

万源传越科技

**万源传越——基于TCP协议RSA与Base64加密的聊天软件**

产品版本 1.1.1

文档版本 01

发布日期 2023-5-9

**版权声明**

版权所有 © 万源传越科技有限公司 2023。 保留一切权利（包括但不限于修订、 最终解释权）。 除非万源传越科技有限公司另行声明或授权，否则本文件及本文件的相关内容所包含或涉及的文字、图像、图片、照片、音频、视频、图 表、色彩、版面设计等的所有知识产权（包括但不限于版权、商标权、专利权、商业 秘密等）及相关权利，均归万源传越科技有限公司所有。未经万源传越科技有限公司书面许可， 任何人不得擅自对本文件及其内容进行使用（包括但不限于复制、转载、摘编、修改、 或以其他方式展示、传播等）

**特别提示**

您购买的产品、服务或特性等应受万源传越科技有限公司商业合同和条款的约束， 本文档中描述的产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，万源传越科技有限公司对本文档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因，本文档内容会不定期进行更新，如有变更，恕不另行通知。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保，万源传越科技有限公司不对本文档中的遗漏、变更及 错误所导致的损失和损害承担任何责任。

**联系我们**

售前咨询热线：18782522759

售后服务热线：13438304895 （中国大陆）

项目开发人员

吴秋林（负责人）:RSA算法，并在最后收工阶段将所有同学的代码进行汇总调试

肖逸峰:RSA算法

翁海鹏:RSA算法

朱俊霖:Base64算法，

熊浪:完成服务器端

谭嘉阳:完成客户端

谢睿:完成可视化界面设计

谢直鹏:完成可视化界面设计

文档版本 05 （2023-5-9）

一、产品简介

1、产品概述

万源传越科技有限公司开发的聊天软件是基于TCP协议RSA与Base64加密理念设计的产品，该产品作为一款适用于在局域网内部的通信工具，具备简单的文字聊天功能。整个系统由服务器端和客户端两部分组成。该产品是在IDEA开发环境中使用java语言实现的，开发产品使用的IntelliJ IDEA 2020.3.2 x64，jdk版本是jdk16.0.2\_x64

产品能在windows操作系统各个版本上运行。

2、产品背景

现阶段，网络用户数量不断增长，网络规模持续扩大，新型互联网业务层出不穷，但是由于时延和速率限制，外部公共网路无法完全保障局域网内部的通信需求和稳定，

面向工业互联网、天地一体化网络等重大需求，攻关超低时延、超高 通量带宽、超大规模连接的网络体系架构及关键技术是我们现在遇到的难题，尤其是在局域网内部的通信，我们是否需要一个新的通信机制和方法来保障内部网络的通畅。而且在聊天过程中，涉及到的信息可能会被黑客、病毒或者是第三方软件窃取，而加密技术可以有效地保护个人隐私和聊天信息的安全。具体来说，对聊天记录进行加密，可以避免聊天内容被黑客、间谍、劫持者等歹徒窃取，保证聊天信息的安全性，同时也可以防止其被网络服务提供商、政府等机构进行监控和追踪。此外，一些职业如律师、医生等可能会需要对部分敏感信息进行加密描述，以保护客户的隐私和安全。因此，对聊天进行加密可以提高对隐私、安全和信息保密的需求。基于socket通信的局域网通信可以很好的解决局域通信的难题。Socket通信优点明显：

（1）传输数据为字节级，传输数据可自定义，数据量小（对于手机应用讲：费用低）

（2）传输数据时间短，性能高  
（3）适合于客户端和服务器端之间信息实时交互  
（4）可以加密,数据安全性强

基于以上优势和社会需求，我们开发出了这款产品，在RSA与Base64的加密下，大力保障了通信的安全性和可靠性。

3、主要加密算法

算法RSA是一种公钥密码算法，它的名字由三位开发者，即Ron Rivest、Adi Shamir和Leonard Adleman的姓氏的首字母组成的。RSA被用于公钥密码和数字签名。RSA公开密钥密码体制是一种使用不同的加密密钥与解密密钥，“由已知加密密钥推导出解密密钥在计算上是不可行的”密码体制。

在公开密钥密码体制中，加密密钥（即公开密钥）PK是公开信息，而解密密钥（即秘密密钥）SK是需要保密的。加密算法E和解密算法D也都是公开的。虽然解密密钥SK是由公开密钥PK决定的，但却不能根据PK计算出SK [2] 。

按照RFC2045的定义，Base64被定义为：Base64内容传送编码被设计用来把任意序列的8位字节描述为一种不易被人直接识别的形式。原理是Base64编码方法要求把每三个8Bit的字节转换为四个6Bit的字节，其中，转换之后的这四个字节中每6个有效bit为是有效数据，空余的那两个 bit用0补上成为一个字节。因此Base64所造成数据冗余不是很严重，Base64是当今比较流行的编码方法，因为它编起来速度快而且简单。虽然编码后不是明文，看不出原文，但是解码也很简单。

3、产品用途

该产品主要面向的是公司、园区、办公楼等需要大量局域网通信的客户，我们在产品设计中围绕着解决大量局域网通信需求问题的思路，设计出了这一款聊天软件，方便用户在局域网的通信，在外网受影响或者外网速率慢、时延高时，仍然能保障内部网络通信的完整性。利用RSA和Base64加密技术，保证了通信的安全性。使用场景可以是公司管理人员与员工之间的交流、办公楼里同事与同事之间的聊天、学校里老师和学生之间的即时通信等多方面用途。

基本思路

要实现对聊天信息的加密解密，依据是socket通信中会将发送和接受收的消息封装成一个包，就可以把RSA和算法写成函数，通过调用函数来加密和解密socket通信中的包，来实现对传输消息加密解密。在RSA加密解密函数中先使用base64对明文进行编码，然后将base64编码后的结果进行加密解密。在通信端内，客户端和服务器可以分别在发送端发送消息的包中调用RSA加密函数来对消息加密，在接收端调用RSA解密函数来对消息解密。如此就可以实现通信中对消息的加解密，保证消息在传输过程中以密文的形式传输，在接受到消息的时候以明文显示。并且服务器和客户端各自的公钥也进过base64加密和传输。

1. 设计方案
2. 本产品使用基于TCP协议RSA与Base64加密实现通信

主要代码如下:

1. RSA加密算法类

首先我们引入了crypto包用于RSA设计



我们在RSAExample类中重定义了构造函数用于初始化公钥私钥。其次我们构造了Ecode解密函数和Decode加密函数。Ecode将字符串加密为byte数组，Decode将byte数组解密为字符串。

postPublicKey函数将对象的公钥Base64加密为字符串写到路径为String S的文件中并返回加密后的公钥字符串

getPublicKey函数将对象的公钥从文件中读取出来并将读取的字符串解密为Publickey对象并将RSAExample类的公钥赋值为得到的公钥，最后返回Publickey

import java.io.\*;  
import java.net.Socket;  
import java.security.\*;  
import java.security.spec.X509EncodedKeySpec;  
import java.util.Arrays;  
import java.util.Base64;  
import javax.crypto.Cipher;  
public class RSAExample implements Serializable {  
 private KeyPairGenerator keyGen;  
 private KeyPair keyPair;  
 PublicKey publicKey;  
 PrivateKey privateKey;  
 private Cipher cipher;  
 public RSAExample() throws Exception {  
 Security.*addProvider*(new org.bouncycastle.jce.provider.BouncyCastleProvider());  
 keyGen = KeyPairGenerator.*getInstance*("RSA", "BC");  
 keyGen.initialize(2048);  
 keyPair = keyGen.generateKeyPair();  
 publicKey = keyPair.getPublic();  
 privateKey = keyPair.getPrivate();  
 cipher = Cipher.*getInstance*("RSA/None/PKCS1Padding", "BC");  
 }  
  
 public byte[] Encode(String plainText,PublicKey publicKey1) throws Exception, IOException {  
 cipher.init(Cipher.*ENCRYPT\_MODE*, publicKey1);  
 byte[] cipherText = cipher.doFinal(plainText.getBytes("UTF-8"));  
 return cipherText;  
 }  
  
 public String Decode(byte[] cipherText) throws Exception {  
 System.*out*.println(privateKey);  
 cipher.init(Cipher.*DECRYPT\_MODE*, privateKey);  
 byte[] decryptedText = cipher.doFinal(cipherText);  
 String decryptedString = new String(decryptedText, "UTF-8");  
 System.*out*.println(decryptedString);  
 return decryptedString;  
 }  
  
// public static void main(String[] args) throws Exception {  
// RSAExample rsaExample=new RSAExample();  
// RSAExample rsaExample1=new RSAExample();  
// System.out.println(rsaExample.publicKey);  
// System.out.println(rsaExample1.publicKey);  
// String s="D:\\rsa1.txt";  
// String s2="D:\\rsa2.txt";  
// rsaExample.postPublicket(s);  
// rsaExample1.postPublicket(s2);  
// rsaExample.getPublicKey(s2);  
// rsaExample1.getPublicKey(s);  
// String ss="hello world";  
// byte[] c=rsaExample.Encode(ss, rsaExample.publicKey);  
// System.out.println(Arrays.toString(c));  
// byte[] c2=rsaExample1.Encode(ss, rsaExample1.publicKey);  
// System.out.println(Arrays.toString(c2));  
// System.out.println(rsaExample.Decode(c2));  
// System.out.println(rsaExample1.Decode(c));  
// }  
 public String postPublicket(String s) throws Exception{  
 File f=new File(s);  
 FileOutputStream o=new FileOutputStream(f);  
 OutputStreamWriter os=new OutputStreamWriter(o);  
 BufferedWriter ps = new BufferedWriter(os);  
 String ws= Base64.*getEncoder*().encodeToString(publicKey.getEncoded());  
 ps.write(ws);  
 ps.close();  
 return ws;  
 }  
 public PublicKey getPublicKey(String s) throws Exception {  
 File f=new File(s);  
 FileInputStream o=new FileInputStream(f);  
 InputStreamReader os=new InputStreamReader(o);  
 BufferedReader ps = new BufferedReader(os);  
 String ws= ps.readLine();  
 byte[] pub=Base64.*getDecoder*().decode(ws);  
 KeyFactory keyFactory=KeyFactory.*getInstance*("RSA");  
 X509EncodedKeySpec spec=new X509EncodedKeySpec(pub);  
 PublicKey p=keyFactory.generatePublic(spec);  
 publicKey=p;  
 ps.close();  
 return p;  
 }  
}

（2）在服务器类中定义了发送按钮和发送公钥按钮，定一了监听端口设置为9999，ServerSocket对象来实现服务器与客户端的通信连接

private JButton Sender = new JButton("发送");  
private JButton Sender2 = new JButton("发送公钥");  
//文本显示区域  
private JTextArea area = new JTextArea("聊天内容："+'\n');  
//发送文本区域  
private JTextField field = new JTextField("");  
private int Port = 9999;//端口  
private ServerSocket SS;  
public JTextField getField() {  
 return field;  
}

（3）在Server类的构造函数中初始化SeverSocket端口和Socket IP地址Server服务器IP为本机地址

try{  
  
 SS = new ServerSocket(Port);  
  
 socket = SS.accept();  
  
 String ipAddress = socket.getInetAddress().getHostAddress();  
  
 System.*out*.println("客户端" + ipAddress + "连接");  
  
}catch(Exception ex){  
  
}

（4）构造函数中初始化发送公钥按钮并为其加监听器用于把公钥发送到客户端

Sender2.setSize(80,30);  
  
this.add(Sender2,BorderLayout.*AFTER\_LINE\_ENDS*);  
  
Sender2.addActionListener((e -> {  
 try {  
 post();  
 }  
 catch (Exception ex) {  
 throw new RuntimeException(ex);  
 }  
  
}));

（5）post()传输将公钥加密后的字符串传给（客户端/服务器）（此函数服务器客户端都定义了）

public void post() throws Exception {  
 OutputStream os=null;  
 try {  
 os = socket.getOutputStream();  
  
 } catch (IOException r) {  
 throw new RuntimeException(r);  
 }  
 PrintStream ps = new PrintStream(os);  
 ps.println(rsa.postPublicket(F));  
  
}

（6）get()函数将接收到的公钥赋值给服务器/客户端的RSA对象的公钥

public void get() throws Exception {  
 InputStream is = null;  
 try {  
 is = socket.getInputStream();  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
  
 }  
  
 BufferedReader br = new BufferedReader(new InputStreamReader(is));  
 String str=br.readLine();  
 byte[] pub=Base64.*getDecoder*().decode(str);  
 KeyFactory keyFactory=KeyFactory.*getInstance*("RSA");  
 X509EncodedKeySpec spec=new X509EncodedKeySpec(pub);  
 PublicKey p=keyFactory.generatePublic(spec);  
 rsa.publicKey=p;  
}

（7）重定义按钮监听器用于发送信息

public void actionPerformed(ActionEvent e) {  
  
 String s2 = this.getField().getText();  
 OutputStream os=null;  
 try {  
 os = socket.getOutputStream();  
  
 } catch (IOException r) {  
 throw new RuntimeException(r);  
 }  
 PrintStream ps = new PrintStream(os);  
 //判断是否为空信息  
 if(s2.isEmpty()){  
 System.*out*.println("请输入有效信息");  
 }  
 else {  
// try {  
// byte[] ss=rsa.Encode(s2,rsa.publicKey);  
//// System.out.println(ss.toString());  
//// String s=new String(ss,"UTF-8");  
// String s= Base64.getEncoder().encodeToString(ss);  
// ps.println(s);  
// System.out.println(s);  
//// byte[] s=new String(ss,"UTF-8").getBytes();  
//// System.out.println(rsa.Decode(ss));  
// }  
// catch (Exception ex) {  
//  
// }  
 ps.println(s2);  
 System.*out*.println(s2);  
  
 }  
 }

（8）定义线程用于接收信息和公钥

class MyThread extends Thread {  
 @Override  
 public void run() {  
  
 try {  
 get();  
 }  
 catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
 InputStream is = null;  
 try {  
 is = socket.getInputStream();  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
  
 BufferedReader br = new BufferedReader(new InputStreamReader(is));  
  
 try {  
 //接受数据并解密  
 String str = br.readLine();  
 byte s[]= Base64.*getDecoder*().decode(str);  
 String c=rsa.Decode(s);  
 area.append("客户端说:"+c+'\n');  
 } catch (Exception e) {  
  
 }  
  
//  
 }  
 }

（9）客户端代码与服务器类似但其中多定义了一个线程类用于登陆成功后的连接

class thread extends Thread {  
 @Override  
 public void run() {  
 System.*out*.println("");  
 if(Server.*f*==true) {  
 Client Cl = new Client();  
 }  
 }

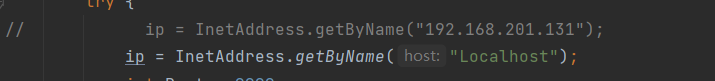
（10）登陆界面监听器用于登陆成功后将客户端显示出来（主要将客户端运行条件设为true）

jbu.addActionListener((e -> {  
 String name = jtf.getText();  
 String password=jtf2.getText();  
 //验证账号和密码，如果正确显示登陆成功的界面  
 //账号123对应密码789  
 if("admin".equals(name)&&"123456".equals(password)){  
 //  
  
// jf.setVisible(false);  
 Server.*f*=true;  
 jf.setVisible(false);  
 System.*out*.println(Server.*f*);  
 JOptionPane.*showMessageDialog*(null, "登陆成功", "Title",JOptionPane.*PLAIN\_MESSAGE*);  
  
 //新的界面（跟旧的差不多后面有代码）  
  
 }else{  
 //登录失败界面写这里  
 JOptionPane.*showMessageDialog*(null, "用户名或密码错误", "Title",JOptionPane.*WARNING\_MESSAGE*);  
  
 }  
 }));

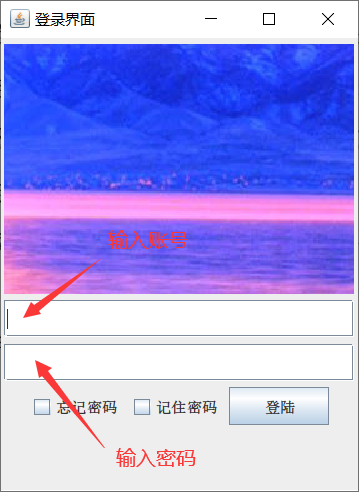
1. 产品使用手册

默认本机通信

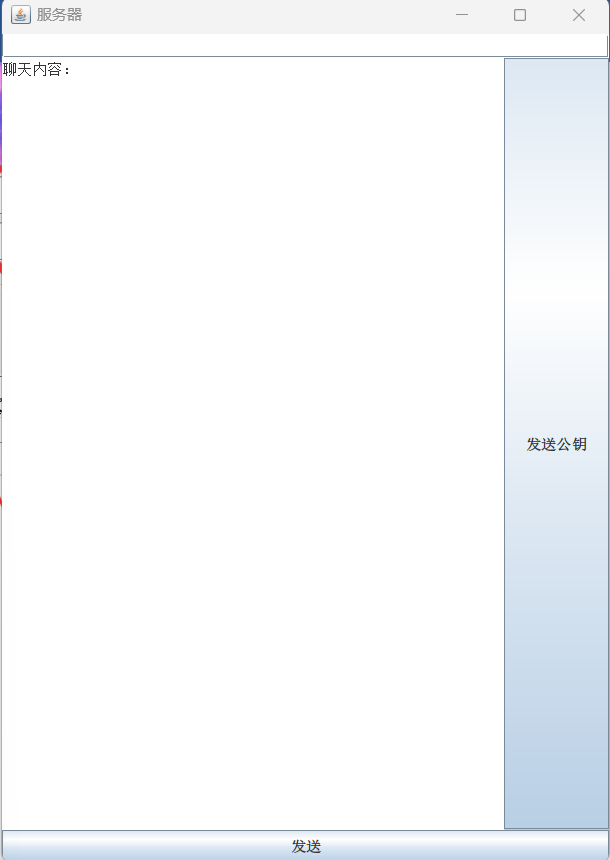
若要局域网通信将Client类中构造函数的IP地址改为服务器地址



1、输入账号和密码（账号默认admin，密码默认123456）



1. 服务器点击发送公钥

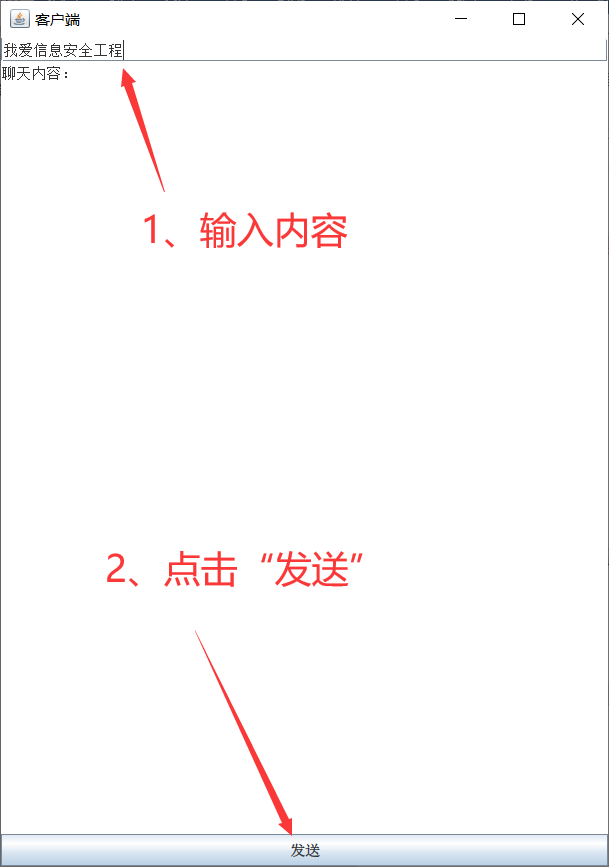


3、客户端点击发送公钥



4、客户端输入内容，点击“发送”，服务器端接收到内容

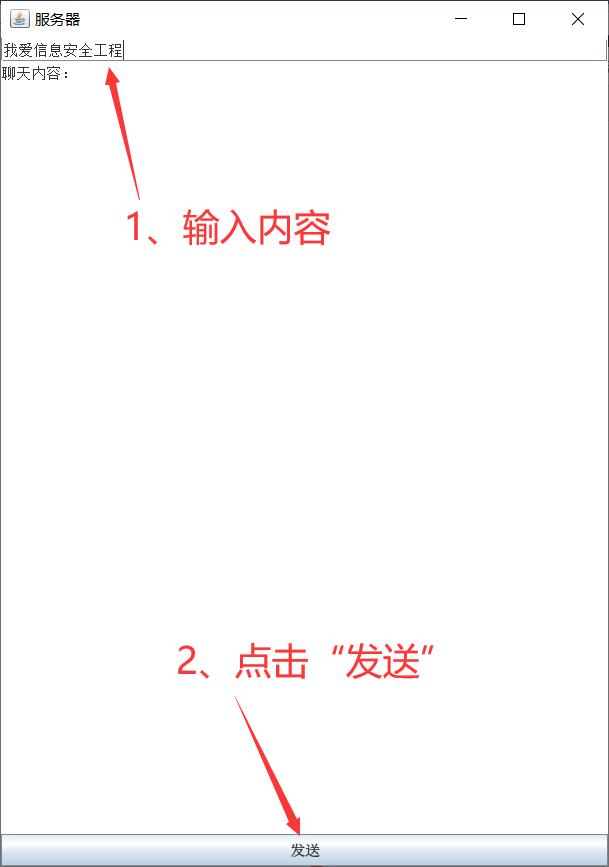






5、服务器端发送内容，点击“发送”，客户端接收到内容







测试报告

测试大纲：

测试依据：该聊天程序基于TCP协议，使用RSA与Base64加密算法进行加密通信。对于基于TCP协议的聊天程序，我们需要对连接建立、数据传输、稳定性和容错性进行测试；对于RSA加密和Base64编码，我们需要测试程序能够正确加密解密和编码解码，以保证通信的安全性和数据的完整性。

TCP协议测试依据：TCP协议是一种面向连接的协议，保证数据传输的可靠性和顺序性。因此，我们需要测试程序能够正确建立和维护连接，发送和接收数据的顺序是否正确，以及在网络状况不稳定的情况下，程序的稳定性和容错性。

RSA加密测试依据：RSA加密算法是一种非对称加密算法，可以保证通信的安全性。我们需要测试程序能够正确生成RSA密钥对，并且能够使用公钥加密消息、使用私钥解密消息。同时，还需要测试程序能够处理加密后消息长度增加的情况，以及在密钥不安全的情况下，程序的安全性能如何。

Base64编码测试依据：Base64是一种常用的编码方式，可以将二进制数据转换成可打印字符，方便传输和存储。我们需要测试程序能够正确进行Base64编码和解码，保证数据的完整性和准确性。

测试方案：

对该聊天程序进行功能测试、性能测试、安全测试和用户体验测试。因为进行功能测试、性能测试、安全测试和用户体验测试是确保基于TCP协议的RSA和Base64加密的聊天程序质量和安全性的关键步骤，可以有效提高程序的可靠性和用户满意度。

评价标准：

功能测试：主要关注程序的基本功能是否正常，包括消息发送和接收、建立连接和断开连接等。功能测试需要验证程序是否符合用户需求和功能要求，并检查是否有任何错误或缺陷。测试人员可以模拟各种情况，包括正常情况、异常情况和边界情况，以确保程序在各种情况下都能正常工作。

性能测试：主要关注程序的性能是否满足实际使用需求，包括消息传输速度、延迟时间、系统资源利用率等。性能测试需要检查程序在高负载情况下是否能够正常工作，并确定程序能够支持的最大用户数等。测试人员可以使用各种工具和方法，例如负载测试、压力测试等，以确定程序的性能瓶颈和优化方向。

安全测试：主要关注程序的安全性能，包括加密算法的安全性、传输过程中的数据保护和防止攻击等。安全测试需要检查程序是否容易受到攻击，如中间人攻击、窃听、篡改等，以及是否能够保护用户的隐私和数据安全。测试人员可以使用各种工具和方法，例如漏洞扫描、网络分析等，以确定程序的安全性能和风险。

用户体验测试：主要关注程序的用户体验，包括操作界面的友好性、交互方式的便捷性、响应速度和用户反馈等。用户体验测试需要检查程序是否符合用户的期望和需求，并提供良好的使用体验。测试人员可以从用户角度出发，使用程序进行实际操作，并记录用户反馈和体验感受，以确定程序的用户体验瓶颈和改进方向。

测试结果：

功能测试结果：经过测试，该聊天程序能够正常实现基本聊天功能，包括发送和接收消息，符合预期要求。

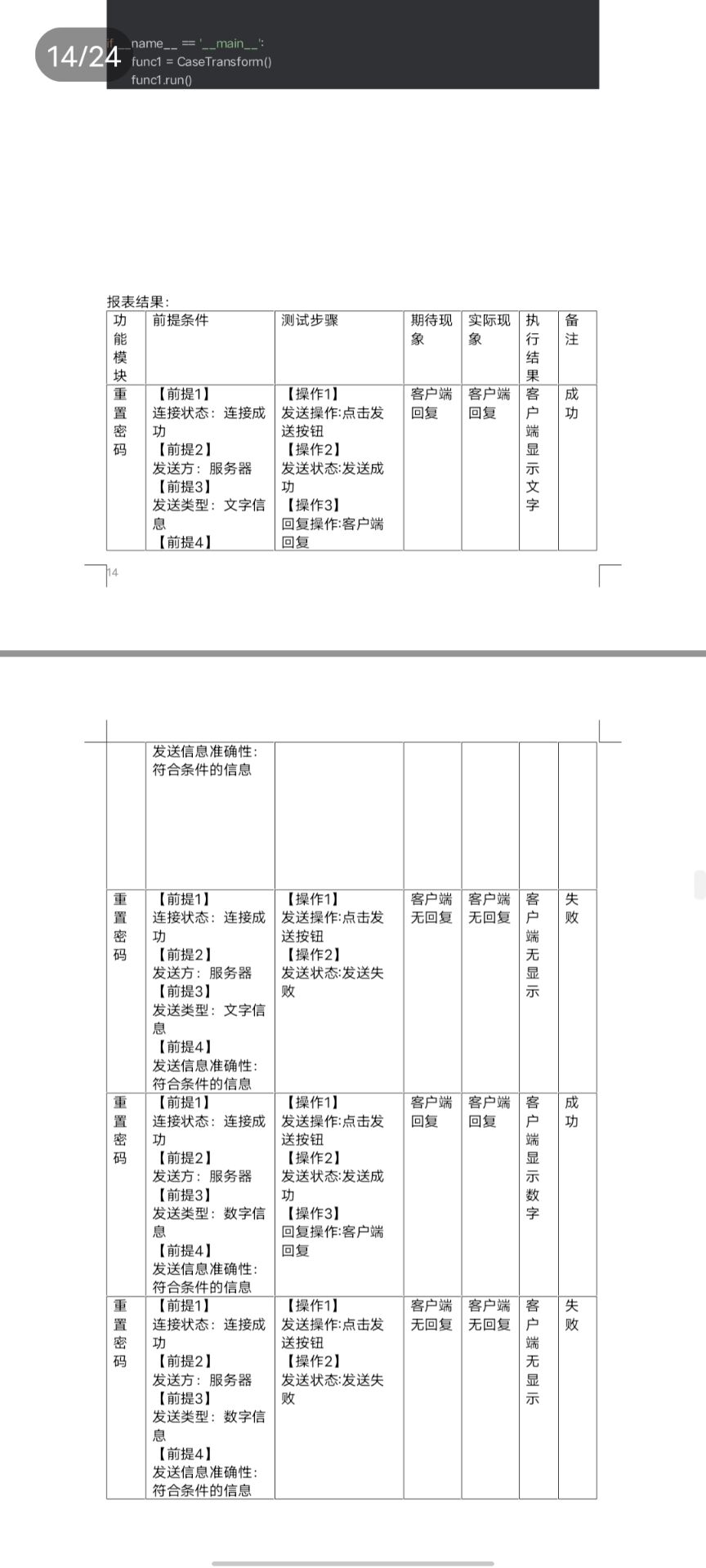
性能测试结果：经过测试，该聊天程序的性能能够满足实际使用需求，消息发送和接收的延迟时间均在可接受的范围内。

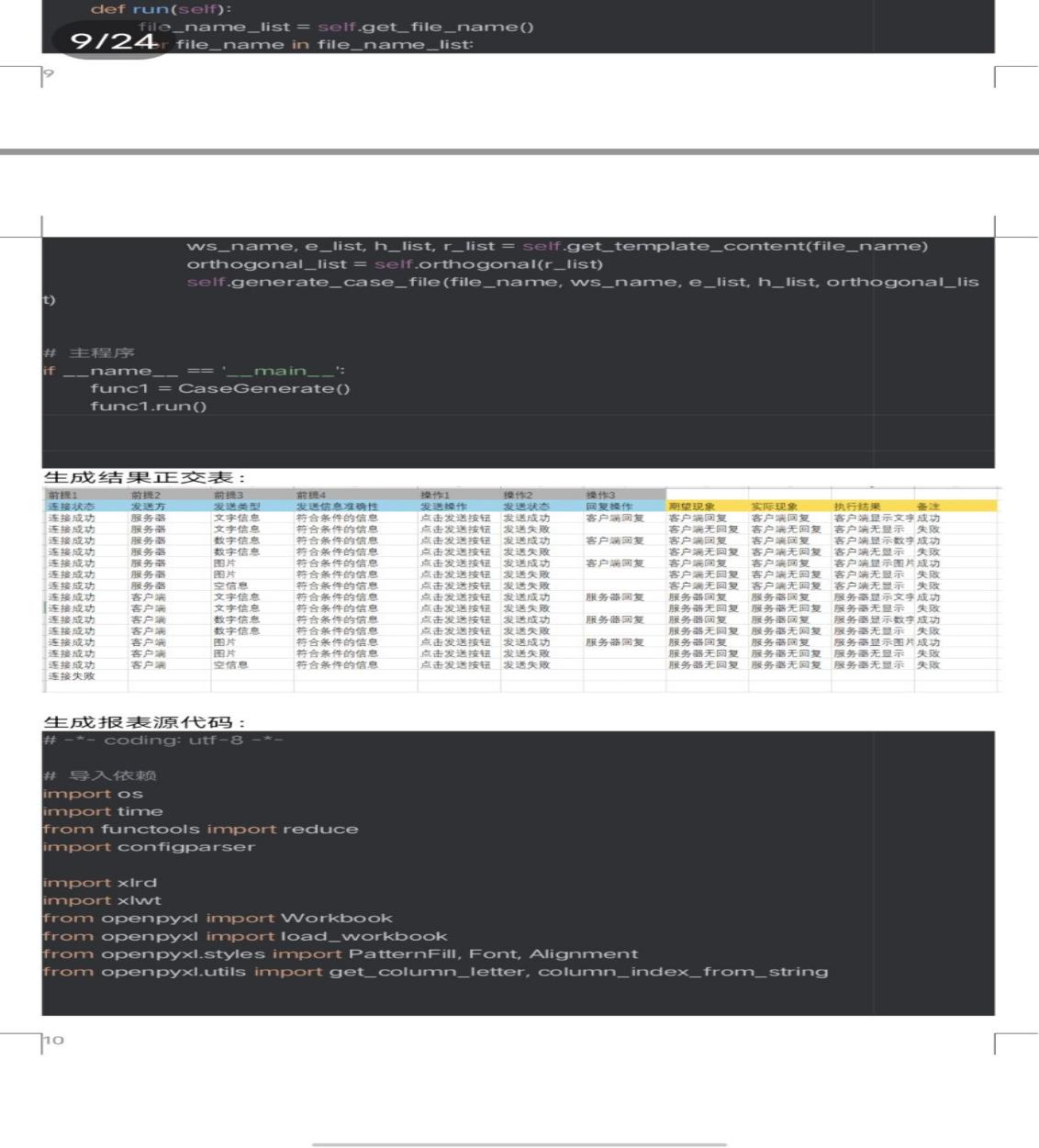
安全测试结果：经过测试，该聊天程序的加密算法能够保障通信安全，加密后的消息无法被窃听和篡改。

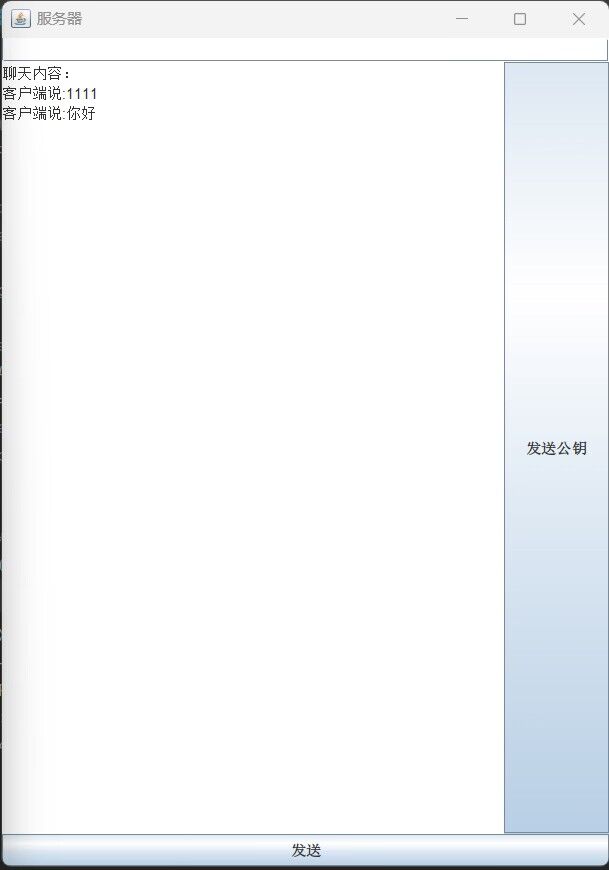
用户体验测试结果：经过测试，该聊天程序的操作界面友好，操作简便，符合用户体验要求。

过程展示：













综合分析：本小组人员参与评分

满分100，8人均12.5，介于负责人工作量极大，加2分