《密码学》课程设计实验报告

实验序号：08　　实验项目名称：密码学综合实验

学号	2016301500327	姓名	肖轩淦	专业、班	16 信安 3-4 班
实验地点	网安学院 A3-1、A3-2 实验室	指导教师	王张宜	时间	2018.12.21

一、　实验目的及要求

教学目的：

(1) 熟悉适合密码应用的领域及应用方法；

(2) 掌握一种密码应用技术。

实验要求：

(1) 掌握对称算法的实现；

(2) 掌握非对称算法的实现；

(3) 实现下列密码学的一种（或多种）应用：

- 计算机文件加密
- 通信加密
- 认证
- 签名
- 密钥管理

(4) 掌握加密、签名、认证、密钥管理等技术的综合运用。

二、实验设备（环境）及要求

Windows 10 64 位

Python 3.6.1

GUI 库：wxpython

三、实验内容与步骤

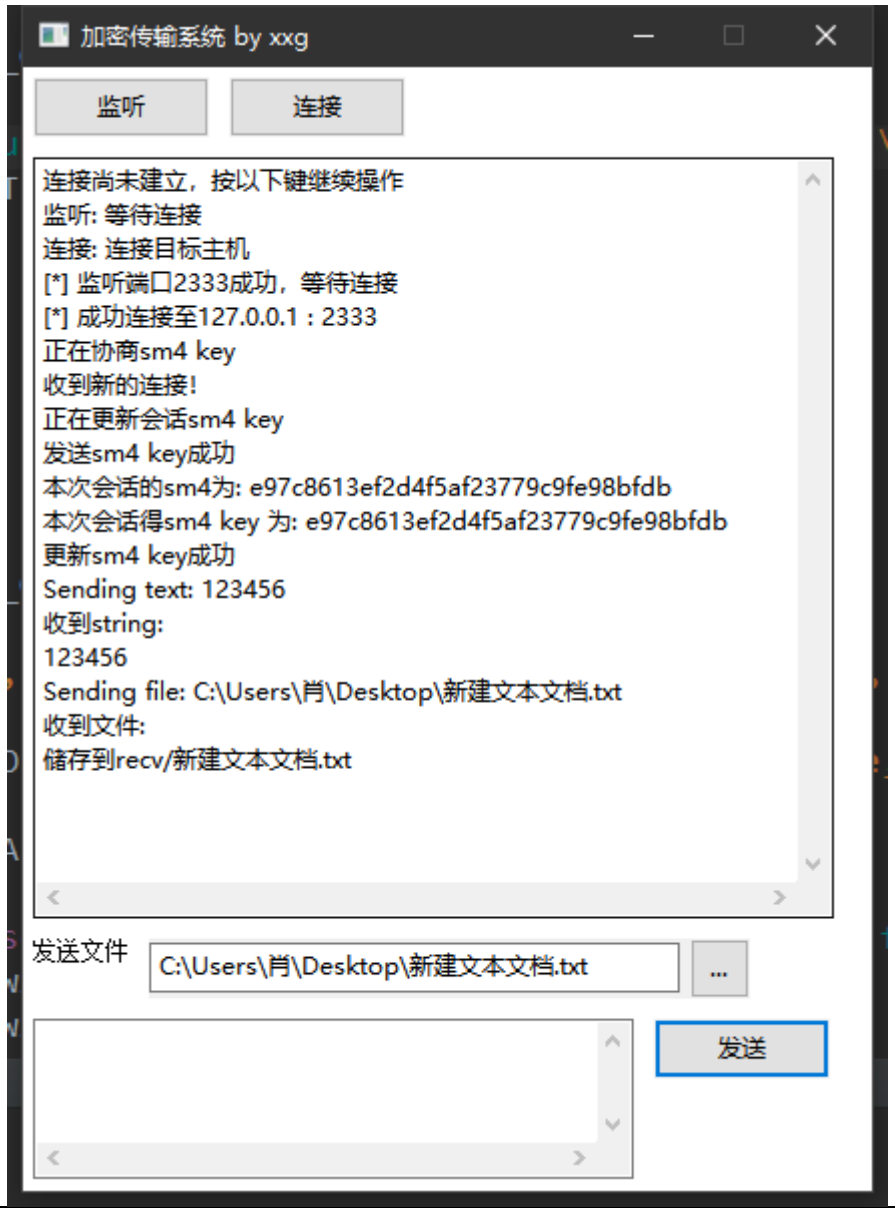
作品名称：基于通信加密的文字文件传输系统

简介：本软件是基于通信加密的文字文件传输软件，实现了接收端、发送端二合

一，可在网络内[1]实现全双工通信[2]。通信加密使用公开加密算法协商单次会话使用的分组密码密钥，使用分组密码进行加密通信，可自定义公开加密算法的相关参数，并设计了一套简单的通信协议及包格式，以校验发送的加密数据的完整性。

注：(1) 可在两台主机都具有公网 ip 时，或是在同一无 NAT 端口映射的局域网时进行全双工通信。

(2)若发送端和接收端在同一台主机上工作时，由于端口的唯一性，应用程序与端口绑定，故只能实现半双工通信。

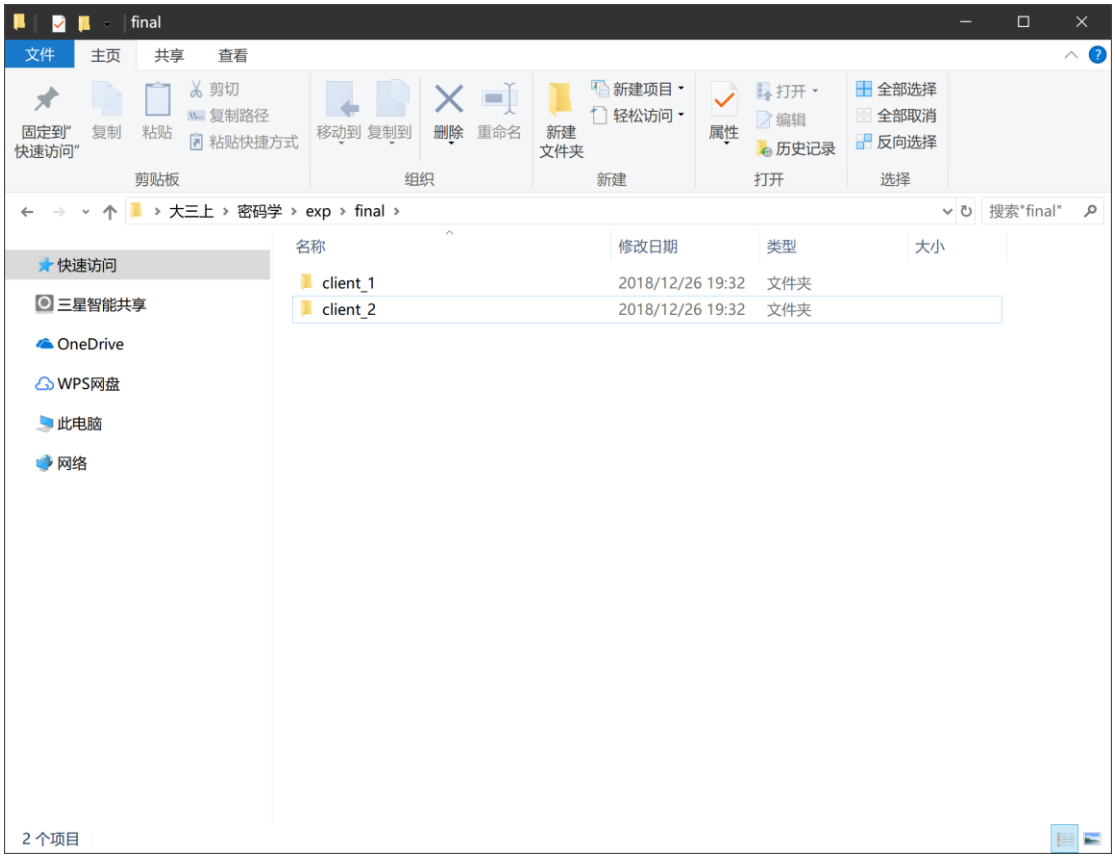


程序运行截图（这是使用单个终端进行发送接收测试）：

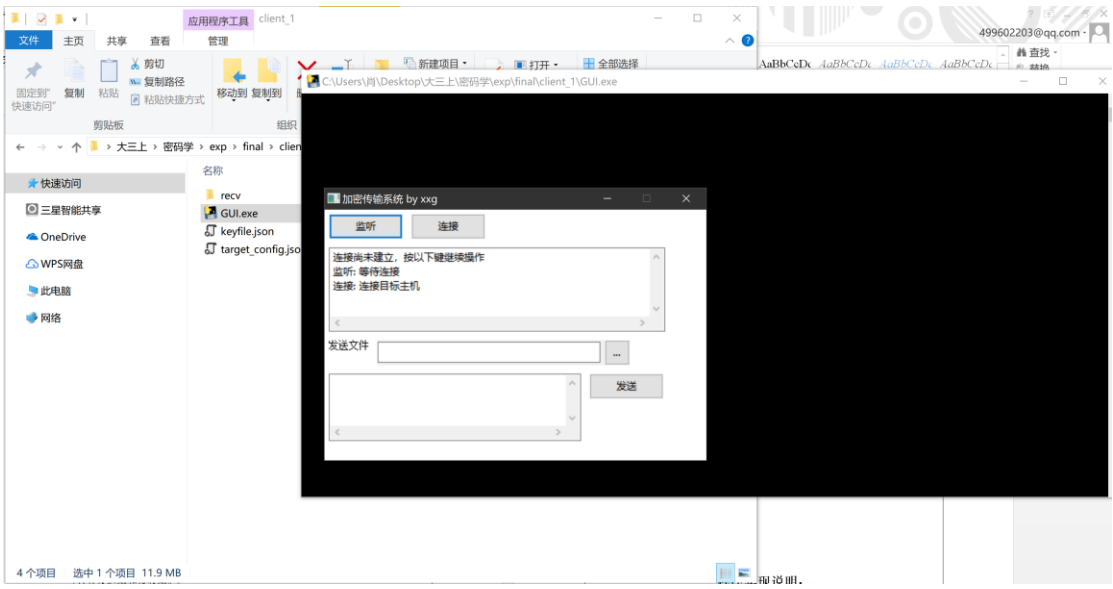
程序使用说明：

以下为单机情况进行半双工通信为例：

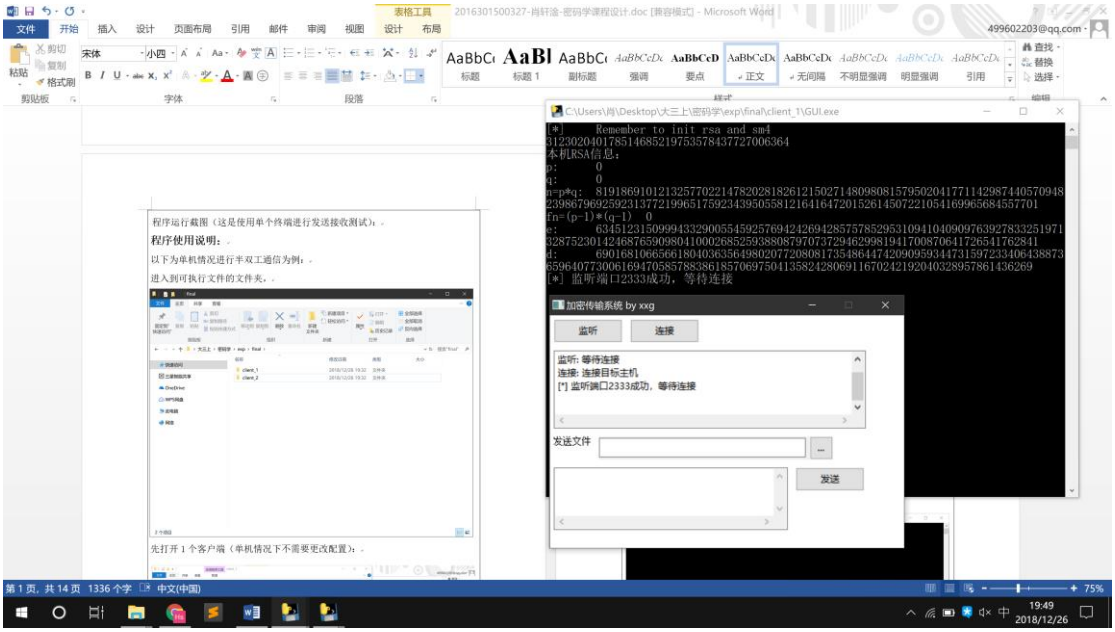
进入到可执行文件的文件夹，



先打开 1 个客户端（单机情况下不需要更改配置）：

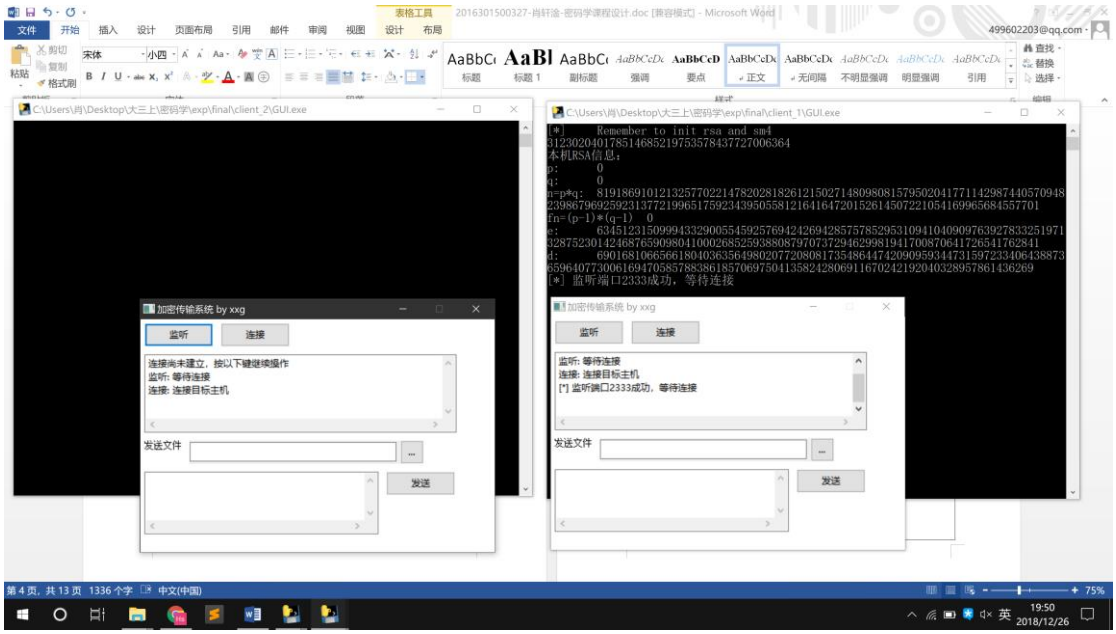


点击监听：

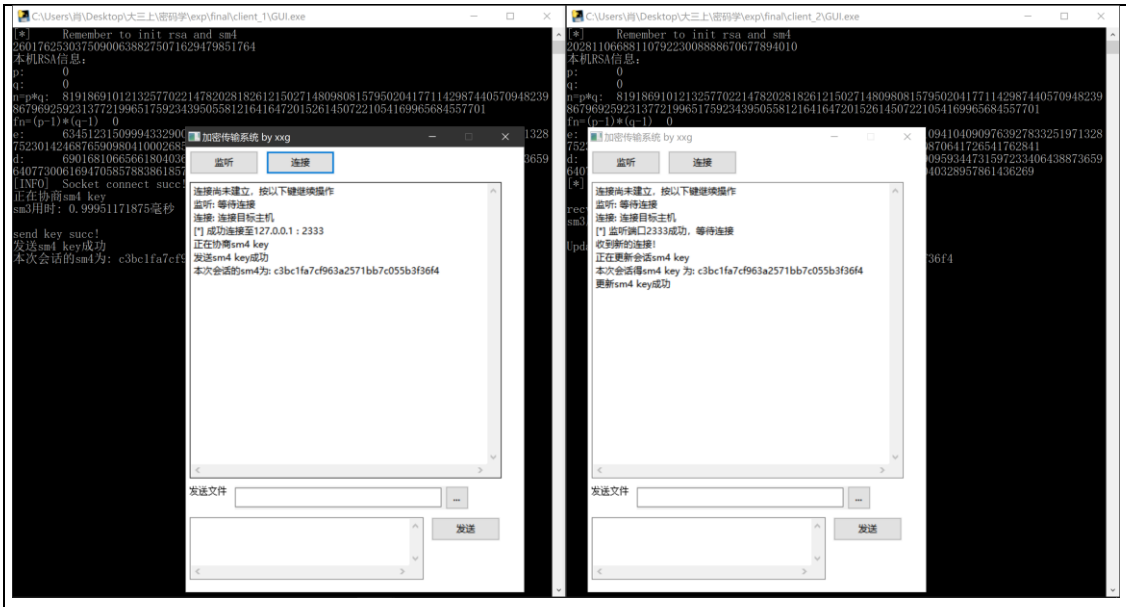


若出现以上界面，说明端口监听成功；若提示失败，转至下方可能出现的问题-端口冲突

打开另外一个客户端（单机情况不需要配置）:（左边为新建客户端，右边为已开启监听的客户端）

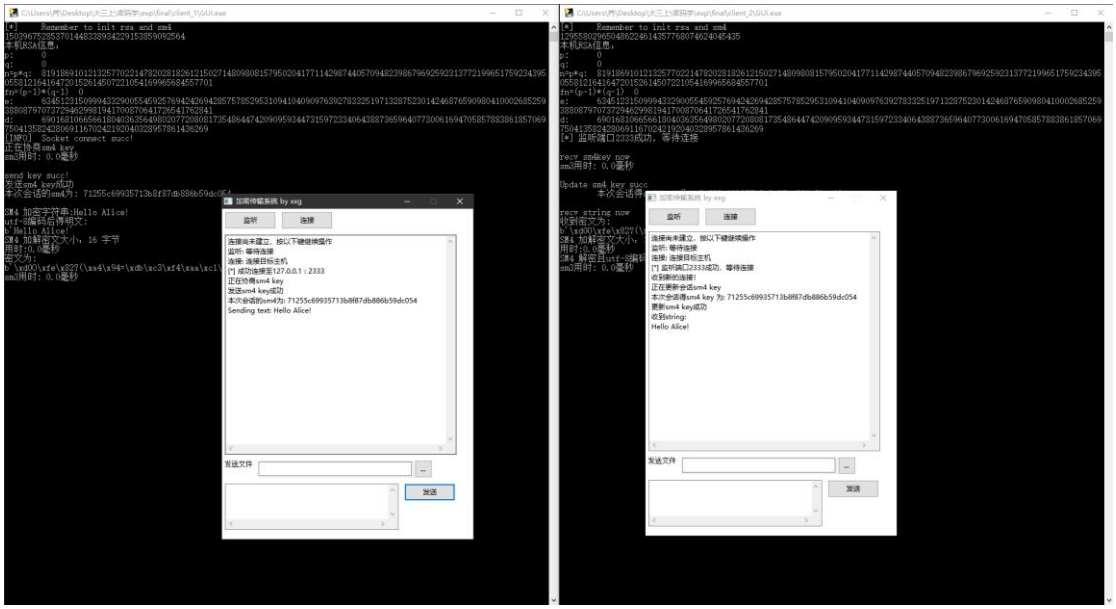


点击新打开客户端的连接：

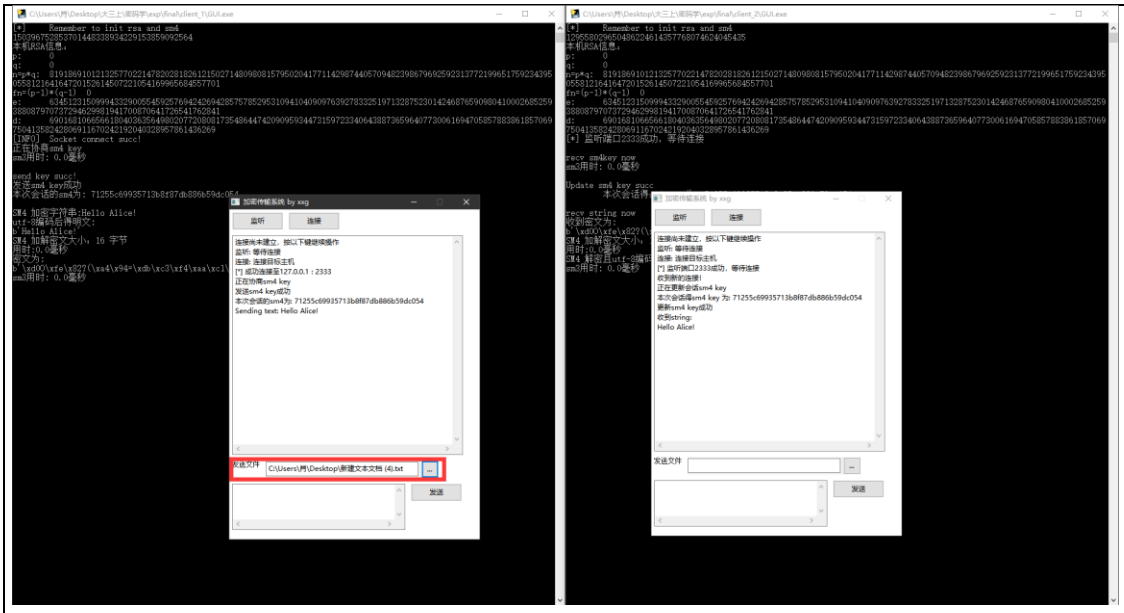


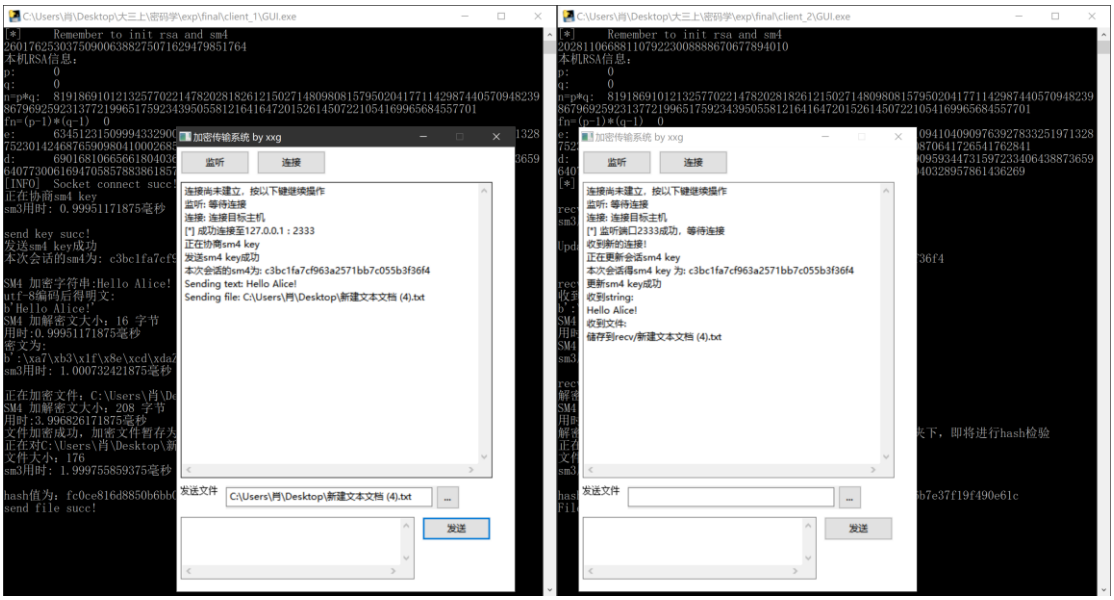
此时，已协商好 sm4 会话密钥。由于现在是在单机上测试，只能进行半双工通信。连接方为发送方，监听方为接收方。

发送方在输入栏输入并发送一个字符串

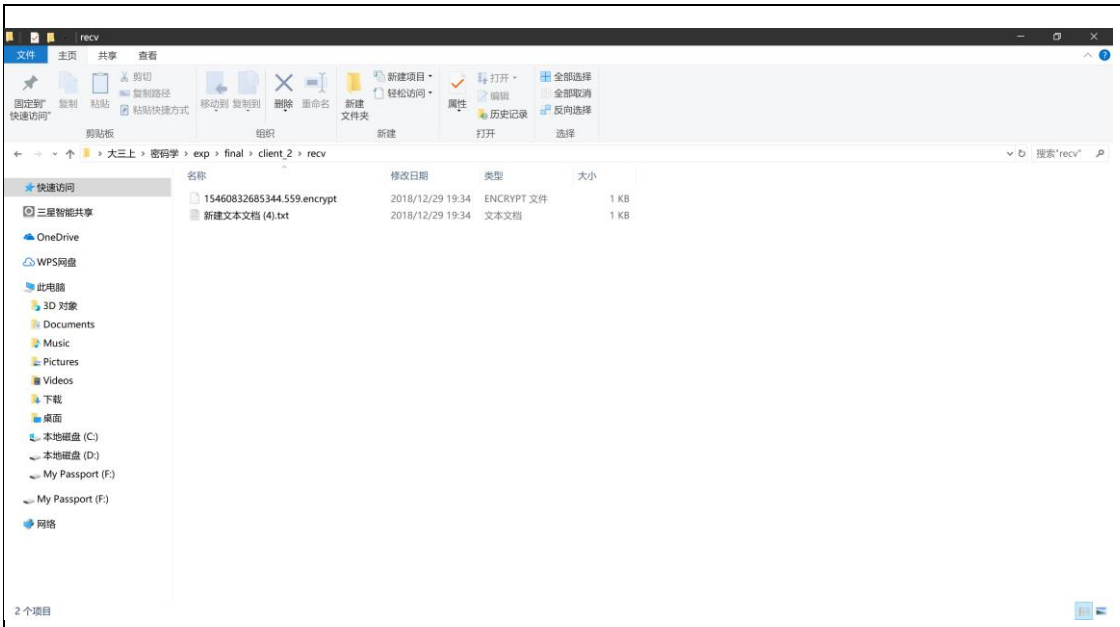


接下来发送一个文件：选择“…”按钮，选择文件，点击发送。

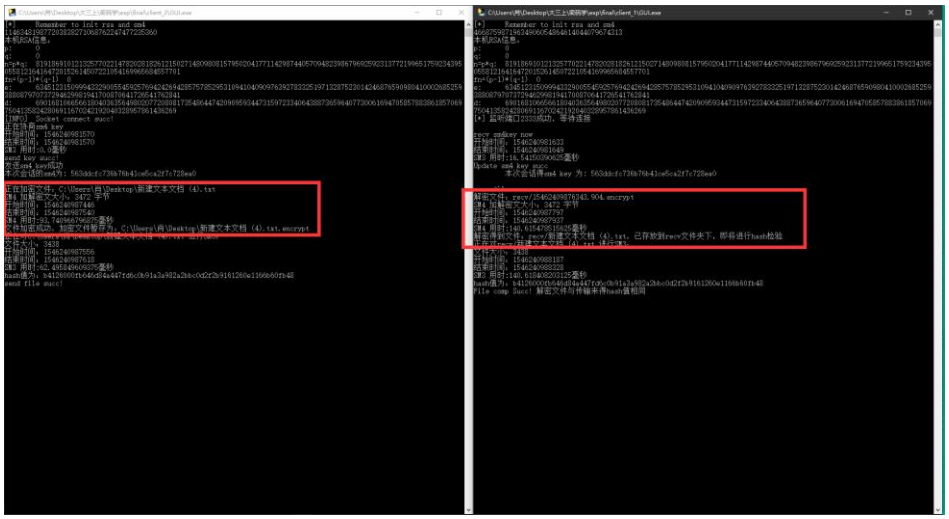




可见接收方收到了文件，文件储存于接收方目录下的 recv 文件夹内，



统计信息可在命令行中查看（尽量使用 1KB-1MB 的文件，避免时间显示不准确）：

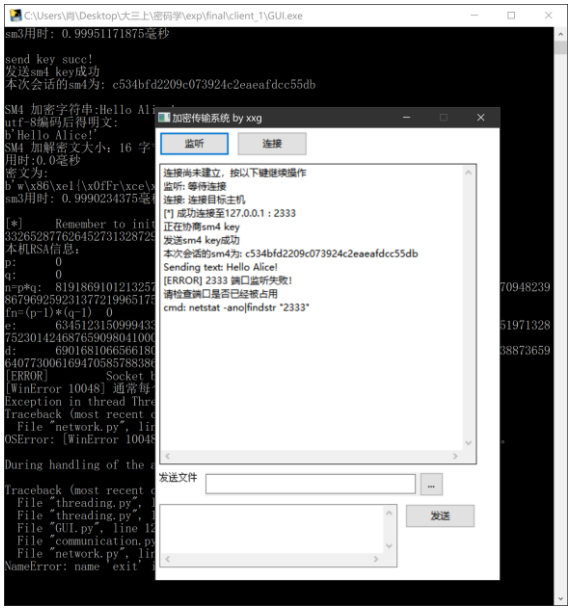


打开文件检查加解密结果

相同！加密传输文件成功。

可能出现的问题：

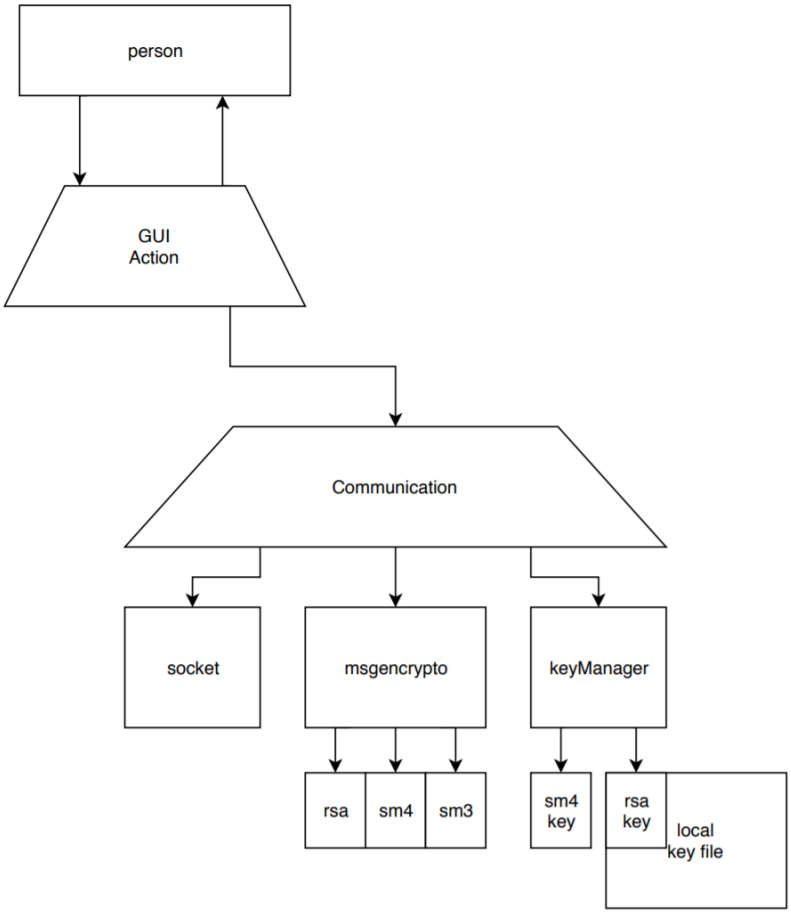
1、端口被占用



通过 netstat -ano|findstr "2333"指令检查是否有程序已经占用了软件所需端口。

程序实现说明：

一、　软件架构



软件架构及各文件作用说明（从上至下）：

- Person指的是用户，用户与GUI界面进行交互。
- GUI模块在./GUI.py中实现，作用是绘制窗口及绑定按钮事件
- Communication模块是在./communication.py中实现，其是对socket、msgencrypto、keymanager三个模块进行封装，并实现了通信协议的收发控制。
- Socket模块是在./network.py中实现，其作用是根据上层给定的ip和端口，建立连接或监听端口，并对实现了发送接收文件、发送接收字符串、发送接收bytes的方法，方便上层模块调用。
- Msgencrypto模块是在crypto/MsgCrypto.py中实现，实现了rsa初始化加解密、sm4初始化及文件字符串的加解密、sm3文件字符串的hash方法，方便上层模块调用。
- Rsa即rsa加解密模块，在crypto/RSA.py中实现
- Sm4为sm4加解密模块，在crypto/SM4.py中实现
- Sm3为sm3 hash模块，在crypto/SM3.py中实现
- Keymanager为密钥管理模块，作用是对用户的rsa公私钥进行管理，并实现了生成sm4会话密钥的方法。

项目文件夹中其他文件的作用：

- Keyfile.json 储存RSA512的相关信息
- Target_config.json 储存接收方的ip、端口
- Config_util.py 一些配置变量，可以调整其，以使得加解密时能够输出更多的信息
- Crypto/number_theory.py 一些数论函数的实现，方便RSA调用

- Recv/ 接收文件文件夹，接收方收到文件后，将储存到这个文件夹下。

二、 通信流程

A、B代表两个客户端，以单工通信为例进行说明。

1	A开始监听	
2		B对A发起连接
3		B生成sm4会话密钥，使用A的公钥加密，发送给A
4	A使用私钥解密sm4会话密钥	
5		B使用sm4会话密钥加密文字并添加校验进行传输
6	A接收密文，进行解密并对比hash	
7		B使用sm4会话密钥加密文件并添加校验进行传输
8	A收到密文，进行解密并对比hash	

主要类及其函数说明

对于一些重要的类，及方法进行说明，大部分函数可以通过其函数名大概了解其功能。

一、 GUI.py

绘制窗口及绑定按钮事件

```
# GUI 界面
class MyFrame(wx.Frame):
    def __init__(self, parent):...

    def __del__(self):...

    # 监听按钮事件，开启监听线程
    def m_button_listen_OnButtonClick(self, event):...

    # 连接按钮事件，开启连接线程
    def m_button_connect_OnButtonClick(self, event):...

    # 发送按钮事件，发送文字或文件
    def m_button_send_OnButtonClick(self, event):...

    # 监听线程
    def listen_thread(self):...

    # 连接线程
    def connect_thread(self):...

    # 显示结果线程
    def disp_text(self):...

    # 开启显示结果线程
    def disp_text_begin(self):...

if __name__ == "__main__":
    app = wx.App(False)
    frame = MyFrame(None)
    frame.Show(True)
    frame.disp_text_begin()
    app.MainLoop()
```

二、 communication.py

对socket、msgencrypto、keymanager三个模块进行封装，并实现了通信协议的收发控制。

```python
# 首部类型字段
msg_type_dict = {
    'TRANSFER_STRING': (10).to_bytes(4, byteorder='little'),
    'TRANSFER_FILE': (11).to_bytes(4, byteorder='little'),
    'UPDATE_SM4_KEY': (20).to_bytes(4, byteorder='little'),
}


class Communication:
    def __init__(self, mode='LISTEN', target_ip='127.0.0.1', port=2333, msg_queue=[]):...

    ''' LISTEN '''
    # 监听的管理程序，判断收到的包类型，调用相应的函数处理发送来的数据
    def listen_manager(self):...

    # 接收字符串
    def recv_string(self):...

    # 接收文件
    def recv_file(self):...

    # 更新sm4 key
    def recv_sm4_key(self):...

    ''' CONNECT '''
    # 发送一个字符串
    def send_string(self, string):...

    # 发送文件
    def send_file(self, f_path):...

    # 协商sm4 key
    def send_sm4_key(self, sm4_key):...

    # 加载接收方的本地配置信息
    def fetch_local_config(self):...

    # 保存接收端的配置信息到本地文件
    def dump_local_config(self):...
```

三、 network.py

根据上层给定的ip和端口，建立连接或监听端口，并对实现了发送接收文件、发送接收字符串、发送接收bytes的方法，方便上层模块调用。

```
class MySocket:

    LIESTEN = 0
    CONNECT = 1

    def __init__(self, msg_queue=[]):...

    # 接收新的连接
    def accept_new_conn(self):...

    def listen(self, port=2333):...

    def recv_bytes(self, size=1024):...

    def recv_str(self, size=1024):...

    # 比较特殊，返回值为临时文件名
    def recv_file(self, size):...

    '''分界线 下面是发送方'''

    def connect(self, ip="127.0.0.1", port=2333):...

    def send_bytes(self, input_bytes):...

    def send_str(self, input_str):...

    def send_file(self, f_name):...

    def test(self):...
```

四、 MsgCrypto.py

```python
class MsgCrypto:
    def __init__(self):...

    # 发送方rsa初始化
    def rsa_my_init(self, p=0, q=0, n=0, e=0, d=0):...

    # 接收方rsa初始化
    def rsa_target_init(self, p=0, q=0, n=0, e=0, d=0):...

    # sm4 初始化
    def sm4_init(self, key, mode=crypto.SM4.ENCRYPT):...

    # 用对方公钥加密
    def rsa_encrypt(self, m):...

    # 用自己的私钥解密
    def rsa_decrypt(self, c):...

    def sm4_encrypt_msg(self, m):...

    def sm4_decrypt_msg(self, c):...

    # sm4 加密文件 正常模式不包括文件名，network模式在文件开头添加f_name+'\0'
    def sm4_encrypt_file(self, f_path, mode='normal'):...

    # sm4 解密文件
    def sm4_decrypt_file(self, f_path=None, mode='normal'):...

    def sm3_hash_str(self, msg):...

    def sm3_hash_file(self, f_path):...
```

五、keymanager.py

```python
class KeyManager:

    def __init__(self):...

    def fetch_local_key(self):...

    def dump_local_key(self):...

    def new_rsa_key(self):...

    def new_SM4_key(self, keylen=128):...

    def get_rsa_public_key(self):...

    def get_rsa_private_key(self):...

    def get_rsa_n(self):
        return self.keys["n"]

    def get_sm4_key(self):...
```

六、SM4

```
class Sm4(object):
    def __init__(self):...

    def sm4_set_key(self, key_data, mode):...

    def sm4_setkey(self, key, mode):...

    def sm4_one_round(self, sk, in_put):...

    def sm4_crypt_ecb(self, input_data, mode=ENCRYPT):...

    # 在加密之前padding
    def sm4_pad_data(self, input_data):...

    # 在解密后unpadding
    def sm4_unpad_data(self, data):...

    def print_selfkey(self):...
```

七、RSA

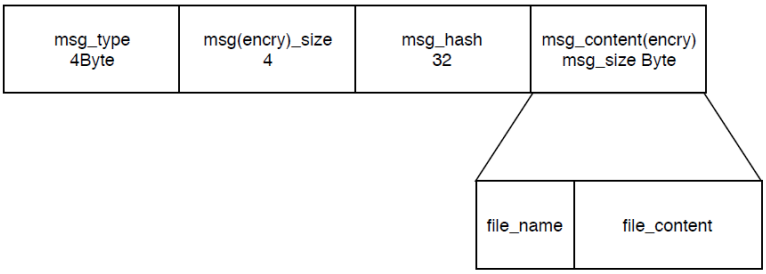
```
class RSA:

    def random_select_p_q(self):...

    def select_e(self):...

    # 输入:
    # (p q)|(n) + (e)|(d)
    def __init__(self, p=0, q=0, n=0, e=0, d=0, keylen=0, mode='SMALL'):...

    def calc_n(self):...

    def calc_fn(self):...

    def calc_e_from_d(self):...

    def calc_d_from_e(self):...

    def encrypt(self, m):...

    def decrypt(self, c):...

    def print_detail(self):...
```

```
# gcd
def gcd(a, b):...


# 扩展gcd
def extendedGCD(a, b):...


# 快速幂
def fastExpMod(b, e, m):...


# 欧拉函数
def euler(n):...


# 素性检验
def primeTest(n):...

# 生成素数
def findPrime(halfkeyLength):...
```

八、SM3

```
class Sm3:

    IV = [0x7380166f, 0x4914b2b9, 0x172442d7, 0xda8a0600, 0xa96f30bc, 0x163138aa, 0xe38dee4d, 0xb0fb0e4e]
    T_j = [...]

    def __int__(self):...

    def rotate_left(self, a, k):...

    def FF_j(self, X, Y, Z, j):...

    def GG_j(self, X, Y, Z, j):...

    def P_0(self, X):...

    def P_1(self, X):...

    def CF(self, V_i, B_i):...

    # 输入 字节流
    def hash_msg(self, msg_bytes):...
```

通信协议及通信数据格式



Msg_type: 4 字节。目前实现了三种：传输字符串、传输文件、协商sm4密钥

Msg(encry)_size: 4字节，其值为密文长度

Msg_hash: 32字节，为明文的sm3的hash值

Msg_content: 长度由长度字段定义，传输字符串模式下为字符串的密文；传输文件模式下为文件名和文件内容的密文，以\x00分隔；协商sm4模式下为公钥加密后的sm4密钥。

在Class Communication中实现。

其他

一、 运行速度

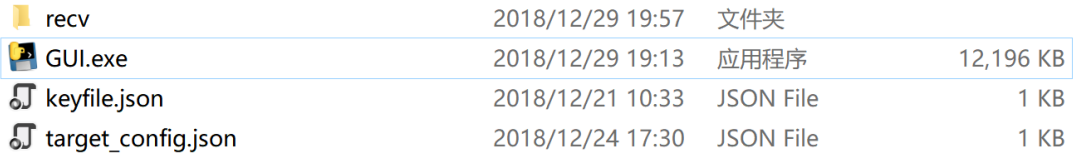
由于是 python 程序，执行速度较 c++偏慢，但由于采用了分组密码，速度也还是不错的。

```
正在加密文件：C:\Users\肖\Desktop\新建文本文档 (4).txt
SM4 加解密文大小：208 字节
用时:2.996826171875毫秒
文件加密成功，加密文件暂存为：C:\Users\肖\Desktop\新建文本文档 (4).txt.encrypt
recv file now
正在对C:\Users\肖\Desktop\新建文本文档 (4).txt 进行SM3：
文件大小：176
sm3用时： 2.000732421875毫秒

hash值为：fc0ce816d8850b6bb0abdbd1eb5e4c2bf1bcc996a06388b6b7e37f19f490e61c
send file succ!

解密文件：recv/154608464122253.318.encrypt
SM4 加解密文大小：208 字节
用时:3.997558593750毫秒
解密得到文件：recv/新建文本文档 (4).txt，已存放到recv文件夹下，即将进行hash检验
正在对recv/新建文本文档 (4).txt 进行SM3：
文件大小：176
sm3用时： 3.0毫秒

hash值为：fc0ce816d8850b6bb0abdbd1eb5e4c2bf1bcc996a06388b6b7e37f19f490e61c
File comp Succ! 解密文件与传输来得hash值相同
```

二、 程序占用空间

由于还要包含相应的 GUI 库及其他库函数，程序显得稍大一些，但也还是可以接

受。



recv	2018/12/29 19:57	文件夹	
GUI.exe	2018/12/29 19:13	应用程序	12,196 KB
keyfile.json	2018/12/21 10:33	JSON File	1 KB
target_config.json	2018/12/24 17:30	JSON File	1 KB

三、 安全功能

安全功能较为完善，使用了公钥协商会话密钥，既保证了安全性，也提高了加解密速度，使用 512 位 RSA，并带有 sm3 hash 校验码，保证加密数据的完整性。生成大整数算法在 crypto/number_theory.py 中实现，但由于需要时间较长，不便于演示，故没有加入重新生成大整数的按钮功能。

四、 算法扩展性

通过多层的封装，可以十分方便的对使用的加密算法进行替换迭代。在 MsgCrypto 中，将加密算法均已封装好，添加新的加密算法也十分方便，只需在 MsgCrypto 中留出接口，并在 keymanager 中对其密钥有相应的管理即可。

```
class MsgCrypto:
    def __init__(self):...

    # 发送方rsa初始化
    def rsa_my_init(self, p=0, q=0, n=0, e=0, d=0):...

    # 接收方rsa初始化
    def rsa_target_init(self, p=0, q=0, n=0, e=0, d=0):...

    # sm4 初始化
    def sm4_init(self, key, mode=crypto.SM4.ENCRYPT):...

    # 用对方公钥加密
    def rsa_encrypt(self, m):...

    # 用自己的私钥解密
    def rsa_decrypt(self, c):...

    def sm4_encrypt_msg(self, m):...

    def sm4_decrypt_msg(self, c):...

    # sm4 加密文件 正常模式不包括文件名，network模式在文件开头添加f_name+'\0'
    def sm4_encrypt_file(self, f_path, mode='normal'):...

    # sm4 解密文件
    def sm4_decrypt_file(self, f_path=None, mode='normal'):...

    def sm3_hash_str(self, msg):...

    def sm3_hash_file(self, f_path):...
```

五、　执行速率和数据吞吐量统计

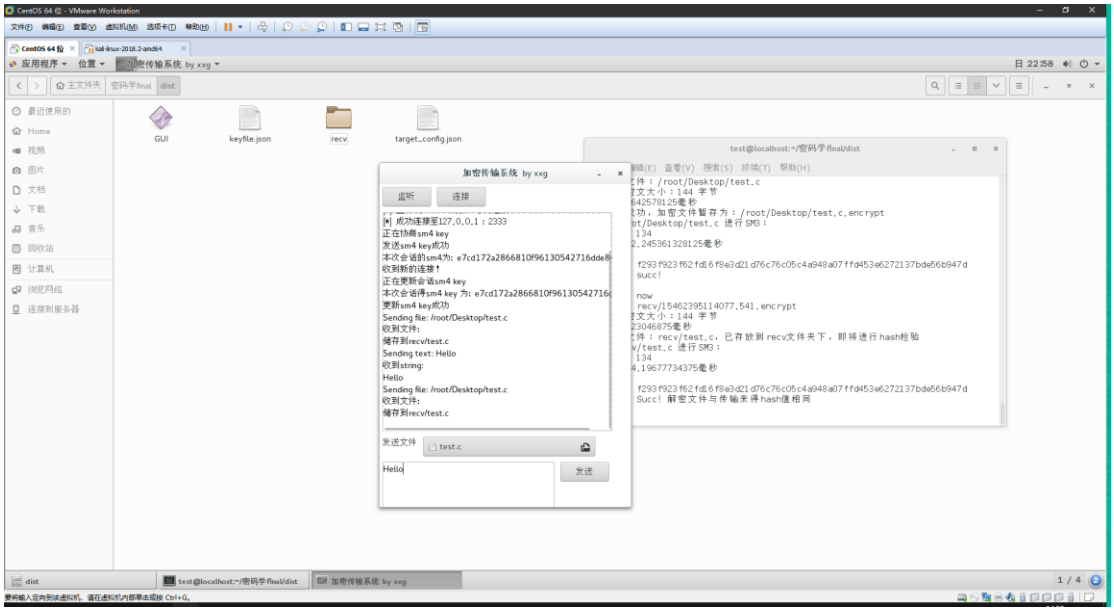
在进行加解密时，都会对文件文字的 sm3、sm4 运算速度进行统计，但是由于 cpu 繁忙程度不同，用时也稍有不同。

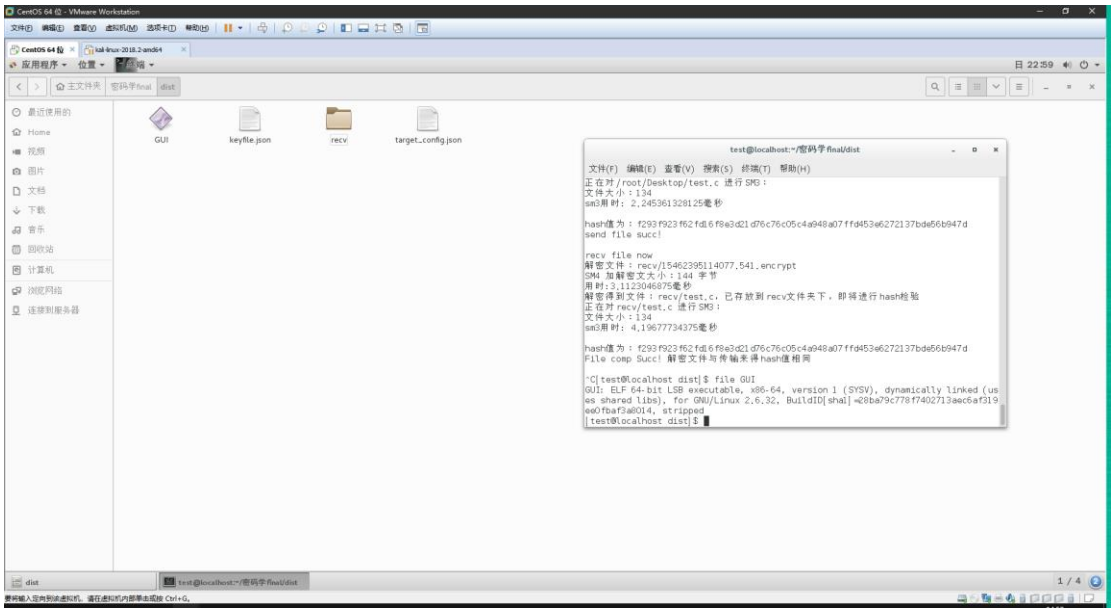
```
正在加密文件：C:\Users\肖\Desktop\新建文本文档 (4).txt
SM4 加解密文大小：208 字节
用时:2.996826171875毫秒
文件加密成功，加密文件暂存为：C:\Users\肖\Desktop\新建文本文档 (4).txt.encrypt
recv file now
正在对C:\Users\肖\Desktop\新建文本文档 (4).txt 进行SM3：
文件大小：176
sm3用时: 2.000732421875毫秒

hash值为：fc0ce816d8850b6bb0abdbd1eb5e4c2bf1bcc996a06388b6b7e37f19f490e61c
send file succ!

解密文件：recv/15460846412253.318.encrypt
SM4 加解密文大小：208 字节
用时:3.99755859375毫秒
解密得到文件：recv/新建文本文档 (4).txt，已存放到recv文件夹下，即将进行hash检验
正在对recv/新建文本文档 (4).txt 进行SM3：
文件大小：176
sm3用时: 3.0毫秒

hash值为：fc0ce816d8850b6bb0abdbd1eb5e4c2bf1bcc996a06388b6b7e37f19f490e61c
File comp Succ! 解密文件与传输来得hash值相同
```

六、　平台兼容性

支持 windows 和 linux

Linux 版本客户端在





我是在 centos 中搭好环境编译生成二进制文件，使用 pyinstall 编译，可以在 64 位 Linux 系统上运行。

Linux 版本与 Windows 版本操作方法一致。

六、教师评语 签名： 日期：	成绩