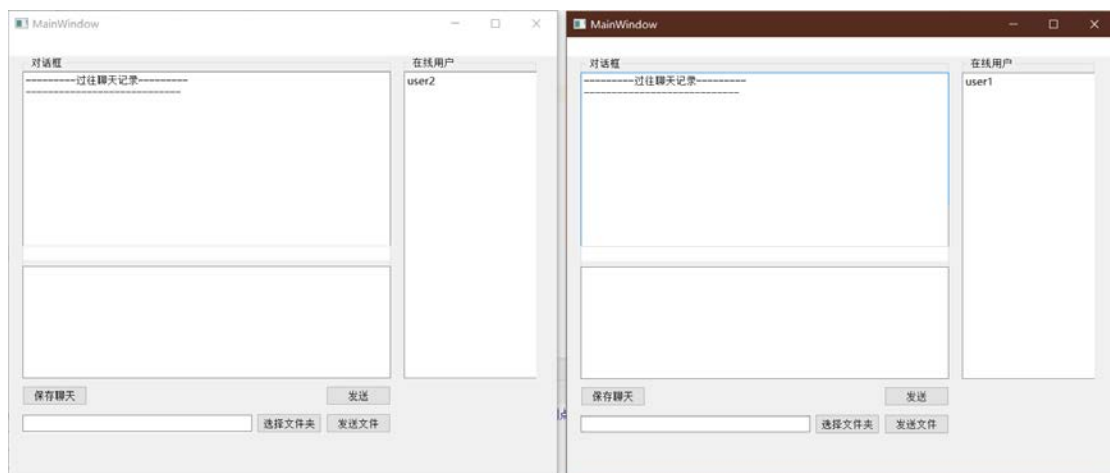


由于现在没有在线用户，因此在线用户为空。

我们再次启动客户端，注册一个用户名 `user2` 密码 `user2` 的用户。



登录后



User1 和 user2 中在线用户互相出现对方。

### 5.3 消息的加密传输

此时我们用 user1 的聊天窗口选择 user2，发送一条消息：Hello， User2！

The screenshot shows a VS Code editor with a Python file named `main.py`. The code is for a chat application. It includes a `login` function that checks if a user exists in a database. If not, it generates RSA keys and adds the user. The `encrypt` function uses the RSA keys to encrypt a message. The `decrypt` function uses the private key to decrypt the message. The `main` function runs the application.

Two sections of the code are highlighted with red boxes:

- Top Red Box:** This section contains the `login` function. It checks if a user exists in the database. If not, it generates RSA keys and adds the user to the database. The code is as follows:
 

```

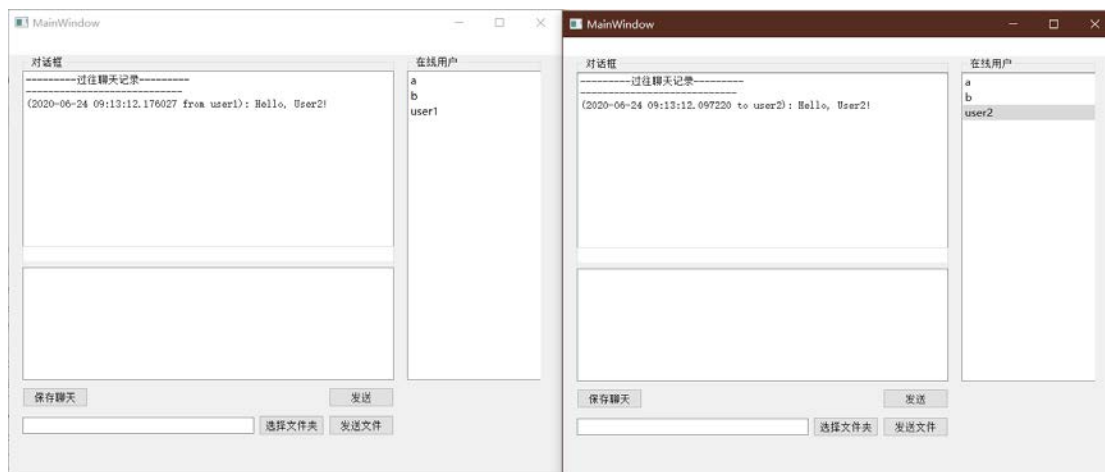
def login():
    user_id = input("Enter user ID: ")
    password = input("Enter password: ")
    if user_id in users:
        if users[user_id] == password:
            print("Login successful")
            return user_id
        else:
            print("Incorrect password")
    else:
        print("User does not exist")
        generate_keys(user_id)
        add_user(user_id, password)
        print("User added successfully")
        return user_id
      
```
- Bottom Red Box:** This section contains the `encrypt` function. It uses the RSA keys to encrypt a message. The code is as follows:
 

```

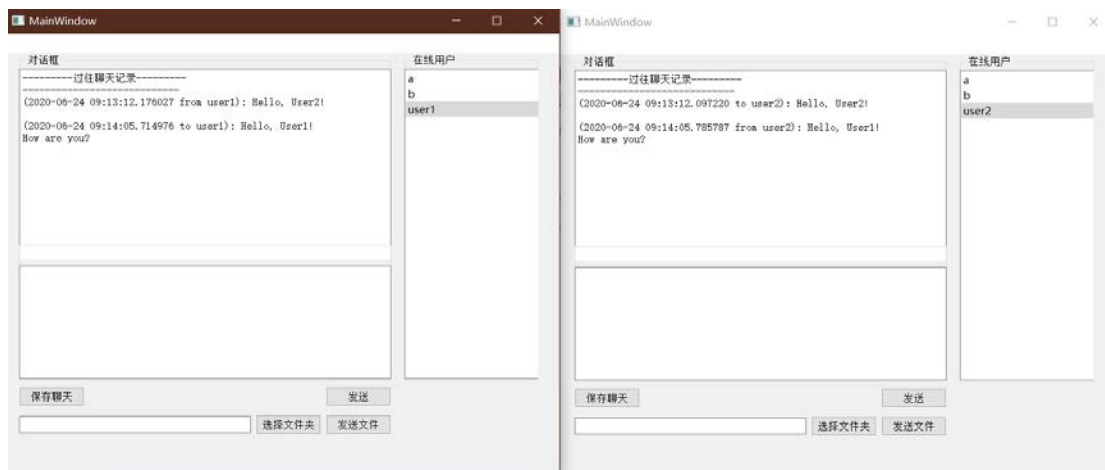
def encrypt(message):
    public_key = get_public_key(user_id)
    encrypted_message = rsa.encrypt(message.encode(), public_key)
    return encrypted_message
      
```

The bottom of the image shows a status bar with the text "Externally added files can be added to Git" and a button "View Files".

首先系统发现 **user1** 没有公钥和私钥，因此服务器自动创建并分发一个公私钥给 **user1**，公私钥创建完毕后 **user1** 用自己的私钥加密消息后发送给 **user2**，**user2** 接收到来自 **user1** 的消息后向服务器请求 **user1** 的公钥，并用公钥解密消息。

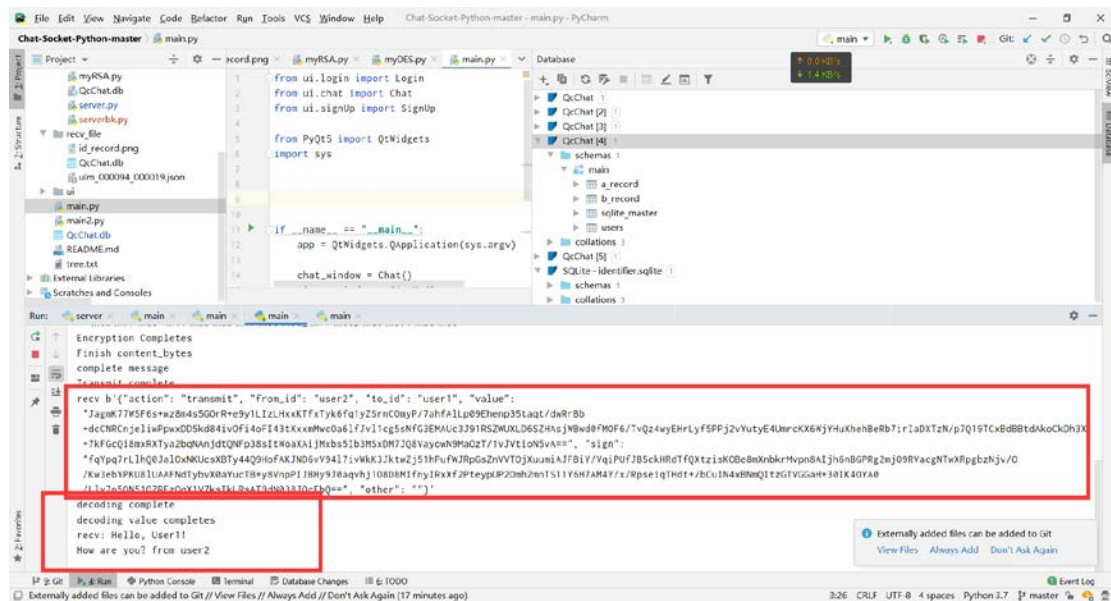


同样的，我们用 `use2` 的账户发送一个消息给 `user1`





同样的, user2 首先向服务器申请一对公私钥, 然后将消息用自己的私钥加密发送给 user1, user1 用 user2 的公钥解密得到消息。

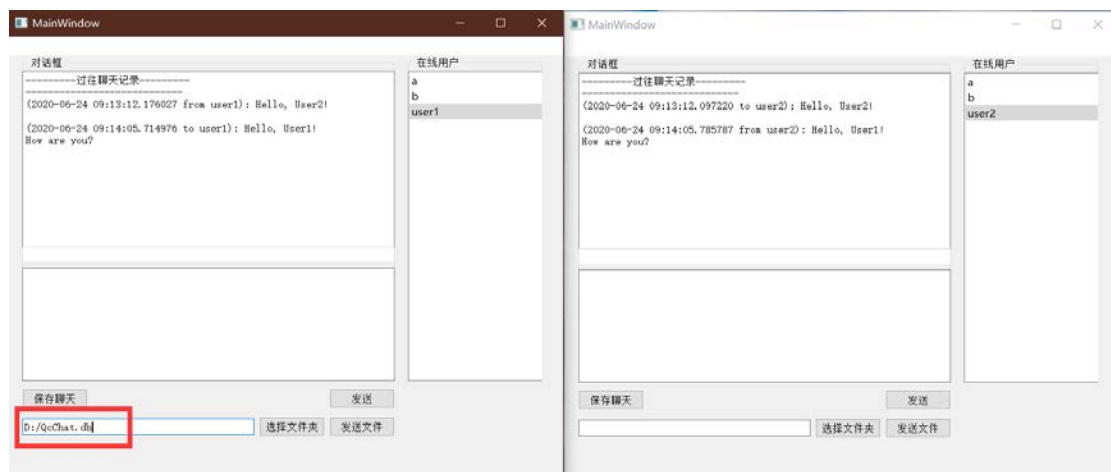


在 user1 与 user2 通信的同时, 若 user1 向 user2 发送消息, 则先发送一个挑战应答。即 user1 发送一串随机数给 user2, user2 用 hash 函数进行计算后反馈给 user1, user1 将函数值进行比对, 准确无误后证明 user2 身份正确, 故开始发送消息。

至此, 消息的加密传输得以实现。

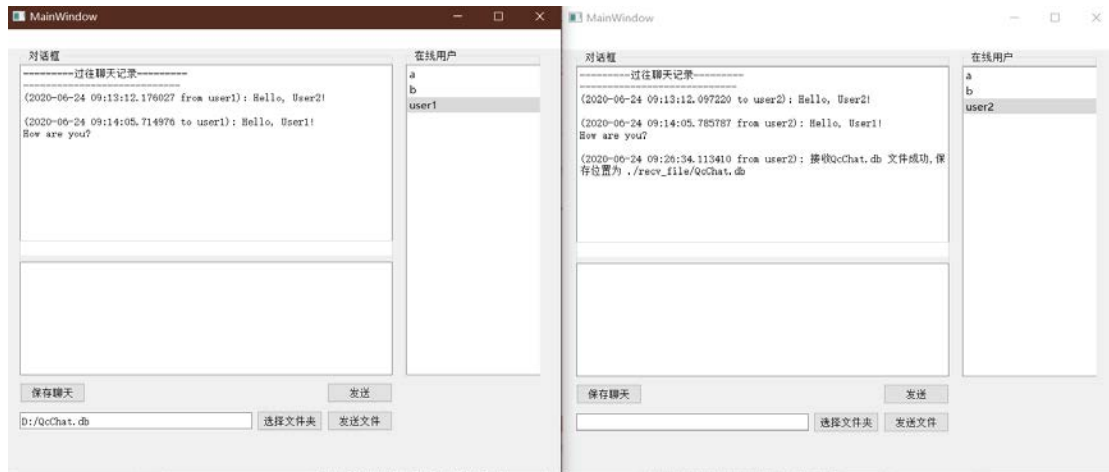
## 5.4 文件的加密传输

我们选择一个文件进行传输



如图所示, 我们使用一个 d 盘下的一个数据库文件进行传输。

单击发送按钮, 文件 user1 首先随机生成一个 DES 密钥, 然后将文件压缩后用 DES 加密, 并用自己的私钥加密 DES 密钥后通过 socket 端口发送给 user2, user2 用 user1 的公钥解密后获得 des 密钥, 用 des 密钥解密文件后获得文件, 至此文件传输成功。



## 6 代码展示

### 6.1 代码逻辑结构

下面是代码逻辑结构图

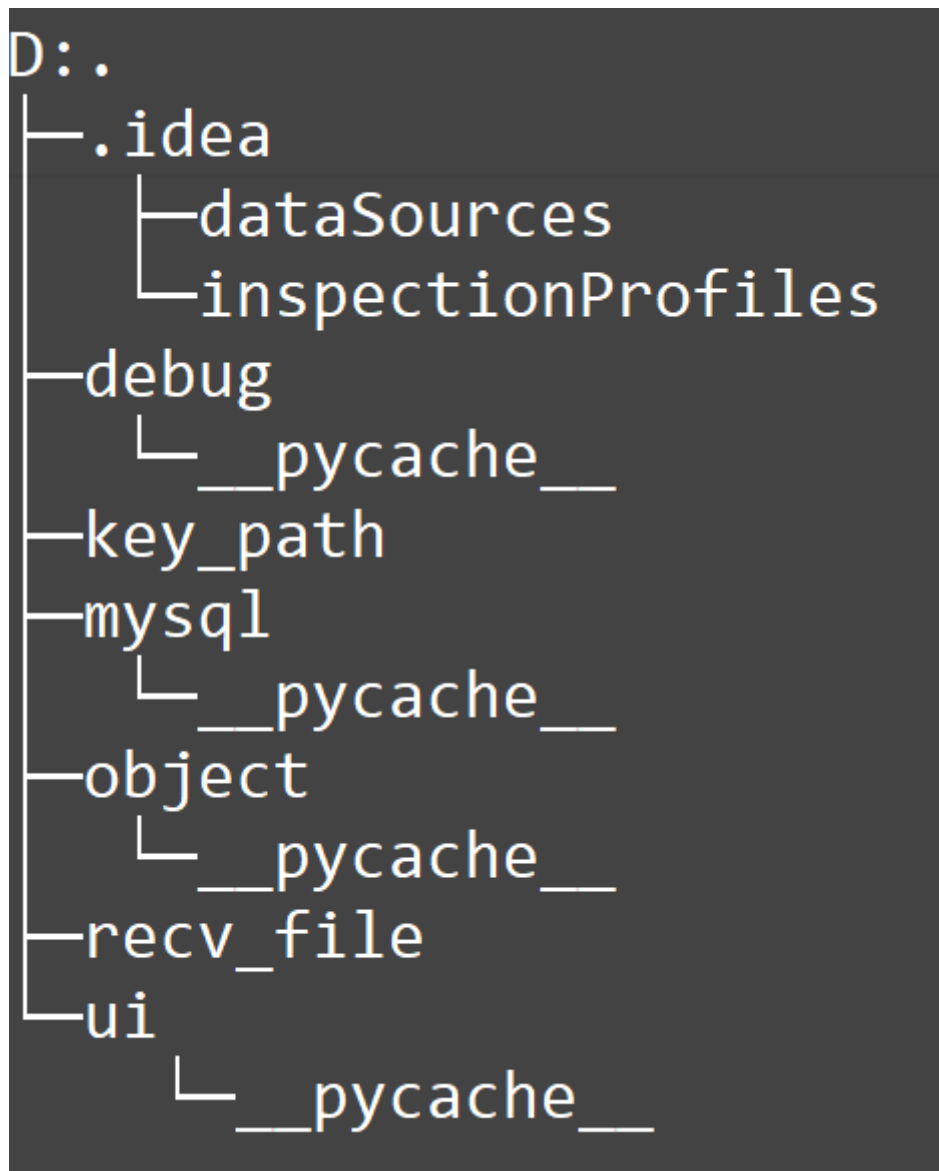


图 7：代码结构图

代码共有 6 个子文件夹。

首先根目录下有 3 个文件：

**Main.py**：第一个客户端启动主程序

**Main2.py**：第二个客户端启动主程序

**QcChat.db:sqlite3** 数据库文件，存储用户用户名密码和聊天信息。

第一个子文件夹 **debug** 下存放了一些编码是调试的 **py** 代码，其目录结构如下：

```
D:.\n  clientTest.py\n  main2.py\n  main3.py\n  muti_mysql.py\n  QcChat.db\n  serverTest.py\n  __init__.py\n\n__pycache__\n    libclient.cpython-36.pyc\n    libserver.cpython-36.pyc\n    init .cpython-36.pyc
```

图 8: debug 文件夹目录结构

Key\_path 文件夹存储了不同用户的公钥和私钥。

Mysql 文件夹下存储连接 sqlite 数据库所需代码文件:

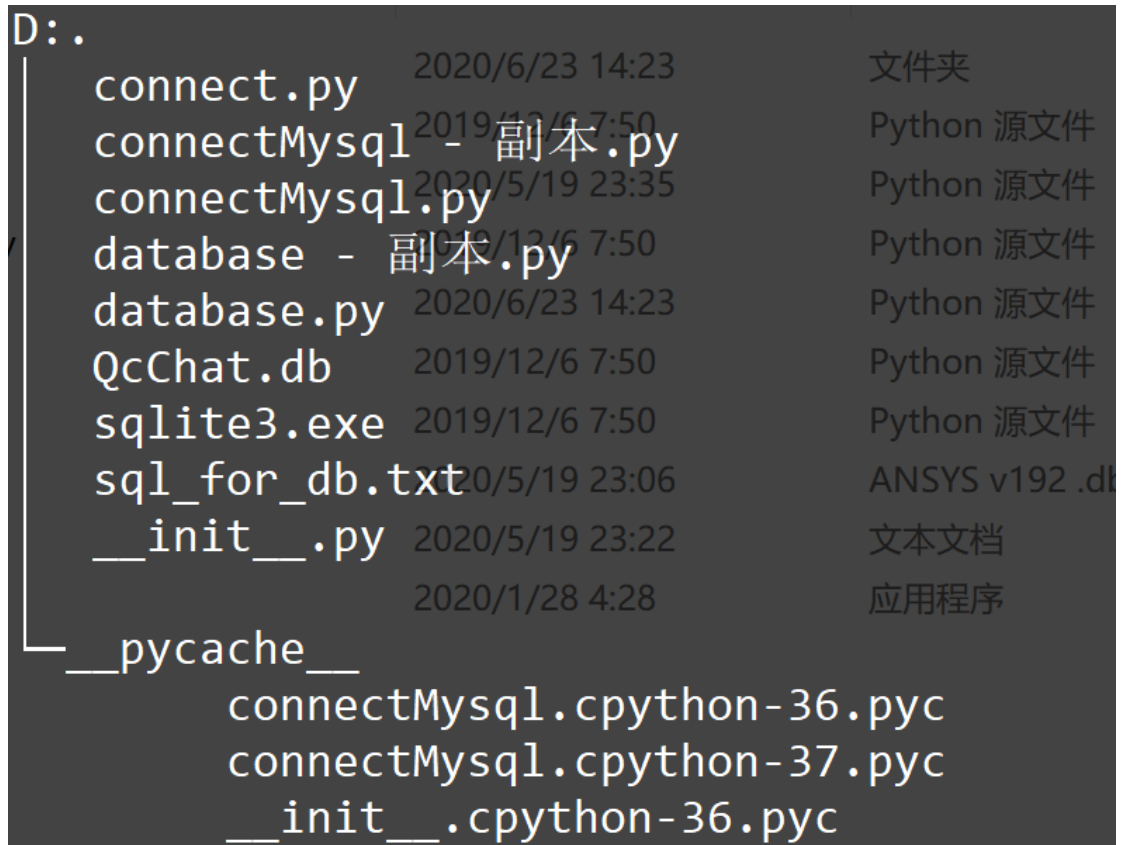


图 9: mysql 文件夹结构

Connect.py 中存放实现数据库增删改查的所有代码;

Database.py 存储了连接数据库的接口。

接下来是 object 文件夹:

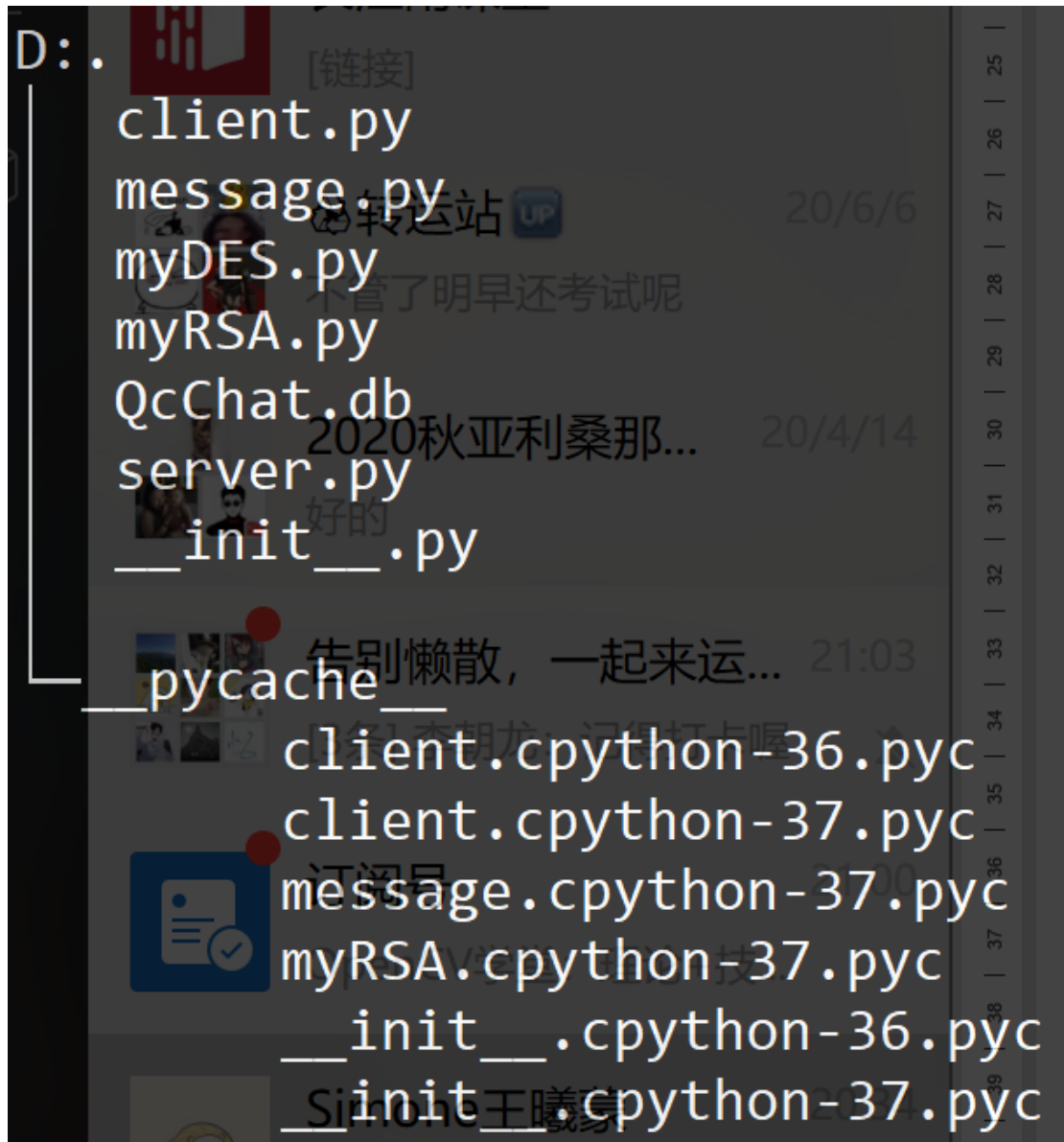


图 10: object 文件夹结构

Client.py 文件中定义了客户端后台操作, 包括用户登录, 收发数据, 查看连接状态等等;

Message.py 文件中定义了数据的处理操作, 包括数据的 RSA 加密和 DES 加密等;

MyDES.py 文件中实现了 DES 加解密算法

MyRSA.py 文件中实现了 RSA 密钥生成, 加解密算法

Server.py 文件中定义了服务器的后台进程代码, 包括消息的转发等;

最后是 ui 文件夹:

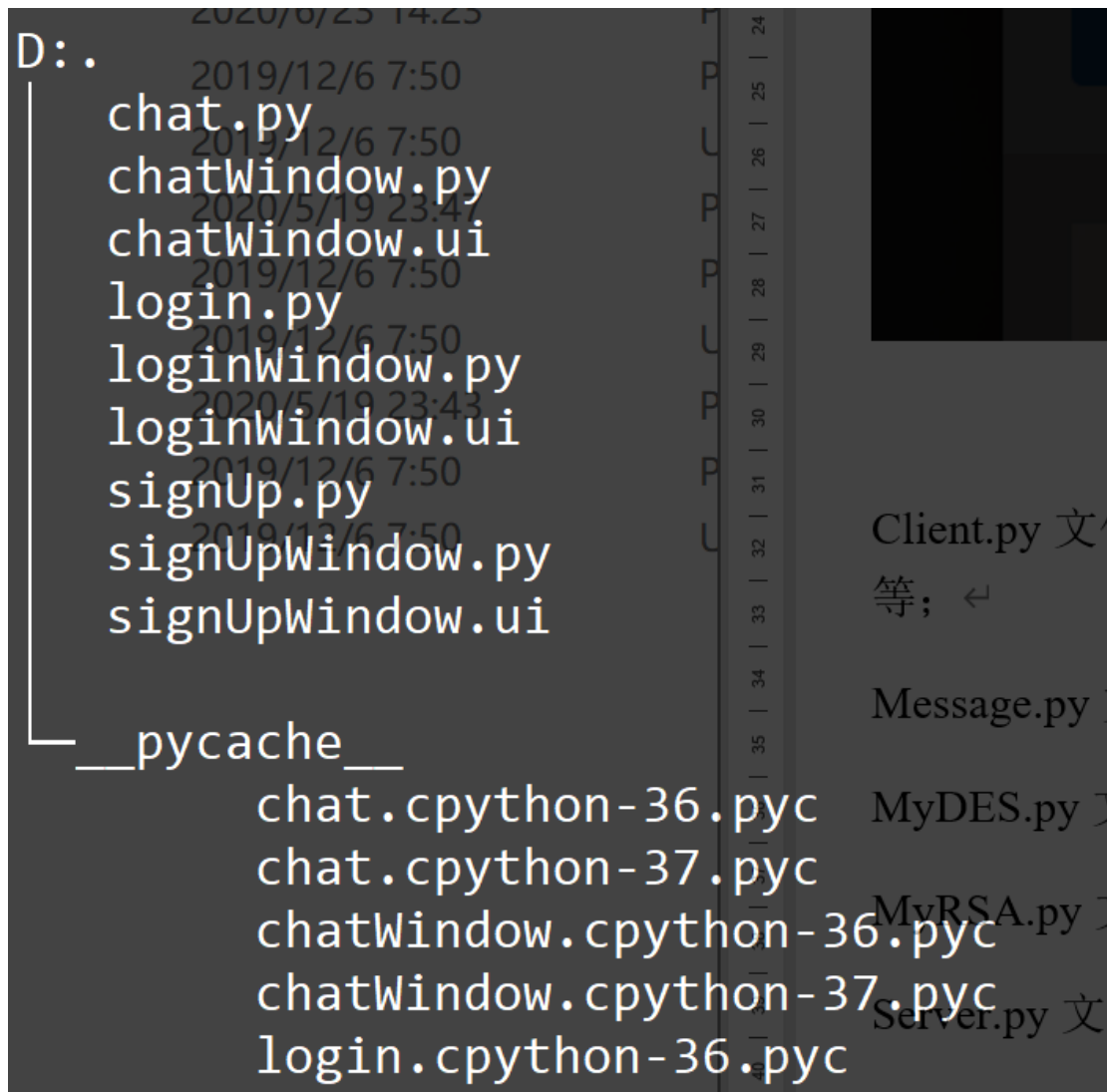


图 11: ui 文件夹结构

Chat.py: 实现聊天功能

chatWindow.py, chatWindow.ui 定义用户界面

login.py: 实现登录功能

loginWindow.py, loginWindow.ui: 实现登录窗口

signUp.py: 实现注册功能

signUpWindow.py, signUpWindow.ui: 实现注册窗口

## 6.2 部分代码展示

### 6.2.1 数据库设计

```
CREATE TABLE users(
```

```
id VARCHAR(20) PRIMARY KEY NOT NULL,

password VARCHAR(20),

ip VARCHAR(20),

port INT(10),

alive INT(1)

);

INSERT INTO users VALUES( 'test', 'test', '111.111.111.111', 223, 0);
```

## 6.2.2 Mysql 文件夹代码展示

### 6.2.2.1 Connect.py

```
#导入 SQLite 驱动:

import sqlite3, pdb

#连接到 SQLite 数据库

#数据库文件是 test.db, 不存在, 则自动创建

conn = sqlite3.connect('QcChat.db')

#创建一个 cursor:

cursor = conn.cursor()

#执行一条 SQL 语句: 创建 user 表

# cursor.execute('create table user(id varchar(20) primary key, name varchar(20))')

#插入一条记录:

cursor.execute('''CREATE TABLE COMPANY

        (ID INT PRIMARY KEY     NOT NULL,

        NAME           TEXT     NOT NULL,

        AGE            INT       NOT NULL,

        ADDRESS        CHAR(50),

        SALARY         REAL);''')

conn.commit()

pdb.set_trace()
```



```

cursor.execute('select id from users;')

result = cursor.fetchall()

print(result)

#通过 rowcount 获得插入的行数:

# print(cursor.rowcount) #reusult 1

#关闭 Cursor:

cursor.close()

#提交事务:

conn.commit()

#关闭 connection:

conn.close()

```

#### 6.2.2.2 connetMysql.py

```

import sqlite3

import pdb, os

# print(os.getcwd())

class ConnetMysql:

    def __init__(self):

        self.conn = None

    def get_conn(self):

        try:

            if self.conn is None:

                self.conn = sqlite3.connect(r'D:\GitProject\Internet-Security-ExpCo
de\实验\点到点加密通讯\Chat-Socket-Python-master\QcChat.db')

            except Exception as e:

                print("Error: {}".format(e))

                # pdb.set_trace()

```

```
def vericate(self, input_id, input_password):  
    self.get_conn()  
    cursor = self.conn.cursor()  
    sql = "select id, password from users"  
    cursor.execute(sql)  
    id_and_password = cursor.fetchall()  
    for id, password in id_and_password:  
        if id == input_id and password == input_password:  
            update_sql = "update users set alive = 1 where id = %s and password  
= %s" %(id, password)  
            cursor.execute(update_sql)  
            print("vericate successfully")  
            self.cut_conn()  
            return True  
    self.cut_conn()  
    return False  
  
def insert(self, sql, params=None):  
    try:  
        self.get_conn()  
        cursor = self.conn.cursor()  
        if params is not None:  
            cursor.execute(sql, params)  
        else:  
            cursor.execute(sql)  
        self.conn.commit()  
  
        self.cut_conn()  
        print("insert successfully")
```

```
except Exception as e:

    print("Error: {}".format(e))

    print("Error: {}".format(sql))


def delete(self, sql, params=None):

    try:

        self.get_conn()

        cursor = self.conn.cursor()

        if params is not None:

            cursor.execute(sql, params)

        else:

            cursor.execute(sql)

        self.conn.commit()

        self.cut_conn()

        print("delete successfully")

    except Exception as e:

        print("Error: {}".format(e))

        print("Error: {}".format(sql))


def search(self, sql, params=None):

    try:

        self.get_conn()

        cursor = self.conn.cursor()

        if params is not None:

            cursor.execute(sql, params)
```

```
        else:

            cursor.execute(sql)

            # self.conn.commit()

            result = cursor.fetchall()

            # print(result)

            self.cut_conn()

        return result

    except Exception as e:

        print("Error: {}".format(e))

        print("Error: {}".format(sql))

        return None

def update(self, sql, params=None):

    try:

        self.get_conn()

        cursor = self.conn.cursor()

        if params is not None:

            cursor.execute(sql, params)

        else:

            cursor.execute(sql)

            self.conn.commit()

        self.cut_conn()

        print("update successfully")

    except Exception as e:

        print("Error: {}".format(e))
```

```
        print("Error: {}".format(sql))

def save_chat_record(self, record, row, id):
    try:
        self.get_conn()

        cursor = self.conn.cursor()

        if len(record) != 0:
            for r in record[row:]:
                date = r[1:11]

                time = r[12:27]

                params = (date, time, r)

                print(params)

                cursor.execute("insert into {} values(%s, %s, %s)".format(id+"_
record"), params)

                row += 1

            self.conn.commit()

        self.cut_conn()

        return row

    except Exception as e:
        print("Error: {}".format(e))

        print("Error: record")

        return

def get_record(self):
    try:
        self.get_conn()

        cursor = self.conn.cursor()

    except Exception as e:
        print("Error: {}".format(e))
```

```
        print("Error: get record")

    def sign_out(self, addr):
        try:
            self.get_conn()

            id = self.search("select id from users where ip = %s and port = %s", addr)[0][0]

            self.update("update users set alive = 0, ip = '0', port = 0 where id = %s", id)

            print("sign out successfully")

            return id

        except Exception as e:
            print("Error: {}".format(e))
            print("Error: {}".format(addr))

    def create_table(self, sql, params=None):
        self.get_conn()

        cursor = self.conn.cursor()

        if params is not None:
            cursor.execute(sql, params)

        else:
            cursor.execute(sql)

        self.cut_conn()

        print("create table successfully")

    def cut_conn(self):
        try:
            self.conn.close()

            self.conn = None
```

```
except Exception as e:

    print("Error: {}".format(e))
```

### 6.2.2.3 database.py

```
import pymysql

def db_execute(sql, params):
    try:
        conn = pymysql.connect(host="localhost",
                                user="root",
                                password="mysqlfgh.00",
                                db="QcChat",
                                port=3306,
                                charset="utf8")

        cursor = conn.cursor()
        cursor.execute(sql, params)

    except Exception as e:
        print("Error: {}".format(e))
```

## 6.2.3 object 文件夹代码展示

### 6.2.3.1 client.py

```
import datetime

import io

import json

import socket

import time

import threading

from object import myRSA

import base64

from object.message import Message
```

```
class Client(object):

    def __init__(self, chat_window, server_host="127.0.0.1", server_port=50000):

        self.chat_window = chat_window

        self._server_host = server_host

        self._server_port = server_port

        self.socket = None

        self.t = None

        self._start_connections()

    def _start_connections(self):

        server_addr = (self._server_host, self._server_port)

        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        self.socket.connect(server_addr)

        self.t = threading.Thread(target=self.read_from_server, daemon=True)

        self.t.start()

    def read_from_server(self):

        try:

            while True:

                recv = self.socket.recv(2048)

                self._process_recv(recv)

                time.sleep(0.5)

        except Exception as e:

            print(e)

            self.sign_out()
```



```
def _process_recv(self, recv_data):

    try:

        print("recv", repr(recv_data))

        data_dict = self._json_decode(recv_data)

        if recv_data is not None:

            action = data_dict.get("action")

            if action == "transmit":

                from_id = data_dict.get("from_id")

                value = data_dict.get("value")

                sign = data_dict.get("sign")

                to_id = data_dict.get("to_id")

                # value = value.encode('utf-8')

                # sign = sign.encode("utf-8")

                #先拿自己私钥解密文件，再拿发送方公钥解密签名

                from_id_RSA_dic = myRSA.generate_RSAkey_with_sig(2048, from_id)

                to_id_RSA_dic = myRSA.generate_RSAkey_with_sig(2048, to_id)

                value = base64.b64decode(value.encode("utf-8"))

                sign = base64.b64decode(sign.encode("utf-8"))

                print("decoding complete")

                # print("value");print(value)

                # print("sign");print(sign)

                value = myRSA.decrypt_with_rsa(value, to_id_RSA_dic['pri_key', ])

                print("decoding value completes")

                verified = myRSA.to_verify_with_public_key(sign, value, from_id_RSA_dic['pub_key'])

                if not verified: print("Signature not verified!")

                print("recv: {} from {}".format(value, from_id))

                browser_msg = "({} from {}): {}".format(datetime.datetime.now(), from_id, value).rstrip()
```

```

        self.chat_window.textBrowser.append(browser_msg + "\n")

        self.chat_window.chat_record.append(browser_msg)

    elif action == "file":

        self.socket.send("ready".encode())

        self._recv_file(data_dict)

    elif action == "login":

        login_id = data_dict.get("from_id")

        self.chat_window.listWidget.addItem(login_id)

    elif action == "out":

        out_id = data_dict.get("from_id")

        for i in range(self.chat_window.listWidget.count()):

            if out_id == self.chat_window.listWidget.item(i).text():

                self.chat_window.listWidget.takeItem(i)

                break

        else:

            print(f'Error: invalid action "{action}"')

except Exception as e:

    print(e)

def _recv_file(self, data_dict):

    file_size = data_dict.get("value")

    file_name = data_dict.get("other")

    from_id = data_dict.get("from_id")

    to_id = data_dict.get("to_id")

    receive_size = 0

    res = b""

    while receive_size < file_size:

        data = self.socket.recv(2048)

        receive_size += len(data)

        res += data

```

```

new_file_name = "./recv_file/"+file_name

f = open(new_file_name, "wb")

f.write(res)

f.close()

browser_msg = "({} from {}): 接收{} 文件成功,保存位置为 {}".format(
    datetime.datetime.now(), from_id, file_name, new_file_name
).rstrip()

self.chat_window.textBrowser.append(browser_msg + "\n")

self.chat_window.chat_record.append(browser_msg)

self.socket.send(Message("transmit", to_id, from_id, "{} 文件已收到".format(
    file_name)).content_bytes)

def _json_decode(self, json_bytes, encoding="utf-8"):
    try:
        tiow = io.TextIOWrapper(
            io.BytesIO(json_bytes), encoding=encoding, newline=""
        )
        obj = json.load(tiow)
        tiow.close()
        return obj
    except Exception as e:
        print(e)
        return

def sign_out(self):
    self.socket.close()

```

### 6.2.3.2 message.py

```

import sys

import json

import struct

```

```

from object import myRSA

class Message(object):

    def __init__(self, action, from_id, to_id, value, other="", type="text/json",
encoding="utf-8", sign=None):

        self._action = action

        self._from_id = from_id

        self._to_id = to_id

        self._value = value###msg 内容需要加密

        # self._from_id_RSA_dic = myRSA.generate_RSAkey_with_sig(2048, self._from_i
d)

        # self._to_id_RSA_dic = myRSA.generate_RSAkey_with_sig(2048, self._to_id)

        # self._RSAvalue = None

        # self._RSAsig = None

        self._sign = None

        if sign: self._sign = sign

        self._other = other

        self._content_type = type

        self._content_encoding = encoding

        self._content = self._get_content()

        self.content_bytes = self._json_encode(self._content, self._content_encodi
ng)

        print("Finish content_bytes")

        self.message = self._get_message()

    def _get_message(self):

        jsonheader = {

            "byteorder": sys.byteorder,

            "content-type": self._content_type,

```

```
        "content-encoding": self._content_encoding,
        "content-length": len(self.content_bytes),
    }

    jsonheader_bytes = self._json_encode(jsonheader, "utf-8")
    message_hdr = struct.pack(">H", len(jsonheader_bytes))
    message = message_hdr + jsonheader_bytes + self.content_bytes

    return message

def _get_content(self):

    if self._action == "transmit" or self._action == "file":

        return dict(
            action=self._action,
            from_id=self._from_id,
            to_id=self._to_id,
            value=self._value,
            sign=self._sign,
            other=self._other
        )

    elif self._action == "login" or self._action == "out":
        return dict(action=self._action, from_id=self._from_id)

    elif self._action == "search":
        return dict(action=self._action, value=self._value)

    else:
        # return bytes(self._action + self._value, encoding="utf-8")

        print("enter rs")

        rs = (self._action + self._value).decode("utf-8, ignore")
```

```
        print("complete rs")

    return rs

def _json_encode(self, obj, encoding):
    return json.dumps(obj, ensure_ascii=False).encode(encoding)
```

### 6.2.3.3 myDES.py

```
import Crypto.Random as Random

from Cryptodome.Cipher import DES

import binascii

def generate_DES_key(key_len=8):
    return Random.get_random_bytes(key_len)

def encrypt_with_des(plain_text, key):
    des = DES.new(key, DES.MODE_ECB)

    plain_text = plain_text + (8 - (len(plain_text) % 8)) * '='

    encrypto_text = des.encrypt(plain_text.encode())

    encrypto_text = binascii.b2a_hex(encrypto_text)

    return encrypto_text

def decrypt_with_des(secret_byte_obj, key):
    des = DES.new(key, DES.MODE_ECB)

    decrypto_text = binascii.a2b_hex(secret_byte_obj)

    decrypto_text = des.decrypt(decrypto_text)

    return decrypto_text

if __name__ == "__main__":
```

```
# 这是密钥

des_key = Random.get_random_bytes(8)

key = b'abcdefgh' # key 需为 8 字节长度.

# 需要去生成一个 DES 对象

des = DES.new(key, DES.MODE_ECB)

# 需要加密的数据

text = 'python spider!'*10000 # 被加密的数据需要为 8 字节的倍数.

text = text + (8 - (len(text) % 8)) * '='

print(text)

# 加密的过程

encrypto_text = des.encrypt(text.encode())

encrypto_text = binascii.b2a_hex(encrypto_text)

# print(encrypto_text)

decrypto_text = binascii.a2b_hex(encrypto_text)

# print(decrypto_text)

decrypto_text = des.decrypt(decrypto_text)

print(decrypto_text)
```

#### 6.2.3.4 myRSA.py

```
import base64

from Crypto.PublicKey import RSA

import Crypto.Signature.PKCS1_v1_5 as sign_PKCS1_v1_5 # 用于签名/验签

from Crypto.Cipher import PKCS1_v1_5 # 用于加密

from Crypto import Hash

import Crypto.Random as Random

import datetime, os
```

```
def generate_RSA_key(key_len=2048):

    public_key_path = 'public_key.pem'
    private_key_path = "private_key.pem"

    x = RSA.generate(2048)

    # x = RSA.generate(2048, Random.new().read)    #也可以使用伪随机数来辅助生成

    s_key = x.exportKey("PEM")    # 私钥

    g_key = x.publickey().exportKey("PEM")    # 公钥

    print(s_key)

    # write_RSA_key(public_key_path, g_key)

    # write_RSA_key(private_key_path, s_key)

    # 实现 RSA 非对称加解密

    my_private_key = get_RSA_key(private_key_path)    # 私钥

    my_public_key = get_RSA_key(public_key_path)    # 公钥

    return my_public_key, my_private_key
```

##### 使用公钥 - 私钥对信息进行“加密” + “解密” #####

'''

作用：对信息进行公钥加密，私钥解密。

应用场景：

A 想要加密传输一份数据给 B，担心使用对称加密算法易被他人破解（密钥只有一份，一旦泄露，则数据泄露），故使用非对称加密。

信息接收方可以生成自己的密钥对，即公私钥各一个，然后将公钥发给他人，私钥自己保留。

A 使用公钥加密数据，然后将加密后的密文发送给 B，B 再使用自己的私钥进行解密，这样即使 A 的公钥和密文均被第三方得到，

第三方也要知晓私钥和加密算法才能解密密文，大大降低数据泄露风险。

'''

```
def generate_RSAkey_with_sig(key_len=2048, owner='Anonymous'):
```



```
if owner == 'Anonymous': print("Signature with Anonymous!")

key_path = r'key_path/'

if not os.path.exists(key_path):

    os.mkdir(key_path)

public_key_path = key_path + owner + '_public_key.pem'

private_key_path = key_path + owner + "_private_key.pem"

if not(os.path.exists(public_key_path) and os.path.exists(private_key_path)):

    print("RSA files don't exist, start generating... ")

    x = RSA.generate(key_len)

    # x = RSA.generate(2048, Random.new().read)  #也可以使用伪随机数来辅助生成

    s_key = x.exportKey("PEM")  # 私钥

    g_key = x.publickey().exportKey("PEM")  # 公钥

    # print(s_key)

    write_RSA_key(public_key_path, g_key)

    write_RSA_key(private_key_path, s_key)

    print("RSA files generating completes.")

# 实现 RSA 非对称加解密

my_private_key = get_RSA_key(private_key_path)  # 私钥

my_public_key = get_RSA_key(public_key_path)  # 公钥

return {

    "name": owner,

    "pri_key": my_private_key,

    "pub_key": my_public_key,

    "date": str(datetime.datetime.now())

}

def encrypt_with_rsa(plain_text, my_public_key):

    # 先公钥加密

    cipher_pub_obj = PKCS1_v1_5.new(RSA.importKey(my_public_key))
```

```
_secret_byte_obj = cipher_pub_obj.encrypt(plain_text.encode())

return _secret_byte_obj


def decrypt_with_rsa(_secret_byte_obj, my_private_key):
    # 后私钥解密

    # _secret_byte_obj = bytes(_secret_byte_obj, encoding='utf-8')

    cipher_pri_obj = PKCS1_v1_5.new(RSA.importKey(my_private_key))

    _byte_obj = cipher_pri_obj.decrypt(_secret_byte_obj, Random.new().read)

    plain_text = _byte_obj.decode()

    return plain_text


def executer_without_signature(pub_key, pri_key):
    # 加解密验证

    text = "I love CA!"

    print("TYPE:", type(encrypt_with_rsa(text, pub_key)))

    assert text == decrypt_with_rsa(encrypt_with_rsa(text, pub_key), pri_key)

    print("rsa test success! ")
```

##### 使用私钥 - 公钥对信息进行“签名” + “验签” #####

'''

作用：对解密后的文件的完整性、真实性进行验证（繁琐但更加保险的做法，很少用到）

应用场景：

A 有一私密文件欲加密后发送给 B, 又担心因各种原因导致 B 收到并解密后的文件并非完整、真实的原文件（可能被篡改或丢失一部分），

所以 A 在发送前对原文件进行签名，将[签名和密文]一同发送给 B 让 B 收到后用做一下文件的[解密 + 验签]，

均通过后-方可证明收到的原文件的真实性、完整性。

'''

```
def to_sign_with_private_key(plain_text, my_private_key):  
    # 私钥签名  
  
    signer_pri_obj = sign_PKCS1_v1_5.new(RSA.importKey(my_private_key))  
  
    rand_hash = Hash.SHA256.new()  
  
    rand_hash.update(plain_text.encode())  
  
    signature = signer_pri_obj.sign(rand_hash)  
  
    return signature  
  
  
def to_verify_with_public_key(signature, plain_text, my_public_key):  
    # 公钥验签  
  
    verifier = sign_PKCS1_v1_5.new(RSA.importKey(my_public_key))  
  
    _rand_hash = Hash.SHA256.new()  
  
    _rand_hash.update(plain_text.encode())  
  
    verify = verifier.verify(_rand_hash, signature)  
  
    return verify # true / false  
  
  
def executer_with_signature(pub_key, pri_key):  
    # 签名/验签  
  
    text = "I love CA!"  
  
    print("to_sign_with_private_key(text, pri_key)", type(to_sign_with_private_key  
(text, pri_key)))
```

```

    assert to_verify_with_public_key(to_sign_with_private_key(text, pri_key), text, pub_key)

    print("rsa Signature verified!")

def get_RSA_key(key_dir):

    with open(key_dir, 'rb') as f:

        key = f.read()

    return key

def write_RSA_key(key_dir, key):

    with open(key_dir, 'wb') as f:

        f.write(key)

if __name__ == '__main__':

    pub_key, pri_key = generate_RSA_key()

    executer_without_signature(pub_key, pri_key) # 只加密不签名

    executer_with_signature(pub_key, pri_key) # 只签名不加密

    # 二者结合食用更佳
'''

```

如果是加密的同时又要签名，这个时候稍微有点复杂。

- 1、发送者和接收者需要各持有一对公私钥，也就是 4 个钥匙。
- 2、接收者的公私钥用于机密信息的加解密
- 3、发送者的公私钥用于机密信息的签名/验签

4、接收者和发送者都要提前将各自的[公钥]告知对方。

'''

#### 6.2.3.5 Server.py

```
import io
import json
import socket
import selectors
import time
import types
import sys, os

current_path = os.path.dirname(os.path.abspath(__file__)) + "/"
sys.path.append(current_path)

from mysql.connectMysql import ConnetMysql
from object.message import Message

class Server():

    def __init__(self, host='127.0.0.1', port=50000):
        self.host = host
        self.port = port
        self.sel = selectors.DefaultSelector()
        self.lsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.id_to_sock = dict()
        self.db = ConnetMysql()
        self.start()

    def start(self):
        self.lsock.bind((self.host, self.port))
```

```
self.lsock.listen()

print("listening on", (self.host, self.port))

self.lsock.setblocking(False)

self.sel.register(self.lsock, selectors.EVENT_READ, data=None)


def main(self):

    try:

        while True:

            events = self.sel.select(timeout=None)

            for key, mask in events:

                # print(mask)

                if key.data is None:

                    # print("Enter accept_wrapper")

                    self.accept_wrapper(key.fileobj)

                else:

                    # print("Enter service_connection")

                    print(key.fileobj)

                    self.service_connection(key, mask)

            except KeyboardInterrupt:

                print("caught keyboard interrupt, exiting")

            except WindowsError:

                print("WindowsError")

        finally:

            self.sel.close()


def accept_wrapper(self, sock):

    conn, addr = sock.accept() # Should be ready to read
```

```
print("accepted connection from", addr)

conn.setblocking(False)

data = types.SimpleNamespace(addr=addr, inb=b"", outb=b"")

events = selectors.EVENT_READ | selectors.EVENT_WRITE

self.sel.register(conn, events, data=data)

self.db.update("update users set ip = '%s', port = '%s' where alive = 1 and ip = '0' and port = 0"%(addr[0], addr[1]))

id = self.db.search("select id from users where ip = '%s' and port = '%s' and alive = 1"%(addr[0], addr[1]))

print("id: ", id)

if id != None and id != []:

    print("id", id)

    self.id_to_sock[id[0][0]] = conn

    self._send_login_message_to_other_clients(id[0][0])

def service_connection(self, key, mask):

    try:

        sock = key.fileobj

        data = key.data

        if mask & selectors.EVENT_READ:

            # print("b")

            recv_data = sock.recv(2048) # Should be ready to read

            print(recv_data.type)

            if recv_data:

                print("enter _process_recv")

                self._process_recv(recv_data, data)

            else:

                self._process_client_sign_out(sock, data)
```

```

except WindowsError:

    print("service_connection WindowsError")

    sock = key.fileobj

    peer_ip, peer_port = sock.getpeername()

    print("service_connection WindowsError")

    update_sql = "update users set ip = '%s', port = '%s' where alive = 0 a
nd ip = '0' and port = 0" % (peer_ip, peer_port)

    print(update_sql)

    self.db.update(update_sql)


except Exception as e:

    print(e)


def _json_decode(self, json_bytes, encoding="utf-8"):

    tiow = io.TextIOWrapper(

        io.BytesIO(json_bytes), encoding=encoding, newline=""

    )

    obj = json.load(tiow)

    tiow.close()

    return obj


def _process_recv(self, recv_data, data):

    try:

        print("recv", repr(recv_data), "from", data.addr)

        data_dict = self._json_decode(recv_data)

        if recv_data is not None:

            action = data_dict.get("action")

            if action == "transmit":

```



```

        to_id = data_dict.get("to_id")

        sock = self.id_to_sock.get(to_id)

        sock.send(recv_data)

        print("send successfully")

    elif action == "file":

        self._transmit_file(recv_data, data_dict)

    elif action == "login" or action == "out":

        pass

    else:

        print(f'Error: invalid action "{action}"')

except Exception as e:

    print(e)

def _send_login_message_to_other_clients(self, id):

    message = Message("login", id, "", "").content_bytes

    for i in self.id_to_sock.keys():

        if i != id:

            self.id_to_sock[i].send(message)

def _process_client_sign_out(self, sock, data):

    print("closing connection to", data.addr)

    id = self.db.sign_out(data.addr)

    self.id_to_sock.pop(id)

    self.sel.unregister(sock)

    sock.close()

    for i in self.id_to_sock.keys():

        self.id_to_sock[i].send(Message("out", id, "", "").content_bytes)

def _transmit_file(self, recv_data, data_dict):

    to_id = data_dict.get("to_id")

```

```
from_id = data_dict.get("from_id")

from_sock = self.id_to_sock.get(from_id)

sock = self.id_to_sock.get(to_id)

file_size = data_dict.get("value")

sock.send(recv_data)

time.sleep(0.5)

ready_msg = sock.recv(2048)

if ready_msg.decode() == "ready":

    total_recv = 0

    res = b""

    while total_recv < file_size:

        file_data = from_sock.recv(2048)

        res += file_data

        total_recv += len(file_data)

    sock.sendall(res)

    time.sleep(0.5)

if __name__ == "__main__":

    server = Server()

    server.main()
```

## 6.2.4 ui 文件夹代码展示

### 6.2.4.1 chat.py

```
import os

from .chatWindow import Ui_MainWindow

from object.client import Client

from object.message import Message

from mysql.connectMysql import ConnetMysql

from object import myRSA
```

```
from PyQt5 import QtWidgets, QtGui

import datetime

import base64

class Chat(QtWidgets.QWidget, Ui_MainWindow):

    def __init__(self):

        super(Chat, self).__init__()

        self.setupUi(self)

        self.db = ConnetMysql()

        self.id = None

        self.client = None

        self.chat_record = []

        self.saved_record_row = 0

        self.pushButton.clicked.connect(self._send)

        self.pushButton_2.clicked.connect(self._save_chat_record)

        self.pushButton_3.clicked.connect(self._send_file)

        self.pushButton_4.clicked.connect(self._choose_directory)

    def login(self, id):

        self.id = id

        result = self.db.search("select id from users where alive = 1 and id != '%s' "%(self.id))

        print(result)

        if result != []:

            users_alive = list(zip(*result))[0]

            self.listWidget.addItem(users_alive)

            self.client = Client(self)

            self._init_browser()
```

```
def _send(self):

    msg = self.textEdit.toPlainText()

    users = self._get_users()

    if users == []:

        QtWidgets.QMessageBox.information(self, "警告", "请在右栏选择发送对象")

        return

    if msg == "":

        QtWidgets.QMessageBox.information(self, "警告", "输入栏为空")

        return

    for user in users:

        self._from_id_RSA_dic = myRSA.generate_RSAkey_with_sig(2048, self.id)

        self._to_id_RSA_dic = myRSA.generate_RSAkey_with_sig(2048, user)

        self._from_id = self.id

        self._to_id = user

        self._value = msg

        # 先拿自己私钥加密增加签名, 再拿对方公钥加密保护数据

        print("Start Encryption")

        # print("before encrypt", type(self._value))

        self._sign = myRSA.to_sign_with_private_key(self._value, self._from_id_RSA_dic["pri_key"])

        # print("Finish signature.")

        init_value = self._value

        self._value = myRSA.encrypt_with_rsa(self._value, self._to_id_RSA_dic["pub_key"])

        decode_value = myRSA.decrypt_with_rsa(self._value, self._to_id_RSA_dic["pri_key"])

        print("tesst", decode_value == init_value)

        # print("after encrypt", type(self._value))

        # self._value = self._value.decode('utf-8', 'ignore')

        print("value")
```

```

        print(self._value)

        print("sign")

        print(self._sign)

        self._value = base64.b64encode(self._value).decode("utf-8")

        # print("before sign", self._sign)

        # self._sign = self._sign.decode('utf-8', 'ignore')

        before_sign = self._sign

        self._sign = base64.b64encode(self._sign).decode("utf-8")

        # print(self._sign == self._sign.decode("utf-8").encode("utf-8"))

        # tmp = self._sign.encode('utf-8')

        # tmp = base64.b64decode(self._sign)

        # print("after sign", tmp==before_sign)

        print("Encryption Completes")

        byte_msg = Message("transmit", self.id, user, self._value, sign=self._sign).content_bytes

        print("complete message")

        self.client.socket.send(byte_msg)

        print("Transmit complete")

        browser_msg = "({} to {}): {}".format(datetime.datetime.now(), user, msg).rstrip()

        self.textBrowser.append(browser_msg + "\n")

        self.chat_record.append(browser_msg)

        self.textEdit.clear()

    def _send_file(self):

        file_path= self.lineEdit.text()

        if not os.path.isfile(file_path):

            QtWidgets.QMessageBox.information(self, "错误", "文件不存在, 请检查文件路径是否正确")

```

```

        return

    file_name = file_path.split("/")[-1]

    file_size = os.path.getsize(file_path)

    users = self._get_users()

    if users == []:

        QtWidgets.QMessageBox.information(self, "警告", "请在右栏选择发送对象")

        return

    f = open(file_path, "rb")

    content = f.read()

    f.close()

    for user in users:

        byte_file_header = Message("file", self.id, user, file_size, file_name).content_bytes

        self.client.socket.send(byte_file_header)

        self.client.socket.sendall(content)

    def _get_file_path_and_name(self):

        return "/home/agwave/PycharmProjects/socket/debug/libserver.py", "libserver.py"

    def _choose_directory(self):

        get_directory_path = QtWidgets.QFileDialog.getExistingDirectory(self,

                                                                           "选取文件所在文件夹",

                                                                           "/home/agwave/PycharmProjects/socket")

        if str(get_directory_path) != "":

            self.lineEdit.setText(str(get_directory_path))

            QtWidgets.QMessageBox.information(self, "提示", "文件夹地址已输入, 请继续输入文件名")

    def _sign_out(self):

```



```

QtWidgets.QMessageBox.Yes | QtWidgets.
s.QMessageBox.No,

QtWidgets.QMessageBox.No)

if reply == QtWidgets.QMessageBox.Yes:
    print("clicked ok button.")
    self._sign_out()
else:
    QCloseEvent.ignore()

```

#### 6.2.4.2 chatWindow.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'chatWindow.ui'
#
# Created by: PyQt5 UI code generator 5.12.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(791, 631)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.frame = QtWidgets.QFrame(self.centralwidget)
        self.frame.setGeometry(QtCore.QRect(10, 20, 771, 561))
        self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtWidgets.QFrame.Raised)

```



```
self.frame.setObjectName("frame")

self.groupBox = QtWidgets.QGroupBox(self.frame)

self.groupBox.setGeometry(QtCore.QRect(10, 10, 531, 291))

self.groupBox.setObjectName("groupBox")

self.textBrowser = QtWidgets.QTextBrowser(self.groupBox)

self.textBrowser.setGeometry(QtCore.QRect(0, 20, 531, 281))

self.textBrowser.setObjectName("textBrowser")

self.textBrowser_3 = QtWidgets.QTextBrowser(self.groupBox)

self.textBrowser_3.setGeometry(QtCore.QRect(530, 210, 211, 251))

self.textBrowser_3.setObjectName("textBrowser_3")

self.groupBox_2 = QtWidgets.QGroupBox(self.frame)

self.groupBox_2.setGeometry(QtCore.QRect(10, 280, 531, 191))

self.groupBox_2.setTitle("")

self.groupBox_2.setObjectName("groupBox_2")

self.textEdit = QtWidgets.QTextEdit(self.groupBox_2)

self.textEdit.setGeometry(QtCore.QRect(0, 30, 531, 161))

self.textEdit.setObjectName("textEdit")

self.groupBox_4 = QtWidgets.QGroupBox(self.frame)

self.groupBox_4.setGeometry(QtCore.QRect(560, 10, 191, 461))

self.groupBox_4.setObjectName("groupBox_4")

self.listWidget = QtWidgets.QListWidget(self.groupBox_4)

self.listWidget.setGeometry(QtCore.QRect(0, 20, 201, 441))

self.listWidget.setObjectName("listWidget")

self.horizontalLayoutWidget = QtWidgets.QWidget(self.frame)

self.horizontalLayoutWidget.setGeometry(QtCore.QRect(10, 480, 531, 33))

self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)

self.horizontalLayout.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout.setObjectName("horizontalLayout")

self.pushButton_2 = QtWidgets.QPushButton(self.horizontalLayoutWidget)
```

```

self.pushButton_2.setObjectName("pushButton_2")

self.horizontalLayout.addWidget(self.pushButton_2)

spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)

self.horizontalLayout.addItem(spacerItem)

self.pushButton = QtWidgets.QPushButton(self.horizontalLayoutWidget)

self.pushButton.setObjectName("pushButton")

self.horizontalLayout.addWidget(self.pushButton)

self.horizontalLayoutWidget_2 = QtWidgets.QWidget(self.frame)

self.horizontalLayoutWidget_2.setGeometry(QtCore.QRect(10, 520, 531, 33))

self.horizontalLayoutWidget_2.setObjectName("horizontalLayoutWidget_2")

self.horizontalLayout_2 = QtWidgets.QHBoxLayout(self.horizontalLayoutWidge
t_2)

self.horizontalLayout_2.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout_2.setObjectName("horizontalLayout_2")

self.lineEdit = QtWidgets.QLineEdit(self.horizontalLayoutWidget_2)

self.lineEdit.setObjectName("lineEdit")

self.horizontalLayout_2.addWidget(self.lineEdit)

self.pushButton_4 = QtWidgets.QPushButton(self.horizontalLayoutWidget_2)

self.pushButton_4.setObjectName("pushButton_4")

self.horizontalLayout_2.addWidget(self.pushButton_4)

self.pushButton_3 = QtWidgets.QPushButton(self.horizontalLayoutWidget_2)

self.pushButton_3.setObjectName("pushButton_3")

self.horizontalLayout_2.addWidget(self.pushButton_3)

# MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 791, 28))

self.menubar.setObjectName("menubar")

# MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

```

```
self.statusbar.setObjectName("statusbar")

# MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
    self.groupBox.setTitle(_translate("MainWindow", "对话框"))
    self.groupBox_4.setTitle(_translate("MainWindow", "在线用户"))
    self.pushButton_2.setText(_translate("MainWindow", "保存聊天"))
    self.pushButton.setText(_translate("MainWindow", "发送"))
    self.pushButton_4.setText(_translate("MainWindow", "选择文件夹"))
    self.pushButton_3.setText(_translate("MainWindow", "发送文件"))
```

#### 6.2.4.3 chatWindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
    <class>MainWindow</class>
    <widget class="QMainWindow" name="MainWindow">
        <property name="geometry">
            <rect>
                <x>0</x>
                <y>0</y>
                <width>791</width>
                <height>631</height>
            </rect>
        </property>
        <property name="windowTitle">
```

```
<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">
  <widget class="QFrame" name="frame">
    <property name="geometry">
      <rect>
        <x>10</x>
        <y>20</y>
        <width>771</width>
        <height>561</height>
      </rect>
    </property>
    <property name="frameShape">
      <enum>QFrame::StyledPanel</enum>
    </property>
    <property name="frameShadow">
      <enum>QFrame::Raised</enum>
    </property>
    <widget class="QGroupBox" name="groupBox">
      <property name="geometry">
        <rect>
          <x>10</x>
          <y>10</y>
          <width>531</width>
          <height>291</height>
        </rect>
      </property>
      <property name="title">
        <string>对话框</string>
      </property>
    </widget>
  </widget>
</widget>
```

```
<widget class="QTextBrowser" name="textBrowser">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>20</y>
            <width>531</width>
            <height>281</height>
        </rect>
    </property>
</widget>

<widget class="QTextBrowser" name="textBrowser_3">
    <property name="geometry">
        <rect>
            <x>530</x>
            <y>210</y>
            <width>211</width>
            <height>251</height>
        </rect>
    </property>
</widget>

</widget>

<widget class="QGroupBox" name="groupBox_2">
    <property name="geometry">
        <rect>
            <x>10</x>
            <y>280</y>
            <width>531</width>
            <height>191</height>
        </rect>
    </property>
```

```
<property name="title">

    <string/>

</property>

<widget class="QTextEdit" name="textEdit">

    <property name="geometry">

        <rect>

            <x>0</x>

            <y>30</y>

            <width>531</width>

            <height>161</height>

        </rect>

    </property>

</widget>

</widget>

<widget class="QGroupBox" name="groupBox_4">

    <property name="geometry">

        <rect>

            <x>560</x>

            <y>10</y>

            <width>191</width>

            <height>461</height>

        </rect>

    </property>

    <property name="title">

        <string>在线用户</string>

    </property>

    <widget class="QListWidget" name="listWidget">

        <property name="geometry">

            <rect>

                <x>0</x>
```

```

        <y>20</y>

        <width>201</width>

        <height>441</height>

    </rect>

</property>

</widget>

</widget>

<widget class="QWidget" name="horizontalLayoutWidget">
    <property name="geometry">
        <rect>

            <x>10</x>

            <y>480</y>

            <width>531</width>

            <height>33</height>

        </rect>

    </property>

    <layout class="QHBoxLayout" name="horizontalLayout">
        <item>

            <widget class="QPushButton" name="pushButton_2">

                <property name="text">

                    <string>保存聊天</string>

                </property>

            </widget>

        </item>

        <item>

            <spacer name="horizontalSpacer">

                <property name="orientation">

                    <enum>Qt::Horizontal</enum>

                </property>

                <property name="sizeHint" stdset="0">

```

```

        <size>

        <width>40</width>

        <height>20</height>

    </size>

</property>

</spacer>

</item>

<item>

    <widget class="QPushButton" name="pushButton">

        <property name="text">

            <string>发送</string>

        </property>

    </widget>

</item>

</layout>

</widget>

<widget class="QWidget" name="horizontalLayoutWidget_2">

    <property name="geometry">

        <rect>

            <x>10</x>

            <y>520</y>

            <width>531</width>

            <height>33</height>

        </rect>

    </property>

    <layout class="HBoxLayout" name="horizontalLayout_2">

        <item>

            <widget class="QLineEdit" name="lineEdit"/>

        </item>

        <item>

```



```

    <widget class="QPushButton" name="pushButton_4">
        <property name="text">
            <string>选择文件夹</string>
        </property>
    </widget>
</item>
<item>
    <widget class="QPushButton" name="pushButton_3">
        <property name="text">
            <string>发送文件</string>
        </property>
    </widget>
</item>
</layout>
</widget>
</widget>
</widget>
<widget class="QMenuBar" name="menubar">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>791</width>
            <height>28</height>
        </rect>
    </property>
</widget>
<widget class="QStatusBar" name="statusbar"/>
</widget>
<resources/>

```

```
<connections/>
```

```
</ui>
```

#### 6.2.4.4 Login.py

```
from .loginWindow import Ui_MainWindow

from mysql.connectMysql import ConnetMysql

from PyQt5 import QtWidgets

class Login(QtWidgets.QWidget, Ui_MainWindow):

    def __init__(self, chat_window, signup_window):

        super(Login, self).__init__()

        self.setupUi(self)

        self.pushButton.clicked.connect(self.login)

        self.pushButton_2.clicked.connect(self.sign_up)

        self.chat_window = chat_window

        self.signup_window = signup_window

        self.db = ConnetMysql()

    def login(self):

        id, password = self.lineEdit.text(), self.lineEdit_2.text()

        is_user = False

        ids_and_passwords = self.db.search("select id, password from users")

        for i, p in ids_and_passwords:

            if i == id and p == password:

                is_user = True

                break
```

```

        if is_user:

            self.db.update("update users set alive = 1 where id = '%s' and password = '%s'" % (id, password))

            self.chat_window.show()

            self.chat_window.login(id)

            self.setHidden(True)

        else:

            QtWidgets.QMessageBox.information(self, "警告", "账号或密码错误")

    def sign_up(self):

        self.signup_window.show()

```

#### 6.2.4.5 loginWindow.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'loginWindow.ui'
#
# Created by: PyQt5 UI code generator 5.12.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(800, 600)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

```

```
self.centralwidget.setObjectName("centralwidget")

self.groupBox = QtWidgets.QGroupBox(self.centralwidget)

self.groupBox.setGeometry(QtCore.QRect(50, 30, 701, 481))

self.groupBox.setObjectName("groupBox")

self.gridLayoutWidget = QtWidgets.QWidget(self.groupBox)

self.gridLayoutWidget.setGeometry(QtCore.QRect(210, 100, 271, 201))

self.gridLayoutWidget.setObjectName("gridLayoutWidget")

self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)

self.gridLayout.setContentsMargins(0, 0, 0, 0)

self.gridLayout.setObjectName("gridLayout")

self.label_2 = QtWidgets.QLabel(self.gridLayoutWidget)

self.label_2.setObjectName("label_2")

self.gridLayout.addWidget(self.label_2, 1, 0, 1, 1)

self.label = QtWidgets.QLabel(self.gridLayoutWidget)

self.label.setObjectName("label")

self.gridLayout.addWidget(self.label, 0, 0, 1, 1)

self.lineEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)

self.lineEdit.setObjectName("lineEdit")

self.gridLayout.addWidget(self.lineEdit, 0, 1, 1, 1)

self.lineEdit_2 = QtWidgets.QLineEdit(self.gridLayoutWidget)

self.lineEdit_2.setObjectName("lineEdit_2")

self.gridLayout.addWidget(self.lineEdit_2, 1, 1, 1, 1)

self.horizontalLayoutWidget = QtWidgets.QWidget(self.groupBox)

self.horizontalLayoutWidget.setGeometry(QtCore.QRect(230, 340, 231, 33))

self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)

self.horizontalLayout.setContentsMargins(0, 0, 0, 0)

self.horizontalLayout.setObjectName("horizontalLayout")

self.pushButton = QtWidgets.QPushButton(self.horizontalLayoutWidget)

self.pushButton.setObjectName("pushButton")
```

```
self.horizontalLayout.addWidget(self.pushButton)

self.pushButton_2 = QtWidgets.QPushButton(self.horizontalLayoutWidget)

self.pushButton_2.setObjectName("pushButton_2")

self.horizontalLayout.addWidget(self.pushButton_2)

# MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 28))

self.menubar.setObjectName("menubar")

# MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

# MainWindow.setStatusBar(self.statusbar)


self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)


def retranslateUi(self, MainWindow):

    _translate = QtCore.QCoreApplication.translate

    MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))

    self.groupBox.setTitle(_translate("MainWindow", "登录"))

    self.label_2.setText(_translate("MainWindow", "密码: "))

    self.label.setText(_translate("MainWindow", "账号: "))

    self.pushButton.setText(_translate("MainWindow", "登录"))

    self.pushButton_2.setText(_translate("MainWindow", "注册"))
```

#### 6.2.4.6 loginWindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

    <class>MainWindow</class>

    <widget class="QMainWindow" name="MainWindow">
```

```
<property name="geometry">

    <rect>

        <x>0</x>

        <y>0</y>

        <width>800</width>

        <height>600</height>

    </rect>

</property>

<property name="windowTitle">

    <string>MainWindow</string>

</property>

<widget class="QWidget" name="centralwidget">

    <widget class="QGroupBox" name="groupBox">

        <property name="geometry">

            <rect>

                <x>50</x>

                <y>30</y>

                <width>701</width>

                <height>481</height>

            </rect>

        </property>

        <property name="title">

            <string>登录</string>

        </property>

        <widget class="QWidget" name="gridLayoutWidget">

            <property name="geometry">

                <rect>

                    <x>210</x>

                    <y>100</y>

                    <width>271</width>
```

```
<height>201</height>

</rect>

</property>

<layout class="QGridLayout" name="gridLayout">

  <item row="1" column="0">

    <widget class="QLabel" name="label_2">

      <property name="text">

        <string>密码: </string>

      </property>

    </widget>

  </item>

  <item row="0" column="0">

    <widget class="QLabel" name="label">

      <property name="text">

        <string>账号: </string>

      </property>

    </widget>

  </item>

  <item row="0" column="1">

    <widget class="QLineEdit" name="lineEdit"/>

  </item>

  <item row="1" column="1">

    <widget class="QLineEdit" name="lineEdit_2"/>

  </item>

</layout>

</widget>

<widget class="QWidget" name="horizontalLayoutWidget">

  <property name="geometry">

    <rect>

      <x>230</x>
```

```
<y>340</y>

<width>231</width>

<height>33</height>

</rect>

</property>

<layout class="QHBoxLayout" name="horizontalLayout">

  <item>

    <widget class="QPushButton" name="pushButton">

      <property name="text">

        <string>登录</string>

      </property>

    </widget>

  </item>

  <item>

    <widget class="QPushButton" name="pushButton_2">

      <property name="text">

        <string>注册</string>

      </property>

    </widget>

  </item>

</layout>

</widget>

</widget>

</widget>

<widget class="QMenuBar" name="menubar">

  <property name="geometry">

    <rect>

      <x>0</x>

      <y>0</y>

      <width>800</width>
```



```
<height>28</height>

</rect>

</property>

</widget>

<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>
```

#### 6.2.4.7 signUp.py

```
from PyQt5 import QtWidgets
from ui.signUpWindow import Ui_MainWindow
from mysql.connectMysql import ConnetMysql

class SignUp(QtWidgets.QWidget, Ui_MainWindow):

    def __init__(self):
        super(SignUp, self).__init__()

        self.setupUi(self)

        self.db = ConnetMysql()

        self.pushButton.clicked.connect(self.sign_up)

    def sign_up(self):

        if self.lineEdit_2.text() != self.lineEdit_3.text():

            return

        id = self.lineEdit.text()

        password = self.lineEdit_2.text()

        users = self.db.search("select id from users")

        for user in users:
```

```

        if user[0] == id:

            QtWidgets.QMessageBox.information(self, "提示", "该用户已存在, 请直接登录")

            return

        self.db.insert("insert into users values('%s', '%s', '0', 0, 0)"%(id, password))

        create_record_sql = "create table {}_record(" \
                               "date char(10) not null," \
                               "time char(15) not null," \
                               "record varchar(500) not null)".format(id)

        self.db.create_table(create_record_sql)

        QtWidgets.QMessageBox.information(self, "提示", "账号注册成功")

        self.close()

```

#### 6.2.4.8 signUpWindow.py

```

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'signUpWindow.ui'
#
# Created by: PyQt5 UI code generator 5.12.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(710, 474)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

```

```
self.centralwidget.setObjectName("centralwidget")

self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)

self.gridLayoutWidget.setGeometry(QtCore.QRect(170, 120, 341, 171))

self.gridLayoutWidget.setObjectName("gridLayoutWidget")

self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)

self.gridLayout.setContentsMargins(0, 0, 0, 0)

self.gridLayout.setObjectName("gridLayout")

self.lineEdit_2 = QtWidgets.QLineEdit(self.gridLayoutWidget)

self.lineEdit_2.setObjectName("lineEdit_2")

self.gridLayout.addWidget(self.lineEdit_2, 2, 1, 1, 1)

self.label_2 = QtWidgets.QLabel(self.gridLayoutWidget)

self.label_2.setObjectName("label_2")

self.gridLayout.addWidget(self.label_2, 2, 0, 1, 1)

self.label = QtWidgets.QLabel(self.gridLayoutWidget)

self.label.setObjectName("label")

self.gridLayout.addWidget(self.label, 0, 0, 1, 1)

self.label_3 = QtWidgets.QLabel(self.gridLayoutWidget)

self.label_3.setObjectName("label_3")

self.gridLayout.addWidget(self.label_3, 4, 0, 1, 1)

self.lineEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)

self.lineEdit.setObjectName("lineEdit")

self.gridLayout.addWidget(self.lineEdit, 0, 1, 1, 1)

spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)

self.gridLayout.addItem(spacerItem, 3, 0, 1, 1)

self.lineEdit_3 = QtWidgets.QLineEdit(self.gridLayoutWidget)

self.lineEdit_3.setObjectName("lineEdit_3")

self.gridLayout.addWidget(self.lineEdit_3, 4, 1, 1, 1)

spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
```

```
self.gridLayout.addItem(spacerItem1, 1, 0, 1, 1)

self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(300, 330, 71, 31))
self.pushButton.setObjectName("pushButton")

self.label_4 = QtWidgets.QLabel(self.centralwidget)
self.label_4.setGeometry(QtCore.QRect(320, 50, 61, 41))
self.label_4.setObjectName("label_4")

# MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 710, 28))
self.menubar.setObjectName("menubar")

# MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")

# MainWindow.setStatusBar(self.statusbar)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
    self.label_2.setText(_translate("MainWindow", "    密码    "))
    self.label.setText(_translate("MainWindow", "    账号"))
    self.label_3.setText(_translate("MainWindow", "    密码"))
    self.pushButton.setText(_translate("MainWindow", "确认"))
    self.label_4.setText(_translate("MainWindow", "登录"))
```

#### 6.2.4.9 signUpWindw.ui

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

    <property name="geometry">

        <rect>

            <x>0</x>

            <y>0</y>

            <width>710</width>

            <height>474</height>

        </rect>

    </property>

    <property name="windowTitle">

        <string>MainWindow</string>

    </property>

    <widget class="QWidget" name="centralwidget">

        <widget class="QWidget" name="gridLayoutWidget">

            <property name="geometry">

                <rect>

                    <x>170</x>

                    <y>120</y>

                    <width>341</width>

                    <height>171</height>

                </rect>

            </property>

            <layout class="QGridLayout" name="gridLayout">

                <item row="2" column="1">

                    <widget class="QLineEdit" name="lineEdit_2"/>

                </item>

                <item row="2" column="0">

                    <widget class="QLabel" name="label_2">
```

```

    <property name="text">

        <string> 密码 </string>

    </property>

</widget>

</item>

<item row="0" column="0">

    <widget class="QLabel" name="label">

        <property name="text">

            <string> 账号 </string>

        </property>

    </widget>

</item>

<item row="4" column="0">

    <widget class="QLabel" name="label_3">

        <property name="text">

            <string> 密码 </string>

        </property>

    </widget>

</item>

<item row="0" column="1">

    <widget class="QLineEdit" name="lineEdit"/>

</item>

<item row="3" column="0">

    <spacer name="verticalSpacer_2">

        <property name="orientation">

            <enum>Qt::Vertical</enum>

        </property>

        <property name="sizeHint" stdset="0">

            <size>

                <width>20</width>

```

```

        <height>40</height>

    </size>

    </property>

</spacer>

</item>

<item row="4" column="1">

    <widget class="QLineEdit" name="lineEdit_3"/>

</item>

<item row="1" column="0">

    <spacer name="verticalSpacer">

        <property name="orientation">

            <enum>Qt::Vertical</enum>

        </property>

        <property name="sizeHint" stdset="0">

            <size>

                <width>20</width>

                <height>40</height>

            </size>

        </property>

    </spacer>

</item>

</layout>

</widget>

<widget class="QPushButton" name="pushButton">

    <property name="geometry">

        <rect>

            <x>300</x>

            <y>330</y>

            <width>71</width>

            <height>31</height>

```

```

    </rect>

</property>

<property name="text">
    <string>确认</string>
</property>
</widget>

<widget class="QLabel" name="label_4">
    <property name="geometry">
        <rect>
            <x>320</x>
            <y>50</y>
            <width>61</width>
            <height>41</height>
        </rect>
    </property>
    <property name="text">
        <string>登录</string>
    </property>
</widget>
</widget>

<widget class="QMenuBar" name="menubar">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>710</width>
            <height>28</height>
        </rect>
    </property>
</widget>

```



```
<widget class="QStatusBar" name="statusbar"/>

</widget>

<resources/>

<connections/>

</ui>
```

## 6.2.5 客户端主程序

### 6.2.5.1 Main.py 与 main2.py

```
from ui.login import Login
from ui.chat import Chat
from ui.signUp import SignUp

from PyQt5 import QtWidgets
import sys

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)

    chat_window = Chat()
    signup_window = SignUp()
    login_window = Login(chat_window, signup_window)
    login_window.show()

    sys.exit(app.exec_())
```

## 7 实验总结

本次实验让我对 RSA, DES 加密算法有了更深入的理解。让我熟悉了密钥的分发过程, 数字信封的制作过程, 公钥证书的制作过程等等, 可以说是收获良多。

本次实验在实现代码上有几个难点:

**首先是通过 socket 协议文字转码的问题。**

由于我一开始并不知道 socket 协议传输数据是只通过二进制字节流传输, 导致传输的协议出现前后消息不一致的问题, 我还试过 utf-8、unicode 等很多编码, 而实际上 python socket 都是不支持的。

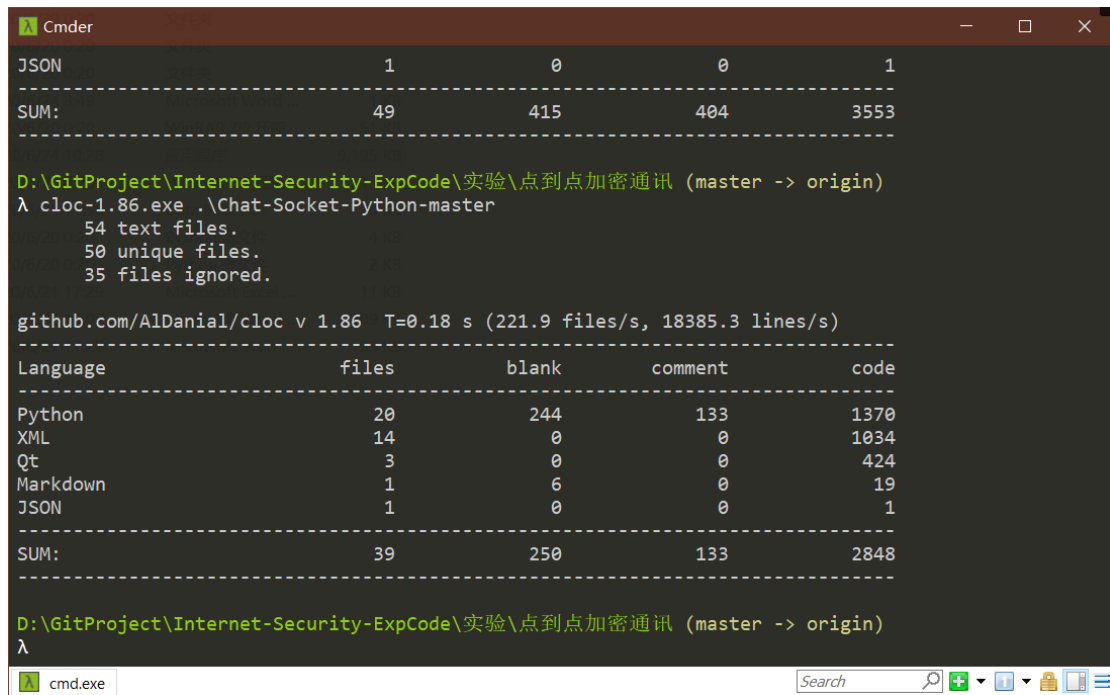
**其次是服务器收发消息的问题。**

我一开始想设计通过服务器转发消息, 这样可以实现多对多通讯, 而每一对通信用户可以分别用独立的公私钥加密通信, 然而这样的一个明显的问题是对服务器的要求很高, 一旦服务器出现问题, 传输将变的不可靠。因此, 我将服务器的角色转换, 从消息的分发者转变为用户的管理者, 服务器只负责提供在线用户名单和管理密钥, 而实际的通讯则用客户端 (在同一局域网) end-to-end 直接发送, 这样在进一步加固了消息传输的可靠性的同时减轻了服务器的负担。

**再次是用户的文件传输问题。**

我这里使用了 DES 加密文件内容, 这样一个明显的好处是加速了文件加密过程, 同时在数字信封的配合下同时保证了机密性。而如果用非对称加密算法在传输大文件时则会出现传输过程过慢问题。

本次大作业花费了很多时间完成编码和调试, 我用 python 的 PyQt5 框架开发客户端界面, 由于之前从未使用过 PyQt5, 这次实验也相当于从 0 学起, 给我带来了不少的挑战。中间也遇到了很多问题, 而不断地查找资料修修补补, 不知不觉代码已经写了这么多了:



```

JSON          1          0          0          1
-----
SUM:          49        415        404       3553
-----

D:\GitProject\Internet-Security-ExpCode\实验\点到点加密通讯 (master -> origin)
λ cloc-1.86.exe .\Chat-Socket-Python-master
  54 text files.
  50 unique files.
  35 files ignored.

github.com/AlDanial/cloc v 1.86 T=0.18 s (221.9 files/s, 18385.3 lines/s)
-----
Language      files      blank      comment      code
-----
Python        20         244         133         1370
XML           14          0           0         1034
Qt            3           0           0          424
Markdown      1           6           0           19
JSON          1           0           0           1
-----
SUM:          39        250        133        2848
-----

D:\GitProject\Internet-Security-ExpCode\实验\点到点加密通讯 (master -> origin)
λ
  
```

图 12: 代码统计

这次实验收获还是很大的。人生苦短，我用 Python！