**应用密码学课程设计任务书**

1. **题目**

**简单保密通信系统设计与实现**

**二、实训的性质和任务**

1、巩固和强化学生的应用密码学基本知识，尤其是密码学中的：数据保密性、数据完整性、数据源认证、身份认证等安全性需求的概念和实现手段；对称加密、公钥加密、Hash函数、数字签名等密码学基本概念和基础方案。

2、培养和训练学生综合利用所学密码学基本知识和基本程序设计能力进行综合应用的能力。

3、培养学生适应岗位需要，独立分析问题、解决问题的基本能力，以及团队合作精神。

**三、实训的基本要求**

通过本次实训，学生应达到以下几个方面的要求：

1、素质要求

以积极认真的态度对待本次实训，遵章守纪。善于发现问题，分析问题，解决问题，努力培养自己的独立工作能力。

2、知识要求

熟练掌握密码学中的数据保密性、数据完整性、数据源认证、身份认证等安全性需求的概念和实现手段；

熟练掌握和应用对称加密、公钥加密、Hash函数、数字签名等密码学基本概念和基础方案。

3、能力要求

学生应具有一定的项目分析能力、任务分解能力、组织实施能力以及理论联系实际应用的能力。

学生应具有一定的查阅教材、各类相关资料及工具书的能力，进而养成自我学习的习惯。

目录

[1 课程设计简介 4](#_Toc135648452)

[1.1 课程设计目标 4](#_Toc135648453)

[1.2 任务简介 4](#_Toc135648454)

[1.3 主要内容 5](#_Toc135648455)

[1.4 对课程设计报告的要求 5](#_Toc135648456)

[2 总体设计流程 7](#_Toc135648457)

[2.1总体设计思路 7](#_Toc135648458)

[2.1.1 保密通信系统原理 7](#_Toc135648459)

[2.1.2 保密通信系统模型 7](#_Toc135648460)

[2.2 各模块功能设计 7](#_Toc135648461)

[3 系统详细设计 9](#_Toc135648462)

[3.1 ××××模块整体设计 9](#_Toc135648463)

[3.2 ××××模块中各函数设计 9](#_Toc135648464)

[3.2.1 ××××函数设计 9](#_Toc135648465)

[3.2.2 ××××函数设计 9](#_Toc135648466)

[3.2.3 ××××函数设计 9](#_Toc135648467)

[3.3 ××××模块整体设计 9](#_Toc135648468)

[3.4 ××××模块中各函数设计 10](#_Toc135648469)

[3.4.1 ××××函数设计 10](#_Toc135648470)

[4 系统的实现 11](#_Toc135648471)

[5 总结 11](#_Toc135648472)

[6 参考文献 11](#_Toc135648473)

# 需求分析：

要求：

以DES、AES、SM4或者祖冲之算法为数据加密算法；以RSA或者ElGamal算法为对称密钥传输的公钥加密算法；以数据认证算法MAC或SHA-256等Hash算法为消息认证算法；以DSA或者SM2算法为数字签字方法；选择适当的公钥管理方案和密钥协商方案，实现一套简单的保密通信系统。

设计一个简单的保密通信系统，并**模拟**一次该系统下的信息传输过程，发送消息的内容为：由学号、姓名、班级构成的字符串。系统配置从以下表格中选择。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对称加密算法 | 对称密钥协商 | | | 消息认证算法 |
| 公钥加密算法 | 数字签名算法 | 公钥证书 |
| AES | RSA | RSA | 简单公钥证书系统 | HMAC |
| DES | ElGamal | DSA | openssl | CBC-MAC |
| SM4 | SM2 | SM2 |  | SHA256 |
| 祖冲之算法 |  |  |  |  |

需求分析的写作可以按照以下结构展开：

1. 引言：

- 对系统的简要介绍和背景说明。

- 提及设计目标和要求。

2. 功能需求：

- 列出系统所需实现的功能需求，包括算法选择和加密/解密操作。

- 对每个功能需求进行详细描述，包括输入、输出和操作步骤。

3. 数据需求：

- 定义系统中涉及的数据元素和数据格式。

- 描述消息内容的结构和要求。

- 讨论密钥的生成、存储和管理。

4. 安全性需求：

- 确定系统的安全性要求，包括机密性、完整性和真实性。

- 讨论如何保证密钥的安全性和防范攻击。

5. 性能需求：

- 讨论系统的性能要求，包括加密效率和认证效率。

- 定义性能指标和限制。

6. 其他需求：

- 讨论与系统相关的其他需求，如易用性、可扩展性、可维护性等。

7. 系统约束：

- 讨论可能对系统设计和实现产生影响的约束条件，如硬件、软件、网络环境等。

8. 附录：

- 提供相关支持材料，如系统流程图、数据结构定义等。

在需求分析的过程中，要确保清晰、具体、可衡量的需求，以便后续的系统设计和开发工作能够基于这些需求进行。同时，需求应该能够满足系统的目标和用户的期望，同时考虑到安全性和性能方面的要求。

1. 引言：

本文将要设计的是一套基于加密算法、公钥加密算法、消息认证算法、数字签名算法、公钥证书和密钥协商方案的简单保密通信系统。该系统旨在提供保密性、完整性和真实性保障，以保护用户传输的敏感数据不受攻击者篡改或窃取。

2. 功能需求：

系统需要实现以下功能需求：

1. 使用AES、RSA、SM4或者祖冲之算法作为对称加密算法，对消息内容进行加密和解密操作。

2. 使用RSA或者ElGamal算法作为公钥加密算法，进行对称密钥协商。

3. 使用MAC或SHA-256等Hash算法作为消息认证算法，验证消息的真实性。

4. 使用DSA或者SM2算法作为数字签名算法，对消息进行数字签名。

5. 使用适当的公钥管理方案和密钥协商方案，保证密钥的安全性。

6. 提供简单的公钥证书系统，以验证对方公钥的真实性。

具体的操作流程为：

- 生成公钥和私钥，并将公钥传给通信对方；

- 通信对方使用公钥加密消息并发送给本方；

- 本方使用私钥解密消息，同时使用MAC或者Hash算法验证消息的真实性；

- 本方对消息进行签名并传回给通信对方；

- 通信对方使用本方的公钥对签名进行验证并使用本地私钥解密收到的消息。

3. 数据需求：

在本系统中，消息内容由学号、姓名、班级构成的字符串组成。对称加密算法和公钥加密算法所需的密钥将采用随机生成的方式进行。对称加密算法和消息认证算法所使用的密钥需要保存在本地，而公钥最好通过公钥证书的方式进行验证。用户的公钥和私钥需要保存在本地，并需要采取相应的措施保证其安全性。

4. 安全性需求：

本系统需要满足机密性、完整性和真实性需求。对称加密算法和公钥加密算法的密钥需要保证其随机性和唯一性，以防止攻击者进行字典攻击或重放攻击。消息认证算法需要使用MAC或Hash算法进行认证，以防止攻击者对消息进行篡改。数字签名算法要求使用DSA或SM2算法，并确保签名的唯一性和正确性。公钥证书系统需要对公钥进行验证，以防止中间人攻击或伪造证书带来的安全风险。

5. 性能需求：

本系统需要满足高效、低延迟的需求。对称加密算法和消息认证算法所需的密钥需要在本地保存，以提高加解密的效率。同时，需要对算法进行优化，选择适当的算法和参数，以提高加解密和数字签名的速度。

6. 其他需求：

本系统需要满足易用性、可扩展性、可维护性等需求，采用友好的用户界面，以方便用户进行操作。同时，系统应该便于扩展和维护，以适应未来的需求变化。

7. 系统约束：

本系统需要满足硬件、软件、网络等方面的约束条件。硬件方面需要满足加解密算法的计算资源要求，软件方面需要满足库文件的支持和兼容性，网络方面需要满足通信的可靠性和稳定性。

8. 附录：

本文档提供了详细的技术支持，包括系统流程图、数据结构定义等，以帮助用户更好地理解系统的实现。同时，提供了代码实现和操作说明，以方便用户进行测试和运行。

# 1 课程设计简介

## 1.1 课程设计目标

以DES、AES、SM4或者祖冲之算法为数据加密算法；以RSA或者ElGamal算法为对称密钥传输的公钥加密算法；以数据认证算法MAC或SHA-256等Hash算法为消息认证算法；以DSA或者SM2算法为数字签字方法；选择适当的公钥管理方案和密钥协商方案，实现一套简单的保密通信系统。

## 1.2 任务简介

设计一个简单的保密通信系统，并**模拟**一次该系统下的信息传输过程，发送消息的内容为：由学号、姓名、班级构成的字符串。系统配置从以下表格中选择。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对称加密算法 | 对称密钥协商 | | | 消息认证算法 |
| 公钥加密算法 | 数字签名算法 | 公钥证书 |
| AES | RSA | RSA | 简单公钥证书系统 | HMAC |
| DES | ElGamal | DSA | openssl | CBC-MAC |
| SM4 | SM2 | SM2 |  | SHA256 |
| 祖冲之算法 |  |  |  |  |

设计要求：

① 1-3人一组完成本次课程设计（最多3人）

② 从上表中选择适当的算法和方案设计一套完整的保密通信系统，要求通信双方在该系统下可以实现安全的保密通信。

③ 在设计部分详细描述保密通信系统模型、所使用的算法模块以及进行一次保密通信的工作流程。

④ 在实现部分钥对加密、消息认证、数字签字等算法给出具体的参数设置和安全级别的评估。

⑤ 使用指定的消息内容，模拟一次完整的消息发送过程，包括：公钥证书发放、密钥协商、消息加密、消息认证和数字签字的生成、数字签字的验证、消息解密等。

⑥ 如果能够实现并使用上表中的紫色算法可以获得加分。

⑦ 如果能够使用网络编程技术真正实现（而非模拟）一个保密通信系统也可以获得加分，网络编程部分可以使用python、java等语言实现。

⑧ 密码算法部分必须使用C、C++实现，不得调用密码学库函数。

## 1.3 主要内容

1、完成系统的总体设计，包括设计系统的总体框架，列出系统所包含的模块以及各模块之间的关系等。

2、完成系统的详细设计，包括设计系统中所要实现的方案和算法，各算法需要使用的库、数据结构、类、主要函数的设计。

3、完成系统的实现，根据设计内容逐一编写每个函数，完成模块设计功能。

4、按步骤运行各个程序，模拟整个系统的运行过程，实现保密通信的目标。

## 1.4 对课程设计报告的要求

1、课程设计报告是体现和总结课程设计成果的载体，主要内容要包括总体设计、详细设计、设计的实现、系统运行等部分。

2、在报告的各部分适当位置要配合相应的系统模型图、功能模块图、算法流程图等图表进行说明。

3、此外，报告还需要记录遇到的问题及解决方法、设计总结、参考文献等。

4、报告要做到内容正确完整，文理通顺，一般不应少于3000字，装订整齐。

5、设计总结部分主要写本人完成工作简介以及自己的设计体会，包括通过课程设计学到了什么，哪里遇到了困难，解决的办法以及在应用密码学方向今后的目标。

# 2 总体设计流程

## 2.1总体设计思路

### 2.1.1 保密通信系统原理

介绍对称钥加密、公钥加密、数字签名、hash函数和消息认证算法的功能、优点、缺点。进而介绍怎样合理使用它们，实现一次点到点（A到B）的保密通信。

解决下面几个问题：

1.用什么方法保证消息的保密性？（对称钥加密）

2.用什么方法来使得AB双方共享一次会话所使用的对称密钥？（密钥协商）

3.密钥协商过程中双方会话的安全性如何保证？（端到端的密钥协商协议，使用公钥密码（公钥加密、数字签名）来保证）

4.公钥的安全性如何保证？（证书系统）

5.证书系统怎样设计和实现？证书的内容是什么？

证书通过模拟文件传输实现，转载了签名后的公钥

6.保密通信中消息的完整性如何保证？（hash函数或者MAC）

### 2.1.2 保密通信系统模型

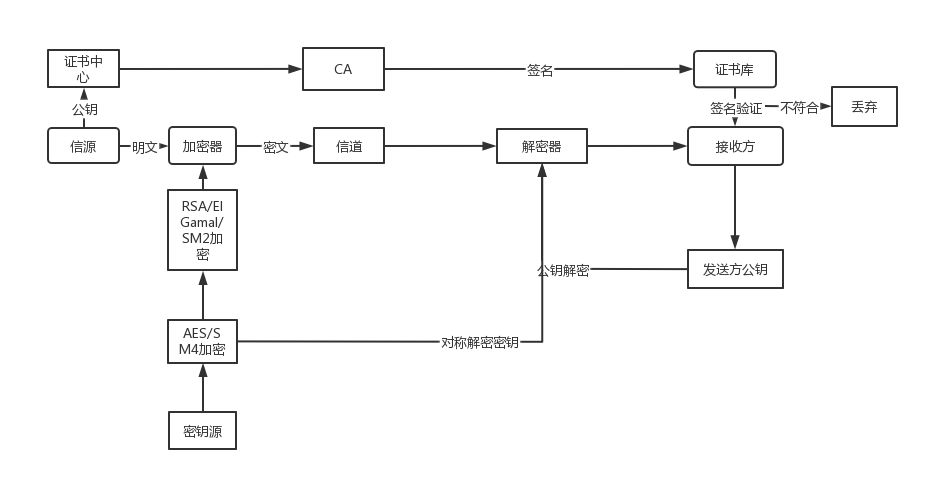


图2-1系统模型图

## 2.2 各模块功能设计

SM4/SM2 :谭嘉阳

SM4对称加密和SM2公钥加密

RSA/Socket/CA:吴秋林

RSA（签名和公钥加密）实现和CA模拟

Socket传输实现（JAVA实现Socket,采只用了JAVA的RSA加密）

AES/ElGamal/SHA225:肖逸峰

AES对称加密，ELGmal公钥加密（包括签名）,SHA256算法

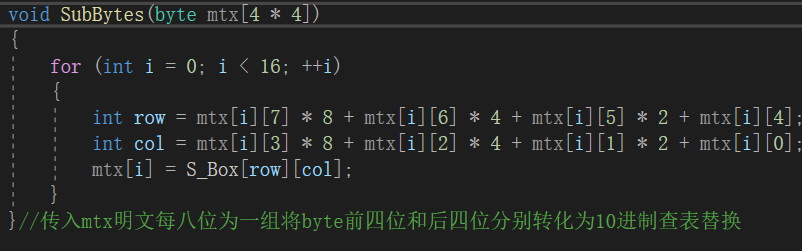
# 3 系统详细设计

## 3.1 AES模块整体设计

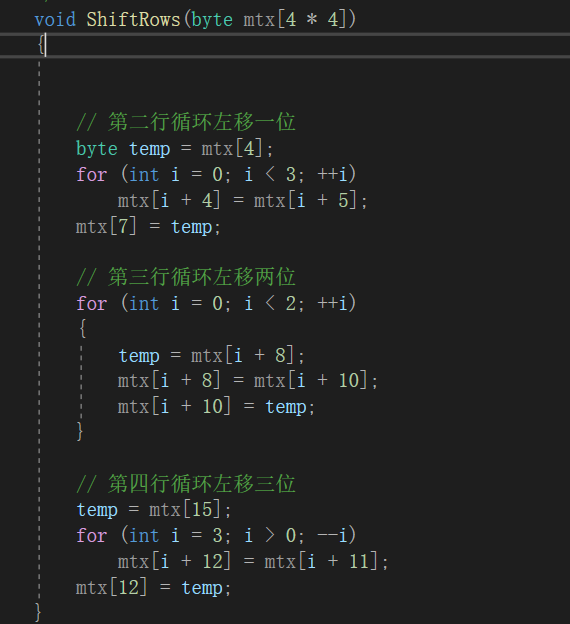
本模块的主要功能是实现AES对称加密算法，计划使用bitset库中的二进制模板类的结构来实现，主要包含5个函数，分别实现位运算、移位、类创建、二进制字符转数字功能。

## 3.2 AES模块中各函数设计

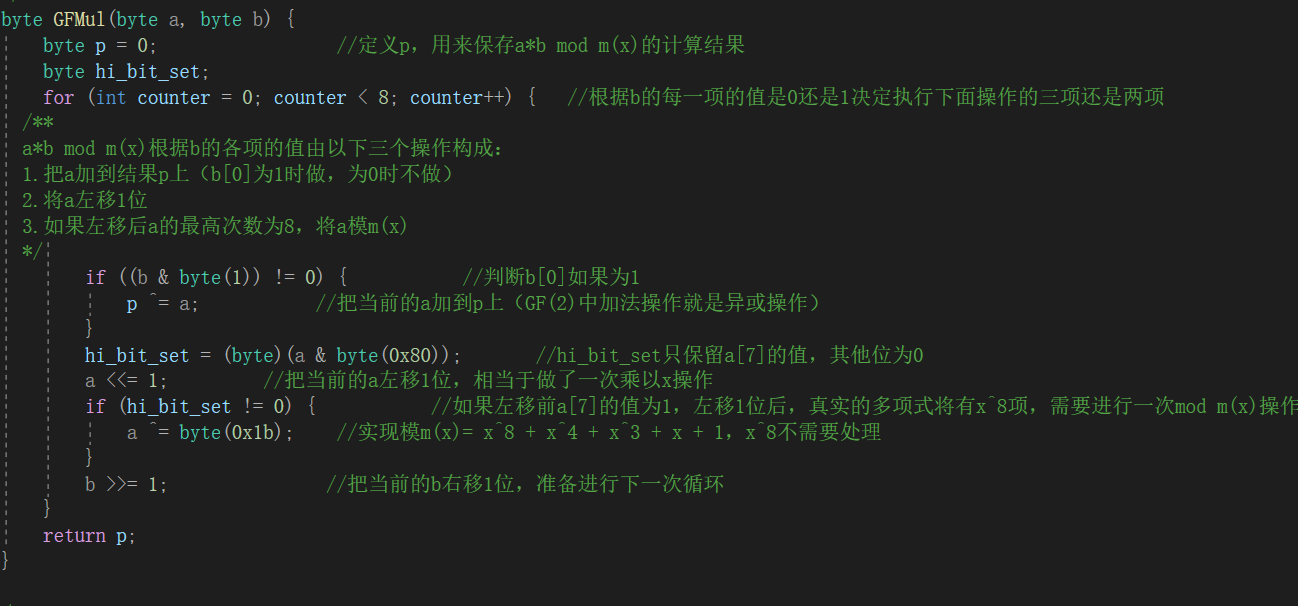
S盒替换



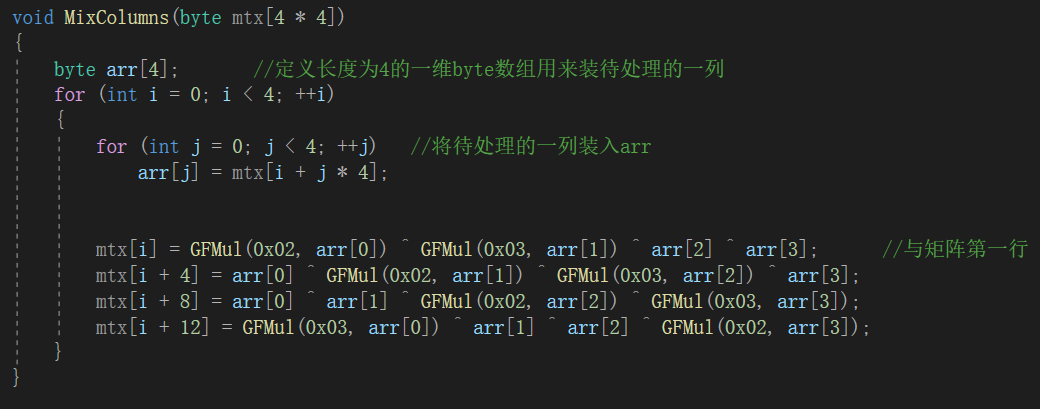
行移位



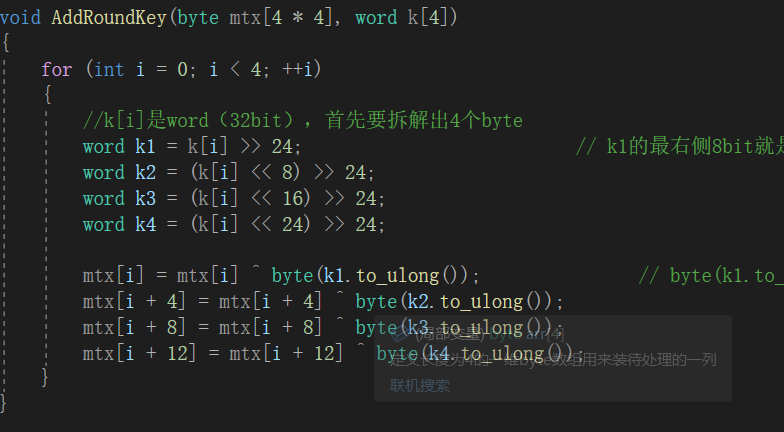
GF(2^8)运算函数



列混合



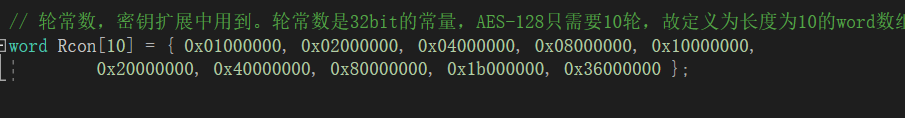
轮密钥加



### 3.2.1 AES加密函数设计

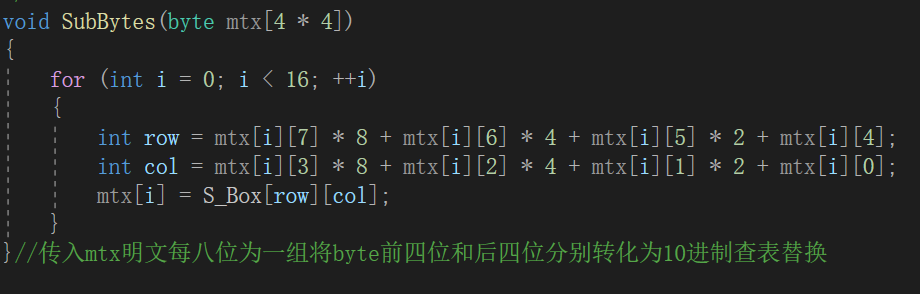
本函数的主要功能是实现AES加密，输入为4\*4的word矩阵，输出为4\*4word矩阵。核心算法是S盒替换、行移位、列混合，主要使用的数据结构为数组，栈。

1、加密函数步骤如下：

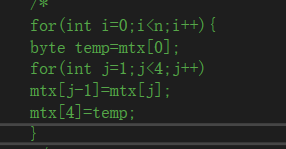


1. 编写一个函数将明文替换

比如0x12明文通过S加密为0XC9,明文前四位表示加密S盒的行后四位表示列。

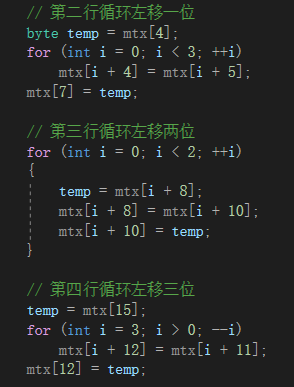


1. 将得到的替换结果向左移位，第一行不变，第二行左移一位，第三行左移2位， 第四行左移3位



若按二维数组形式传参则函数如图伪代码n表示左移位数

本代码传一个4\*4的一维数组需对不同行不同分析。



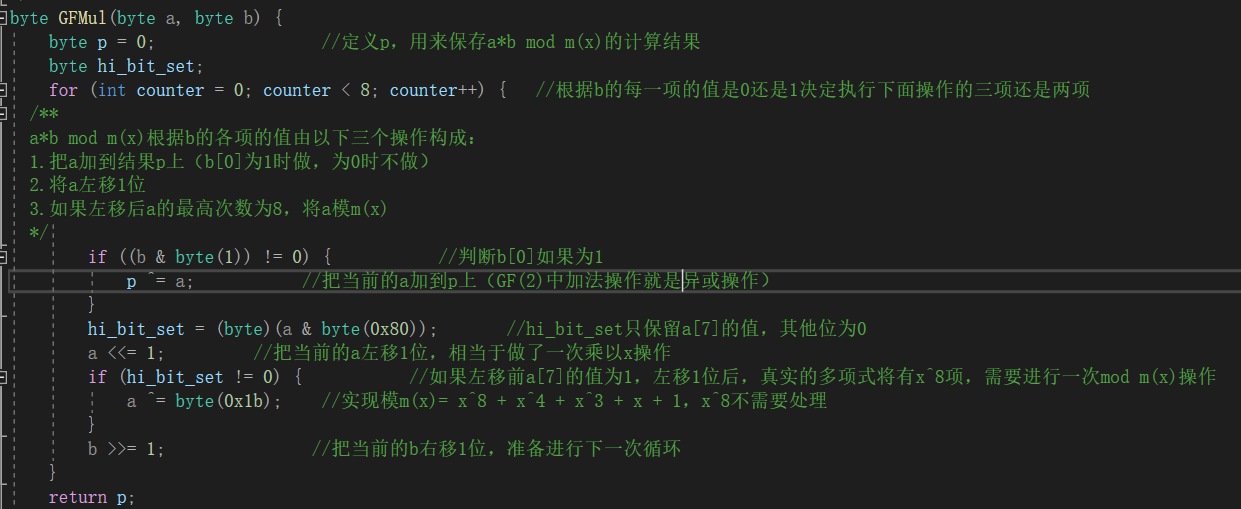
（3）下面定义GFMul函数，实现有限域上的乘法 GF(2^8)，在列混合这一步骤中使用函数输入为两个GF(2^8)中元素a和b，输出为a\*b mod m(x)（模m(x)的多项式乘法）其中a的多项式形式是

a[7]x^7+a[6]x^6+a[5]x^5+a[4]x^4+a[3]x^3+a[2]x^2+a[1]x+a[0]

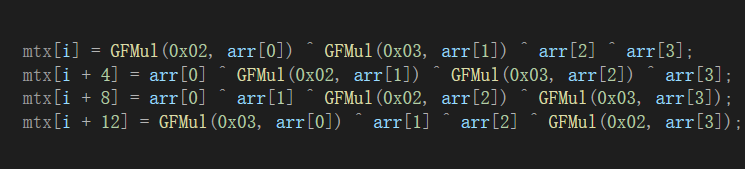
定义p存放结果

算法：具体可以化为b\*a（x）取余求和，b是方程b的每一项a(x)就是方程a。

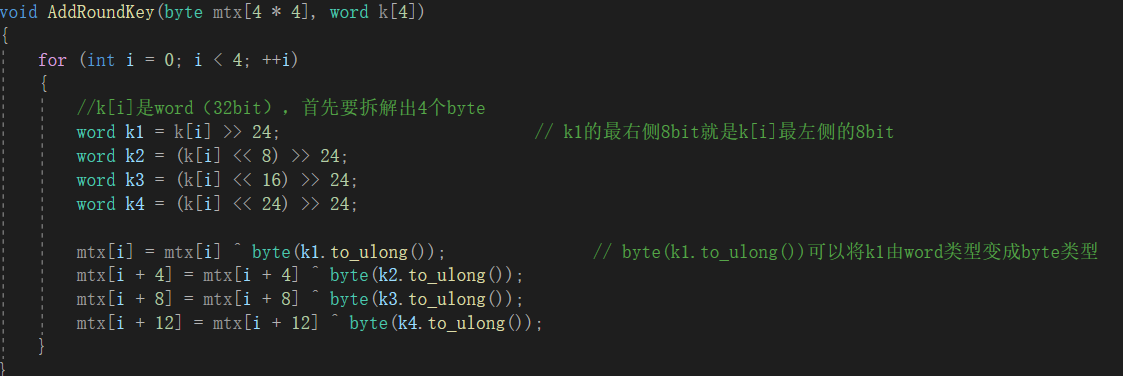
当b为1时直接把当前的a加到p上（GF(2)中加法操作就是异或操作）b最低位结果得到后算b的高一位相当于a(x)\*x相当于a左移一位，如果a原来的最高位为1左移后相当于出现x^8需要a mod m(x) ，最后b右移继续循环



1. 列混合将行移位后的每列与给定矩阵在GF(2^8)上做乘法得到新的混合后的一列。



1. 轮密钥加变换 - 将每一列与扩展密钥进行异或



### 3.2.2 AES解密函数设计

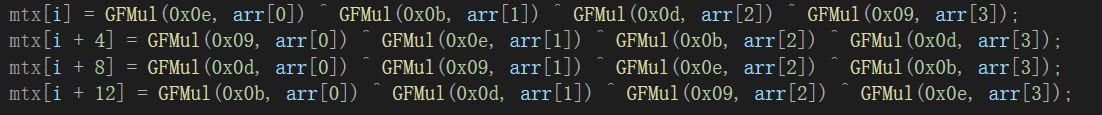
本函数的主要功能是实现AES解密，输入为4\*4的wrod密文，输出为4\*4的word明文。核心算法是S盒变换，逆行变换，列混合，主要使用的数据结构为循环，数组。

1. 解密函数步骤如下：

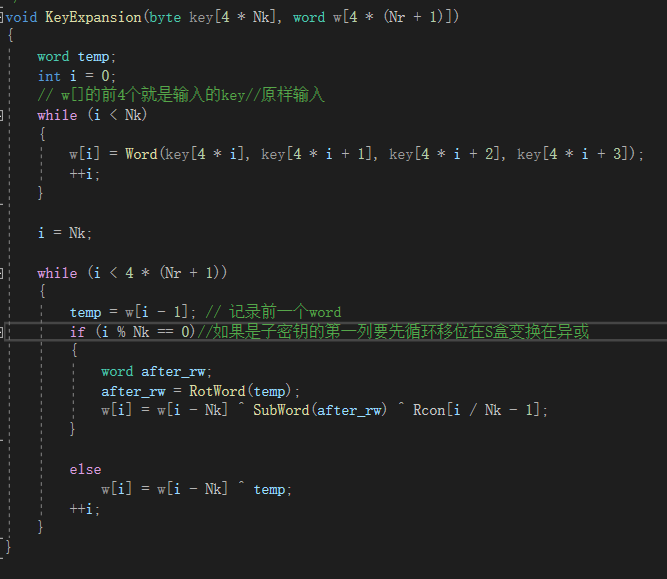
（1）逆S盒变换，和加密函数相似

（2）逆行变换，以字节为单位右移，算法与正变换类似

（3）列混合将行移位后的每列与给定矩阵在GF(2^8)上做乘法得到新的混合后的一列。变换矩阵与正变换不同。

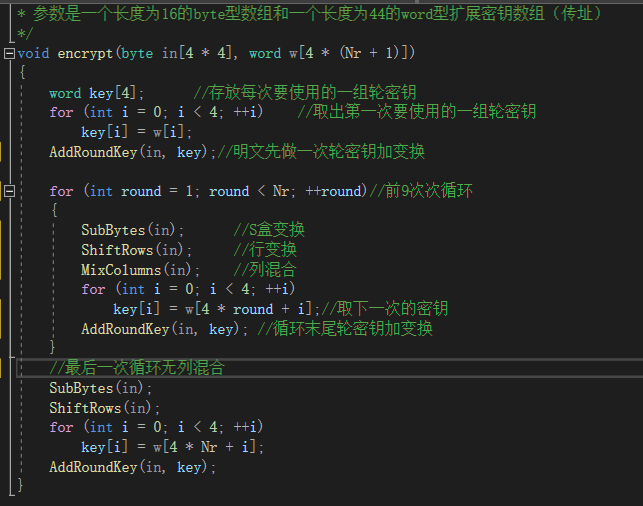


1. 密钥扩展函数步骤如下：
2. 将128bit扩展为11轮128bit

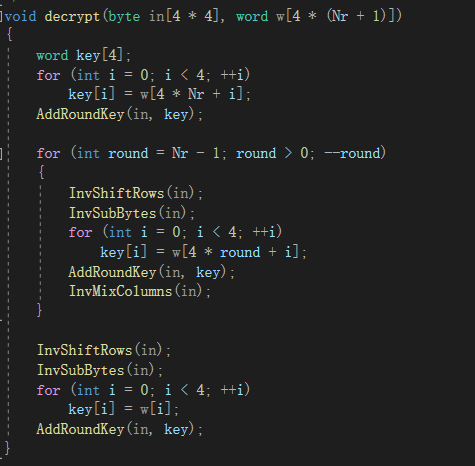


### 3.2.3 加解密总函数设计

本函数的主要功能是实现加密解密，调用以上函数。



解密函数与加密类似



## 3.3 CA模块整体设计

本模块的主要功能是实现CA证书模拟，计划使用NTL库中的ZZ.h结构来实现，主要包含2个函数，分别传输和接受功能。

## 3.4 CA模块中各函数设计

1、发送公钥

void CAIN(ZZ publickey, ZZ privatekey)

2.接受公钥

void CAOUT(RSA rsa)

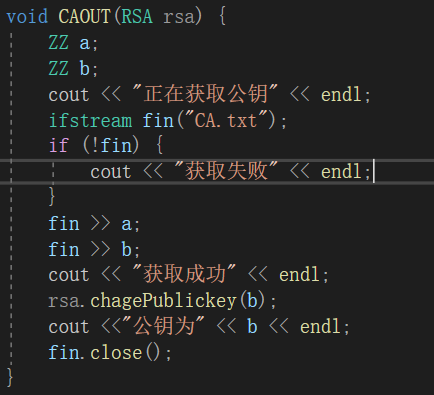
### 3.4.1 CAIN函数设计

函数接受公钥和私钥，将公钥传输到一个文件中



### 3.4.2 CAOUT函数设计

函数传入一个RSA对象，将文件的公钥传给对象



## 3.5 SOCKET模块整体设计

本模块的主要功能是实现SOCKET传输采用JAVA实现，计划使用SOCKET库和crypto，主要包含2个函数，分别传输和接受功能。

## 3.6 SOCKET模块中各函数设计

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

调用了bcprov-ext-jdk15to18-173.jar库

RSA库使用

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 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;

}

}

### 3.6.1服务器函数实现

（1）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;

}

（2）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){

}

（3）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);

}

}));

（4）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));

}

（5）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;

}

（6）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 {

ps.println(s2);

System.out.println(s2);

}

}

（7）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) {

}

}

}

### 3.6.2客户端代码实现

（1）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();

}

}

（2）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);

}

}));

# 4 系统的实现

## 通信系统主函数

#include "AES.h"

#include "RSA.h"

#include "CA.h"

#include "ELGamal.h"

#include "SHA256.h"

#include<NTL/ZZ.h>

#include<iostream>

#include<string>

using namespace std;

NTL\_CLIENT

void displayMenu(){

cout << "=== 保密通信系统 ===" << endl;

cout << "1. 发送消息" << endl;

cout << "2. 退出" << endl;

cout << "===================" << endl;

cout << "请选择操作：";

}

void sendMessage() {

string studentID, name, className;

cout << "请输入12位学号：";

cin >> studentID;

cout << "请输入姓名：";

cin >> name;

cout << "请输入班级：";

cin >> className;

//AES加密

byte plain[16] = { byte(studentID[0]),byte(studentID[1]),byte(studentID[2]),byte(studentID[3]),

byte(studentID[4]),byte(studentID[5]),byte(studentID[6]),byte(studentID[7]),

byte(studentID[8]),byte(studentID[9]),byte(studentID[10]),byte(studentID[11]),

byte(name[0]),byte(name[1]),byte(name[2]),byte(className[0])

};

AES aes;

word w[4 \* (Nr + 1)]; //w数组用于存储轮密钥字（128bit密钥版共使用11个轮密钥，每个轮密钥128bit=4\*32bit，因此一共44个轮密钥字

aes.KeyExpansion(key, w); //密钥扩展函数，实现从种子密钥key到11个轮密钥w的扩展

// 输出待加密的明文

cout << endl << "待加密的明文：" << endl;

for (int i = 0; i < 16; ++i)

{

cout << hex <<plain[i].to\_ulong() << " ";

if ((i + 1) % 4 == 0)

cout << endl;

}

cout << endl;

// aes加密，输出密文

aes.encrypt(plain, w); //明文数组plain和轮密钥数组w输入加密函数，加密得到密文（仍存放在plain数组中）

cout << "aes加密后的密文：" << endl;

for (int i = 0; i < 16; ++i)

{

cout << hex << plain[i].to\_ulong() << " ";

if ((i + 1) % 4 == 0)

cout << endl;

}

cout << endl;

cout << "===================" << endl;

int choice;

cout << "请选择你想用的公钥加密算法：1：RSA,2:ELGamal,3:SM2：";

cin >> choice;

ZZ m;

//初始化RSA

ZZ c[16];

RSA rsa;

//初始化ELGamal对象和储存

ELGamal elgamal;

ZZ \_c1[16], \_c2[16];

if (choice == 1) {

cout << "===================" << endl;

cout << "RSA加密中，请等待" << endl;

//RSA加密w

//证书获取公钥

CA ca;

ca.CAIN(rsa.getPublickey(), rsa.getPrivatekey());

ca.CAOUT(rsa);

for (int i = 0; i < 16; i++) {

m = plain[i].to\_ulong();

c[i] = rsa.RSA\_ENCODE(m);

}

}

else if (choice == 2) {

cout << "===================" << endl;

cout << "ELGamal加密中，请等待" << endl;

for (int i = 0; i < 16; i++) {

m = plain[i].to\_ulong();

elgamal.ElGamal\_Encode(m);

\_c1[i] = elgamal.c1;

\_c2[i] = elgamal.c2;

}

}

//ELGamal签名

//传入message作为签名

cout << "===================" << endl;

cout << "ELGamal签名中，请等待" << endl;

cout << "===================" << endl;

cout << "SHA256加密传入" << endl;

word message[2] = { 0x428a2f98, 0x71374491};

SHA256 sha;

sha.SHA\_Encode(message,2);

cout << "开始签名" << endl;

ZZ el=elgamal.ElGamal\_QM();

cout << "发送签名" << endl;

//验证签名

elgamal.ElGamal\_YZ(el);

cout << "消息已发送！" << endl;

// 在这里添加你的消息解密和发送逻辑

if (choice == 1) {

//rsa解密

for (int i = 0; i < 16; i++) {

m = rsa.RSA\_DECODE(c[i]);

int \_m = conv<int>(m);

plain[i] = byte(\_m);

}

}

else if (choice == 2) {

for (int i = 0; i < 16; i++) {

m = elgamal.ElGamal\_Decode(\_c1[i],\_c2[i]);

int \_m = conv<int>(m);

plain[i] = byte(\_m);

}

}

//aes解密

aes.decrypt(plain, w);

cout << "解密后的明文：" << endl;

for (int i = 0; i < 12; ++i)

{

studentID[i] = char(plain[i].to\_ulong());

}

for (int i = 12; i < 15; ++i)

{

name[i-12] = char(plain[i].to\_ulong());

}

className[0]= char(plain[15].to\_ulong());

cout << endl;

cout <<"接受到的学号："<< studentID << endl;

cout << "接受到的学生姓名：" << name << endl;

cout << "班级：" << className << endl;

}

int main() {

int choice;

do {

displayMenu();

cin >> choice;

switch (choice) {

case 1:

sendMessage();

break;

case 2:

cout << "系统已退出。" << endl;

break;

default:

cout << "无效的选择，请重新输入。" << endl;

break;

}

} while (choice != 2);

return 0;

}

## SOCKET传输JAVA代码

### 服务器代码

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

import java.security.KeyFactory;

import java.security.KeyPair;

import java.security.PublicKey;

import java.security.spec.X509EncodedKeySpec;

import java.util.Arrays;

import java.util.Base64;

public class Server extends JFrame implements ActionListener {

RSAExample rsa;

PublicKey p;

private String F="D:\\rsa.txt";

private String FF="D:\\rsa2.txt";

public static boolean f=false;

Socket socket;

//定义按钮

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;

}

public Server(){

//初始化rsa密钥

try {

rsa = new RSAExample();

}

catch (Exception e) {

throw new RuntimeException(e);

}

//将公钥加密上传到F文件

try {

rsa.postPublicket(F);

}

catch (Exception e) {

throw new RuntimeException(e);

}

//端口初始化

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){

}

//以下是界面设计\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

this.setTitle("服务器");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

area.setEditable(false);

area.setLineWrap(true);

// area.setBackground(Color.red);

this.add(area, BorderLayout.CENTER);

this.add(field,BorderLayout.NORTH);

field.addActionListener(this);

this.add(Sender,BorderLayout.SOUTH); //设置窗口布局，发送按钮在下

Sender2.setSize(80,30);

this.add(Sender2,BorderLayout.AFTER\_LINE\_ENDS);

Sender2.addActionListener((e -> {

try {

post();

}

catch (Exception ex) {

throw new RuntimeException(ex);

}

}));

Sender.addActionListener(this);

this.setSize(500, 700);

this.setLocationRelativeTo(null);

this.setVisible(true);

//-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// thread2 thread2=new thread2();

// while (true) {

// thread2.run();

// if(Server.f) {

// break;

// }

// }

//线程读取信息

MyThread t=new MyThread();

while (true) {

t.run();

}

}

public static void main(String[] args) throws Exception {

Server Se=new Server();

}

//公钥

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));

}

//重写actionPerformed方法发送信息

@Override

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);

}

}

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;

}

//重写run方法接受数据并显示

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) {

}

//

}

}

class thread2 extends Thread {

@Override

public void run() {

// System.out.println("");

if(Server.f==true) {

try {

rsa.getPublicKey(FF);

}

catch (Exception e) {

throw new RuntimeException(e);

}

}

}

}

}

### 客户端代码

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.net.InetAddress;

import java.net.Socket;

import java.security.KeyFactory;

import java.security.PublicKey;

import java.security.spec.X509EncodedKeySpec;

import java.util.Base64;

public class Client extends JFrame implements ActionListener {

RSAExample rsa1;

private String F="D:\\rsa.txt";

private String FF="D:\\rsa2.txt";

//

// {

// try {

// rsa1 =Server.rsa;

// }

// catch (Exception e) {

// throw new RuntimeException(e);

// }

// }

// PublicKey p1;

Socket socket;

//按钮

private JButton Sender = new JButton("发送");

private JButton Sender2 = new JButton("发送公钥");

//显示信息区域

private JTextArea area = new JTextArea("聊天内容：" + '\n');

//文本输入框

private JTextField field = new JTextField("");

//定义返回文本函数

public JTextField getField() {

return field;

}

public Client() {

//初始化rsa密钥

{

try {

rsa1 = new RSAExample();

}

catch (Exception e) {

throw new RuntimeException(e);

}

}

try {

rsa1.postPublicket(FF);

rsa1.getPublicKey(F);

}

catch (Exception e) {

throw new RuntimeException(e);

}

InetAddress ip = null;

try {

// ip = InetAddress.getByName("192.168.201.131");

ip = InetAddress.getByName("Localhost");

int Port = 9999;

this.socket = new Socket(ip, Port);

} catch (Exception e) {

throw new RuntimeException(e);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*

this.setTitle("客户端");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

area.setEditable(false);

area.setLineWrap(true);

// area.setBackground(Color.red);

this.add(area, BorderLayout.CENTER);

this.add(field, BorderLayout.NORTH);

field.addActionListener(this);

this.add(Sender, BorderLayout.SOUTH); //设置窗口布局，发送按钮在下

field.addActionListener(this);

this.add(Sender,BorderLayout.SOUTH); //设置窗口布局，发送按钮在下

Sender2.setSize(80,30);

this.add(Sender2,BorderLayout.AFTER\_LINE\_ENDS);

Sender2.addActionListener((e -> {

try {

post();

}

catch (Exception ex) {

throw new RuntimeException(ex);

}

}));

Sender.addActionListener(this);

this.setSize(500, 700);

// this.setLocationRelativeTo(null);

this.setVisible(true);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//创造并启动线程

MyThread t=new MyThread();

while (true) {

t.run();

}

}

//发送公钥

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(rsa1.postPublicket(FF));

}

public static void main(String[] args) throws Exception {

Login login=new Login();

login.showUI();

thread t=new thread();

while (true) {

t.run();

}

}

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 p2=keyFactory.generatePublic(spec);

rsa1.publicKey=p2;

}

@Override

public void actionPerformed(ActionEvent e) {

OutputStream os = null;

try {

os = socket.getOutputStream();

} catch (IOException r) {

throw new RuntimeException(r);

}

PrintStream ps = new PrintStream(os);

String s2 = this.getField().getText();

if(s2.isEmpty()){

System.out.println("请输入有效信息");

}

else {

try {

byte[] ss=rsa1.Encode(s2,rsa1.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.printl1n(s2);

// System.out.println(s2);

}

}

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 {

// String str = br.readLine();

// System.out.println(str);

// area.append("服务器说:"+str+'\n');

String str = br.readLine();

System.out.println(str);

byte s[]= Base64.getDecoder().decode(str);

// System.out.println(Arrays.toString(s));

// System.out.println(s.toString());

// System.out.println(new String(s,"UTF-8"));

// System.out.println(rsa1.publicKey);

String c=rsa1.Decode(s);

System.out.println("解密："+c);

area.append("服务器说:"+c+'\n');

} catch (Exception e) {

}

//

}

}

}

class thread extends Thread {

@Override

public void run() {

System.out.println("");

if(Server.f==true) {

Client Cl = new Client();

}

}

}

### 登录界面代码

import javax.swing.\*;

public class Login {

// public static void main(String[] args){

//

// Login lo=new Login();

// lo.showUI();

//

// }

public void showUI(){

javax.swing.JFrame jf= new javax.swing.JFrame();

jf.setTitle("登录界面");

jf.setSize(300,400);

jf.setDefaultCloseOperation(3);

jf.setLocationRelativeTo(null);

java.awt.FlowLayout flow=new java.awt.FlowLayout();

jf.setLayout(flow);

javax.swing.ImageIcon icon = new javax.swing.ImageIcon("UI.jpg");

javax.swing.JLabel jla= new javax.swing.JLabel(icon);

java.awt.Dimension dm0=new java.awt.Dimension(280,200);

jla.setPreferredSize(dm0);

jf.add(jla);

javax.swing.JTextField jtf=new javax.swing.JTextField();

java.awt.Dimension dm1=new java.awt.Dimension(280,30);

jtf.setPreferredSize(dm1);

jf.add(jtf);

javax.swing.JTextField jtf2=new javax.swing.JTextField();

java.awt.Dimension dm2=new java.awt.Dimension(280,30);

jtf2.setPreferredSize(dm2);

jf.add(jtf2);

javax.swing.JCheckBox jcb3 = new javax.swing.JCheckBox("忘记密码");

jf.add(jcb3);

javax.swing.JCheckBox jcb4 = new javax.swing.JCheckBox("记住密码");

jf.add(jcb4);

javax.swing.JButton jbu = new javax.swing.JButton("登陆");

java.awt.Dimension dm3=new java.awt.Dimension(80,30);

jbu.setPreferredSize(dm3);

jf.add(jbu);

//给按钮添加动作监听器方法

//创建一个监听器

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);

}

}));

//把监听器加在“登录”按钮上

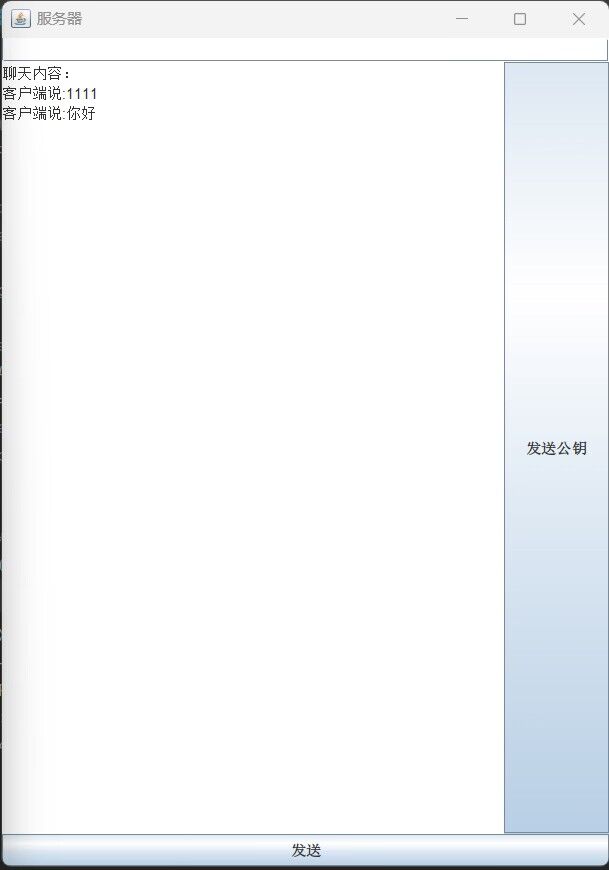
jf.setVisible(true);

}

}

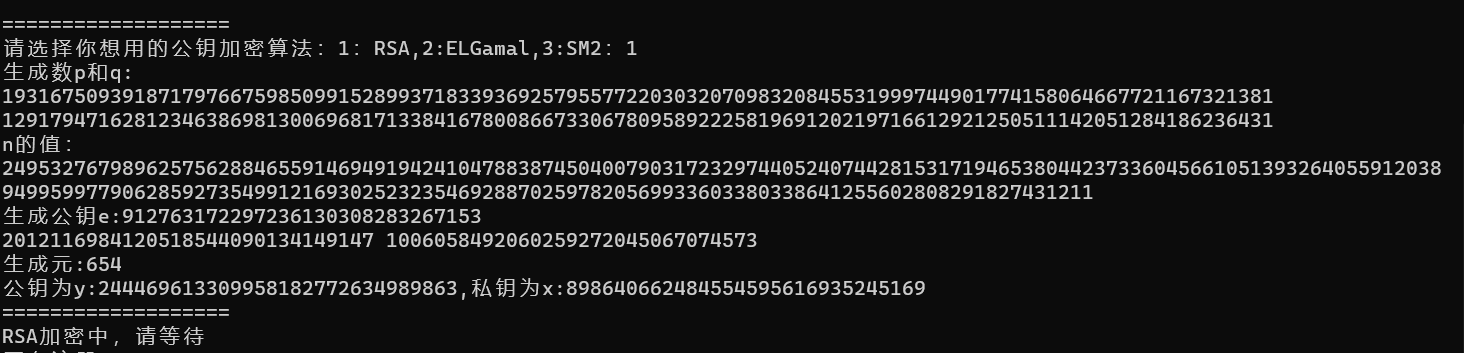
## 实验截图

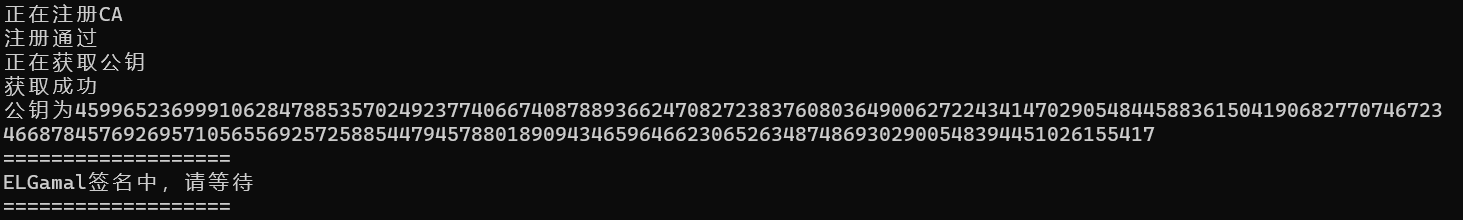


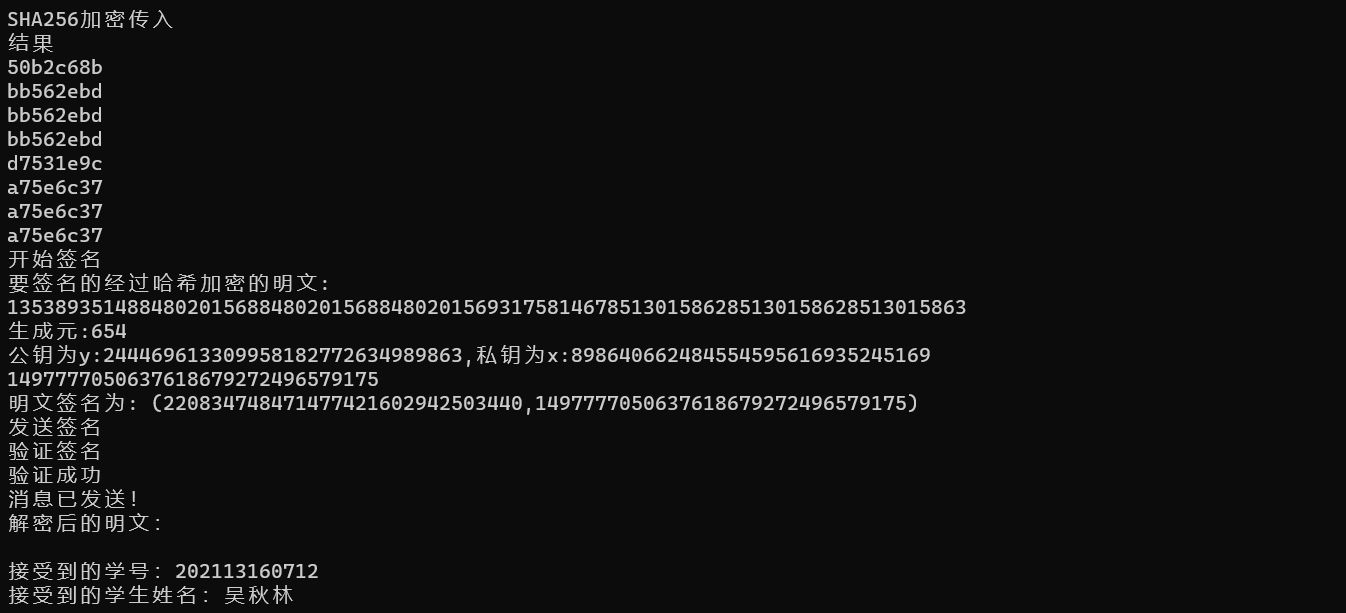












# 5 总结

# 6 参考文献

[1] 甘勇.《Ｃ++语言程序设计》. 河南科学技术出版社. 2016年