# S-Aes开发手册

## 项目背景

S-AES（Simplified Advanced Encryption Standard）是一种简化的高级加密标准（AES）算法，是AES的简化版本，旨在提供基本的数据加密和解密功能，适用于教育、学术和轻量级安全需求。S-AES是AES的教学和演示目的的简化版本，通常用于介绍AES的基本概念和操作步骤。

S-AES的加密步骤如下：

1. 将要加密的内容复制到状态数组中。
2. 初始化密钥。
3. 执行4轮函数，每轮函数包括以下步骤：
   1. 字节代换：将状态数组中的每个字节替换为一个预定义的值。
   2. 行移位：将状态数组中的每一行循环左移不同的位数。
   3. 列混淆：将状态数组中的每一列进行混淆。
   4. 密钥加：将当前轮次的密钥与状态数组进行异或操作。
4. 执行最后一轮函数，不包括列混淆步骤。
5. 将状态数组中的内容输出为密文。

需要注意的是，S-AES的密钥长度只有16位，因此其安全性相对较低，不适用于对安全性要求较高的场合。

以下是S-AES的关键特点和操作步骤：

1. 密钥长度： S-AES使用32位密钥，这与标准AES相比较为简单。密钥长度对于加密算法的安全性至关重要。
2. 初始轮操作： S-AES与标准AES不同，它仅执行一个加密轮（SubBytes、ShiftRows、MixColumns和AddRoundKey），没有多轮迭代。这是为了简化算法。
3. 四个基本操作： S-AES使用四个基本操作来执行加密和解密：

·SubBytes： 将字节替换为S-盒中的值，S-盒是固定的。

·ShiftRows： 对状态矩阵中的行进行循环移位操作。

·MixColumns： 使用固定的矩阵乘法将状态矩阵中的列混合。

·AddRoundKey： 将轮密钥与状态矩阵进行按位异或运算。

1. 轮密钥生成： 为了执行AddRoundKey操作，需要生成轮密钥。这些轮密钥是从初始密钥派生的，并在每个轮次中使用。
2. 加密过程： 加密过程包括初始轮操作，然后执行四个基本操作，最后一个轮操作。
3. 解密过程： 解密过程与加密过程相反，需要执行逆操作来还原原始数据。本开发手册包含以下内容：

通过阅读此开发手册，您可以初步了解项目的开发环境、主要结构和作用函数等。

## S-Aes开发环境

IntelliJ IDEA, VUE3, Mac

## 项目的主要结构

为在提高项目的开发效率、可维护性和可扩展性的同时给用户提供更好的使用体验，项目采用了前后端分离部署的开发方式。下面将分别从前端和后端的开发两方面进行介绍。

* **前端**

Vue是一个轻量级的JavaScript框架，易于学习和使用，可以快速构建高效的单页应用程序；具有响应式的数据绑定和组件化的框架，可以提高开发效率和代码的可维护性；具有丰富的插件和工具生态系统，可以轻松地扩展和定制应用程序；支持前后端分离，可以与各种后端技术（如Java、PHP、Node.js等）进行集成，实现更灵活的应用程序开发和部署；可以通过打包工具（如Webpack）将应用程序打包成静态文件，可以方便地部署到各种Web服务器上，如Nginx、Apache等。

鉴于使用Vue进行前端开发具有以上优点，故我们选用Vue3进行前端开发的实现。

下为前端的文件结构：

├── App.vue

├── assets

│   ├── base.css

│   ├── decrypt.png

│   ├── encrtptText.png

│   ├── encrypt.png

│   ├── logo.svg

│   └── main.css

├── components

│   ├── Decrypt.vue

│   ├── Encrypt.vue

│   ├── Header.vue

│   ├── Introduction.vue

│   ├── MidAttack.vue

│   ├── PageHead.vue

│   └── ShowPage.vue

└── main.js

其中decrypt.png为解码页面的运行示例，encryptText.png为中间相遇攻击页面的运行示例，encrypt.png为加密页面的运行示例。

后缀为.css的文件则用于保证格式的重用以及网页的修改与维护、实现对网页的精确控制（包括但不限于网页的布局、字体、颜色、背景）以及实现多个网页的同时更新。

后缀为.vue的文件则是用于Vue.js框架的单文件组件格式，其中包含了每个组件的相关代码，它们的数据互不影响。其中Encrypt.vue文件为加密页面的相关代码，Decrypt.vue为解密页面的相关代码，MidAttack.vue为中间相遇攻击的相关代码，其余则均为首页的组件的相关代码。

* **后端**

Maven是一个Java项目管理工具，主要用于管理Java项目的构建、依赖关系和文档等。它可以自动下载所需的依赖项，并将它们添加到项目中，从而简化了项目的构建和管理过程。Maven还提供了一些标准化的目录结构和构建生命周期，使得项目的构建和部署更加规范化和可靠。此外，Maven还可以生成项目文档和报告，方便项目管理和团队协作。Maven可以自动执行项目的构建过程，包括编译、测试、打包、部署等，从而减少了手动操作的工作量。

基于Maven的便利性及可靠性，我们选用Maven进行后端的开发。

后端的结构如下：

├── main

│   ├── java

│   │   ├── edu

│   │   │   └── security

│   │   │   └── saesbackend

│   │   │   ├── SAesBackEndApplication.java

│   │   │   ├── config

│   │   │   │   └── config.java

│   │   │   ├── controller

│   │   │   │   └── SAESController.java

│   │   │   └── pojo

│   │   │   ├── Decryption.java

│   │   │   ├── Encryption.java

│   │   │   ├── RequestData.java

│   │   │   ├── ResponseData.java

│   │   │   ├── Result.java

│   │   │   └── SAES.java

│   │   └── service

│   │   └── AesService.java

│   └── resources

│   ├── META-INF

│   │   └── MANIFEST.MF

│   ├── application.yml

│   ├── static

│   └── templates

└── test

└── java

└── edu

└── security

└── saesbackend

└── SAesBackEndApplicationTests.java

**下面将对项目的主要部分（加密、解密和后端交互函数）进行简单显示。**

## Dcryption的主要函数

public void setKey0(String key0){  
 this.key0=key0;  
 }  
 public void setKey1(String key1){  
 this.key1=key1;  
 }  
 public static int[] RotNib(int[] inputArray) {  
 if (inputArray.length != 8) {  
 throw new IllegalArgumentException("Input array must be of length 8");  
 }  
 int[] rotatedArray = new int[8];  
  
 *// Swap the first 4 elements with the last 4 elements* for (int i = 0; i < 8; i++) {  
 if (i < 4) {  
 rotatedArray[i] = inputArray[i + 4];  
 } else {  
 rotatedArray[i] = inputArray[i - 4];  
 }  
 }  
 return rotatedArray;  
 }  
 public int[] getKey0Array(){  
 return this.key0Array;  
 }  
 public int[] getKey4Array(){  
 return this.key4Array;  
 }  
 public int[] getKey1Array() {  
 return this.key1Array;  
 }  
 public int[] getKey2Array() {  
 return this.key2Array;  
 }  
 public int[] getKey3Array(){  
 return this.key3Array;  
 }  
 public int[] getKey5Array(){  
 return this.key5Array;  
 }  
 public static int[] XOR(int[] array1, int[] array2) {  
 if (array1.length != array2.length) {  
 throw new IllegalArgumentException("Input arrays must be of the same length");  
 }  
 int[] result = new int[array1.length];  
  
 for (int i = 0; i < array1.length; i++) {  
 result[i] = array1[i] ^ array2[i];  
 }  
 return result;  
 }  
 *// Helper function to convert a binary array to a decimal value* public static int binaryToDecimal(int[] binaryArray) {  
 int decimalValue = 0;  
 for (int i = 0; i < binaryArray.length; i++) {  
 decimalValue = decimalValue \* 2 + binaryArray[i];  
 }  
 return decimalValue;  
 }  
  
 *// Helper function to convert a decimal value to a binary array* public static int[] decimalToBinary(int decimalValue) {  
 int[] binaryArray = new int[4];  
 for (int i = 3; i >= 0; i--) {  
 binaryArray[i] = decimalValue % 2;  
 decimalValue /= 2;  
 }  
 return binaryArray;  
 }  
 public static int[] SubNib(int[] inputArray) {  
 if (inputArray.length != 8) {  
 throw new IllegalArgumentException("Input array must be of length 8");  
 }  
  
 *// Divide the input into two 4-bit nibbles* int[] nibble1 = Arrays.*copyOfRange*(inputArray, 0, 4);  
 int[] nibble2 = Arrays.*copyOfRange*(inputArray, 4, 8);  
  
 *// Convert each nibble to decimal values for rows and columns* int row1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1, 0, 2));  
 int col1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1, 2, 4));  
  
 int row2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2, 0, 2));  
 int col2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2, 2, 4));  
  
 *// Look up the values in the S-Box* int substitutedValue1 = *sBoxKey*[row1][col1];  
 int substitutedValue2 = *sBoxKey*[row2][col2];  
  
 *// Convert the substituted values back to binary (4-bit)* int[] substitutedNibble1 = *decimalToBinary*(substitutedValue1);  
 int[] substitutedNibble2 = *decimalToBinary*(substitutedValue2);  
  
 *// Combine the two substituted nibbles into an 8-bit array* int[] substitutedArray = new int[8];  
 System.*arraycopy*(substitutedNibble1, 0, substitutedArray, 0, 4);  
 System.*arraycopy*(substitutedNibble2, 0, substitutedArray, 4, 4);  
 return substitutedArray;  
 }  
 public int[] round1(int[] plainTextArray){  
  
*// Inverse Shift Row (same as normal)* int[] rowShiftedArray= new int[16];  
 for(int i=0;i<16;i++){  
 if(i>=3&&i<=7){  
 rowShiftedArray[i]=plainTextArray[i+8];  
 }else if(i>=11&&i<=15){  
 rowShiftedArray[i]=plainTextArray[i-8];  
 }else{  
 rowShiftedArray[i]=plainTextArray[i];  
 }  
 }  
 *//log.info("shift rows {}",AesService.BinaryToString(rowShiftedArray));  
  
 // Inverse Nibble Sub (use the inverse or decryption S-box)* int[] nibble1\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 0, 4);  
 int[] nibble2\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 4, 8);  
 int[] nibble3\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 8, 12);  
 int[] nibble4\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 12, 16);  
  
 int row1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 0, 2));  
 int col1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 2, 4));  
  
 int row2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 0, 2));  
 int col2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 2, 4));  
  
 int row3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 0, 2));  
 int col3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 2, 4));  
  
 int row4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 0, 2));  
 int col4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 2, 4));  
  
 int substitutedValue1 = *sBox*[row1][col1];  
 int substitutedValue2 = *sBox*[row2][col2];  
 int substitutedValue3 = *sBox*[row3][col3];  
 int substitutedValue4 = *sBox*[row4][col4];  
  
 *// Convert the substituted values back to binary (4-bit)* int[] substitutedNibble1 = *decimalToBinary*(substitutedValue1);  
 int[] substitutedNibble2 = *decimalToBinary*(substitutedValue2);  
 int[] substitutedNibble3 = *decimalToBinary*(substitutedValue3);  
 int[] substitutedNibble4 = *decimalToBinary*(substitutedValue4);  
  
  
 int[] substitutedArray = new int[16];  
 System.*arraycopy*(substitutedNibble1, 0, substitutedArray, 0, 4);  
 System.*arraycopy*(substitutedNibble2, 0, substitutedArray, 4, 4);  
 System.*arraycopy*(substitutedNibble3, 0, substitutedArray, 8, 4);  
 System.*arraycopy*(substitutedNibble4, 0, substitutedArray, 12, 4);  
 *//log.info("inverse Sbox {}", AesService.BinaryToString(substitutedArray));  
 // Add Round 1 Key* int[] plusArray=*XOR*(substitutedArray,subKey1);  
  
 *//log.info("add round 1 key {}",AesService.BinaryToString(plusArray));* int[] nibble1 = Arrays.*copyOfRange*(plusArray, 0, 4);  
 int[] nibble2 = Arrays.*copyOfRange*(plusArray, 4, 8);  
 int[] nibble3 = Arrays.*copyOfRange*(plusArray, 8, 12);  
 int[] nibble4 = Arrays.*copyOfRange*(plusArray, 12, 16);  
 *//log.info("Nibble1 is {}",nibble1);  
 //log.info("Nibble2 is {}",nibble2);  
 //log.info("Nibble3 is {}",nibble3);  
 //log.info("Nibble4 is {}",nibble4);* int[] newNibble1=*XOR*(*decimalToBinary*(*GFProduct*[9][*binaryToDecimal*(nibble1)]),*decimalToBinary*(*GFProduct*[2][*binaryToDecimal*(nibble2)]));  
 int[] newNibble2=*XOR*(*decimalToBinary*(*GFProduct*[9][*binaryToDecimal*(nibble2)]),*decimalToBinary*(*GFProduct*[2][*binaryToDecimal*(nibble1)]));  
 int[] newNibble3=*XOR*(*decimalToBinary*(*GFProduct*[2][*binaryToDecimal*(nibble4)]),*decimalToBinary*(*GFProduct*[9][*binaryToDecimal*(nibble3)]));  
 int[] newNibble4=*XOR*(*decimalToBinary*(*GFProduct*[2][*binaryToDecimal*(nibble3)]),*decimalToBinary*(*GFProduct*[9][*binaryToDecimal*(nibble4)]));  
 *//log.info("new Nibble00 is {}",newNibble1);  
 //log.info("new Nibble01 is {}",newNibble2);  
 //log.info("new Nibble10 is {}",newNibble3);  
 //log.info("new Nibble11 is {}",newNibble4);* int[] mixColArray =new int[16];  
 System.*arraycopy*(newNibble1, 0, mixColArray, 0, 4);  
 System.*arraycopy*(newNibble2, 0, mixColArray, 4, 4);  
 System.*arraycopy*(newNibble3, 0, mixColArray, 8, 4);  
 System.*arraycopy*(newNibble4, 0, mixColArray, 12, 4);  
 *//log.info("Mixcol {}",AesService.BinaryToString(mixColArray));* return round2(mixColArray);  
 }  
 public int[] round2(int[] plainTextArray){  
  
 int[] rowShiftedArray= new int[16];  
 for(int i=0;i<16;i++){  
 if(i>=3&&i<=7){  
 rowShiftedArray[i]=plainTextArray[i+8];  
 }else if(i>=11&&i<=15){  
 rowShiftedArray[i]=plainTextArray[i-8];  
 }else{  
 rowShiftedArray[i]=plainTextArray[i];  
 }  
 }  
 *//log.info("shitf rows {}",AesService.BinaryToString(rowShiftedArray));  
  
  
 // Inverse Nibble Sub (use the inverse or decryption S-box)* int[] nibble1\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 0, 4);  
 int[] nibble2\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 4, 8);  
 int[] nibble3\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 8, 12);  
 int[] nibble4\_round1 = Arrays.*copyOfRange*(rowShiftedArray, 12, 16);  
  
 int row1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 0, 2));  
 int col1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 2, 4));  
  
 int row2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 0, 2));  
 int col2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 2, 4));  
  
 int row3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 0, 2));  
 int col3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 2, 4));  
  
 int row4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 0, 2));  
 int col4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 2, 4));  
  
 int substitutedValue1 = *sBox*[row1][col1];  
 int substitutedValue2 = *sBox*[row2][col2];  
 int substitutedValue3 = *sBox*[row3][col3];  
 int substitutedValue4 = *sBox*[row4][col4];  
  
 *// Convert the substituted values back to binary (4-bit)* int[] substitutedNibble1 = *decimalToBinary*(substitutedValue1);  
 int[] substitutedNibble2 = *decimalToBinary*(substitutedValue2);  
 int[] substitutedNibble3 = *decimalToBinary*(substitutedValue3);  
 int[] substitutedNibble4 = *decimalToBinary*(substitutedValue4);  
  
 int[] substitutedArray = new int[16];  
 System.*arraycopy*(substitutedNibble1, 0, substitutedArray, 0, 4);  
 System.*arraycopy*(substitutedNibble2, 0, substitutedArray, 4, 4);  
 System.*arraycopy*(substitutedNibble3, 0, substitutedArray, 8, 4);  
 System.*arraycopy*(substitutedNibble4, 0, substitutedArray, 12, 4);  
  
  
 *//log.info("inverse sbox {}",AesService.BinaryToString(substitutedArray));* int[] plusArray=*XOR*(substitutedArray,subKey0);  
 *//log.info("add round 0 key {}",AesService.BinaryToString(plusArray));* return plusArray;  
 }  
  
 public int[] decrypt(Object plainText) {  
 int[] plainTextArray=new int[16];  
 if (plainText instanceof Character) {  
 plainTextArray=*decryptChar*((char) plainText);  
 } else if (plainText instanceof String) {  
 plainTextArray=*decryptString*((String) plainText);  
 }  
  
 this.key0Array=new int[8];  
 this.key1Array=new int[8];  
 this.key2Array=new int[8];  
 this.key3Array=new int[8];  
 this.key4Array=new int[8];  
 this.key5Array=new int[8];  
 for (int i = 0; i < 8; i++) {  
 key0Array[i] = Integer.*parseInt*(this.key0.substring(i, i + 1));  
 key1Array[i] = Integer.*parseInt*(this.key1.substring(i, i + 1));  
 }  
  
 key2Array=*XOR*(key0Array,*XOR*(new int[]{1,0,0,0,0,0,0,0},*SubNib*(*RotNib*(key1Array))));  
 key3Array=*XOR*(key2Array,key1Array);  
 key4Array=*XOR*(key2Array,*XOR*(new int[]{0,0,1,1,0,0,0,0},*SubNib*(*RotNib*(key3Array))));  
 key5Array=*XOR*(key4Array,key3Array);  
 System.*arraycopy*(key0Array, 0, subKey0, 0, 8);  
 System.*arraycopy*(key1Array, 0, subKey0, 8, 8);  
 System.*arraycopy*(key2Array, 0, subKey1, 0, 8);  
 System.*arraycopy*(key3Array, 0, subKey1, 8, 8);  
 System.*arraycopy*(key4Array, 0, subKey2, 0, 8);  
 System.*arraycopy*(key5Array, 0, subKey2, 8, 8);  
 *//log.info("key0 is {}",subKey0);  
 //log.info("key1 is {}",subKey1);  
 //log.info("key2 is {}",subKey2);  
 //log.info("ori is {}",AesService.BinaryToString(plainTextArray));  
// Add round 2 key* plainTextArray=*XOR*(plainTextArray,subKey2);  
 *//log.info("add round 2 key {}",AesService.BinaryToString(plainTextArray));  
// round 1* return round1(plainTextArray);  
 }  
  
 public static int[] decryptChar(char plainText) {  
 String binaryText = String.*format*("%16s", Integer.*toBinaryString*(plainText)).replace(' ', '0');  
 int[] binaryArray = new int[16];  
 for (int i = 0; i < 16; i++) {  
 binaryArray[i] = binaryText.charAt(i) - '0';  
 }  
 return binaryArray;  
 }  
  
 public static int[] decryptString(String binaryText) {  
 int[] binaryArray = new int[16];  
 for (int i = 0; i < 16; i++) {  
 binaryArray[i] = binaryText.charAt(i) - '0';  
 }  
 return binaryArray;  
 }  
}

## Encryption的主要函数

static {  
 for (int i = 0; i < 16; i++) {  
 for (int j = 0; j < 16; j++) {  
 *GFPlus*[i][j] = i ^ j;  
 }  
 }  
 }  
 public int[] getKey0Array(){  
 return this.key0Array;  
 }  
 public int[] getKey4Array(){  
 return this.key4Array;  
 }  
 public int[] getKey1Array() {  
 return this.key1Array;  
 }  
 public int[] getKey2Array() {  
 return this.key2Array;  
 }  
 public int[] getKey3Array(){  
 return this.key3Array;  
 }  
 public int[] getKey5Array(){  
 return this.key5Array;  
 }  
 public void setKey0(String key0){  
 this.key0=key0;  
 }  
 public void setKey1(String key1){  
 this.key1=key1;  
 }  
 public static int[] RotNib(int[] inputArray) {  
 if (inputArray.length != 8) {  
 throw new IllegalArgumentException("Input array must be of length 8");  
 }  
 int[] rotatedArray = new int[8];  
  
 *// Swap the first 4 elements with the last 4 elements* for (int i = 0; i < 8; i++) {  
 if (i < 4) {  
 rotatedArray[i] = inputArray[i + 4];  
 } else {  
 rotatedArray[i] = inputArray[i - 4];  
 }  
 }  
 return rotatedArray;  
 }  
 public static int[] XOR(int[] array1, int[] array2) {  
 if (array1.length != array2.length) {  
 throw new IllegalArgumentException("Input arrays must be of the same length");  
 }  
 int[] result = new int[array1.length];  
  
 for (int i = 0; i < array1.length; i++) {  
 result[i] = array1[i] ^ array2[i];  
 }  
 return result;  
 }  
 *// Helper function to convert a binary array to a decimal value* public static int binaryToDecimal(int[] binaryArray) {  
 int decimalValue = 0;  
 for (int i = 0; i < binaryArray.length; i++) {  
 decimalValue = decimalValue \* 2 + binaryArray[i];  
 }  
 return decimalValue;  
 }  
  
 *// Helper function to convert a decimal value to a binary array* public static int[] decimalToBinary(int decimalValue) {  
 int[] binaryArray = new int[4];  
 for (int i = 3; i >= 0; i--) {  
 binaryArray[i] = decimalValue % 2;  
 decimalValue /= 2;  
 }  
 return binaryArray;  
 }  
 public static int[] SubNib(int[] inputArray) {  
 if (inputArray.length != 8) {  
 throw new IllegalArgumentException("Input array must be of length 8");  
 }  
  
 *// Divide the input into two 4-bit nibbles* int[] nibble1 = Arrays.*copyOfRange*(inputArray, 0, 4);  
 int[] nibble2 = Arrays.*copyOfRange*(inputArray, 4, 8);  
  
 *// Convert each nibble to decimal values for rows and columns* int row1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1, 0, 2));  
 int col1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1, 2, 4));  
  
 int row2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2, 0, 2));  
 int col2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2, 2, 4));  
  
 *// Look up the values in the S-Box* int substitutedValue1 = *sBox*[row1][col1];  
 int substitutedValue2 = *sBox*[row2][col2];  
  
 *// Convert the substituted values back to binary (4-bit)* int[] substitutedNibble1 = *decimalToBinary*(substitutedValue1);  
 int[] substitutedNibble2 = *decimalToBinary*(substitutedValue2);  
  
 *// Combine the two substituted nibbles into an 8-bit array* int[] substitutedArray = new int[8];  
 System.*arraycopy*(substitutedNibble1, 0, substitutedArray, 0, 4);  
 System.*arraycopy*(substitutedNibble2, 0, substitutedArray, 4, 4);  
 return substitutedArray;  
 }  
 public int[] round1(int[] plainTextArray){  
 *// Round 1* int[] nibble1\_round1 = Arrays.*copyOfRange*(plainTextArray, 0, 4);  
 int[] nibble2\_round1 = Arrays.*copyOfRange*(plainTextArray, 4, 8);  
 int[] nibble3\_round1 = Arrays.*copyOfRange*(plainTextArray, 8, 12);  
 int[] nibble4\_round1 = Arrays.*copyOfRange*(plainTextArray, 12, 16);  
  
 int row1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 0, 2));  
 int col1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 2, 4));  
  
 int row2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 0, 2));  
 int col2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 2, 4));  
  
 int row3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 0, 2));  
 int col3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 2, 4));  
  
 int row4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 0, 2));  
 int col4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 2, 4));  
  
 int substitutedValue1 = *sBox*[row1][col1];  
 int substitutedValue2 = *sBox*[row2][col2];  
 int substitutedValue3 = *sBox*[row3][col3];  
 int substitutedValue4 = *sBox*[row4][col4];  
  
 *// Convert the substituted values back to binary (4-bit)* int[] substitutedNibble1 = *decimalToBinary*(substitutedValue1);  
 int[] substitutedNibble2 = *decimalToBinary*(substitutedValue2);  
 int[] substitutedNibble3 = *decimalToBinary*(substitutedValue3);  
 int[] substitutedNibble4 = *decimalToBinary*(substitutedValue4);  
  
 int[] substitutedArray = new int[16];  
 System.*arraycopy*(substitutedNibble1, 0, substitutedArray, 0, 4);  
 System.*arraycopy*(substitutedNibble2, 0, substitutedArray, 4, 4);  
 System.*arraycopy*(substitutedNibble3, 0, substitutedArray, 8, 4);  
 System.*arraycopy*(substitutedNibble4, 0, substitutedArray, 12, 4);  
 *//log.info("SBOX{}",AesService.BinaryToString(substitutedArray));* int[] rowShiftedArray= new int[16];  
 for(int i=0;i<16;i++){  
 if(i>=3&&i<=7){  
 rowShiftedArray[i]=substitutedArray[i+8];  
 }else if(i>=11&&i<=15){  
 rowShiftedArray[i]=substitutedArray[i-8];  
 }else{  
 rowShiftedArray[i]=substitutedArray[i];  
 }  
 }  
 *//log.info("Shift ROWS{}",AesService.BinaryToString(rowShiftedArray));* int[] nibble1 = Arrays.*copyOfRange*(rowShiftedArray, 0, 4);  
 int[] nibble2 = Arrays.*copyOfRange*(rowShiftedArray, 4, 8);  
 int[] nibble3 = Arrays.*copyOfRange*(rowShiftedArray, 8, 12);  
 int[] nibble4 = Arrays.*copyOfRange*(rowShiftedArray, 12, 16);  
 *//log.info("Nibble 1 is {}",nibble1);  
 //log.info("Nibble 2 is {}",nibble2);  
 //log.info("Nibble 3 is {}",nibble3);  
 //log.info("Nibble 4 is {}",nibble4);* int[] newNibble1=*XOR*(nibble1,*decimalToBinary*(*GFProduct*[4][*binaryToDecimal*(nibble2)]));  
 int[] newNibble2=*XOR*(nibble2,*decimalToBinary*(*GFProduct*[4][*binaryToDecimal*(nibble1)]));  
 int[] newNibble3=*XOR*(*decimalToBinary*(*GFProduct*[4][*binaryToDecimal*(nibble4)]), nibble3);  
 int[] newNibble4=*XOR*(*decimalToBinary*(*GFProduct*[4][*binaryToDecimal*(nibble3)]),nibble4);  
 *//log.info("new Nibble1 is {}",newNibble1);  
 //log.info("new Nibble2 is {}",newNibble2);  
 //log.info("new Nibble3 is {}",newNibble3);  
 //log.info("new Nibble4 is {}",newNibble4);* int[] mixColArray =new int[16];  
 System.*arraycopy*(newNibble1, 0, mixColArray, 0, 4);  
 System.*arraycopy*(newNibble2, 0, mixColArray, 4, 4);  
 System.*arraycopy*(newNibble3, 0, mixColArray, 8, 4);  
 System.*arraycopy*(newNibble4, 0, mixColArray, 12, 4);  
 *//log.info("mixCol{} ",AesService.BinaryToString(mixColArray));* int[] plusArray=*XOR*(mixColArray,subKey1);  
 *//log.info("add round 1 key {}",AesService.BinaryToString(plusArray));* return round2(plusArray);  
 }  
 public int[] round2(int[] plainTextArray){  
 int[] nibble1\_round1 = Arrays.*copyOfRange*(plainTextArray, 0, 4);  
 int[] nibble2\_round1 = Arrays.*copyOfRange*(plainTextArray, 4, 8);  
 int[] nibble3\_round1 = Arrays.*copyOfRange*(plainTextArray, 8, 12);  
 int[] nibble4\_round1 = Arrays.*copyOfRange*(plainTextArray, 12, 16);  
  
 int row1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 0, 2));  
 int col1 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble1\_round1, 2, 4));  
  
 int row2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 0, 2));  
 int col2 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble2\_round1, 2, 4));  
  
 int row3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 0, 2));  
 int col3 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble3\_round1, 2, 4));  
  
 int row4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 0, 2));  
 int col4 = *binaryToDecimal*(Arrays.*copyOfRange*(nibble4\_round1, 2, 4));  
  
 int substitutedValue1 = *sBox*[row1][col1];  
 int substitutedValue2 = *sBox*[row2][col2];  
 int substitutedValue3 = *sBox*[row3][col3];  
 int substitutedValue4 = *sBox*[row4][col4];  
  
 *// Convert the substituted values back to binary (4-bit)* int[] substitutedNibble1 = *decimalToBinary*(substitutedValue1);  
 int[] substitutedNibble2 = *decimalToBinary*(substitutedValue2);  
 int[] substitutedNibble3 = *decimalToBinary*(substitutedValue3);  
 int[] substitutedNibble4 = *decimalToBinary*(substitutedValue4);  
  
 int[] substitutedArray = new int[16];  
 System.*arraycopy*(substitutedNibble1, 0, substitutedArray, 0, 4);  
 System.*arraycopy*(substitutedNibble2, 0, substitutedArray, 4, 4);  
 System.*arraycopy*(substitutedNibble3, 0, substitutedArray, 8, 4);  
 System.*arraycopy*(substitutedNibble4, 0, substitutedArray, 12, 4);  
 *//log.info("sbox {}",AesService.BinaryToString(substitutedArray));* int[] rowShiftedArray= new int[16];  
 for(int i=0;i<16;i++){  
 if(i>=3&&i<=7){  
 rowShiftedArray[i]=substitutedArray[i+8];  
 }else if(i>=11&&i<=15){  
 rowShiftedArray[i]=substitutedArray[i-8];  
 }else{  
 rowShiftedArray[i]=substitutedArray[i];  
 }  
 }  
 *//log.info("shift rows {}",AesService.BinaryToString(rowShiftedArray));* int[] plusArray=*XOR*(subKey2,rowShiftedArray);  
 *//log.info("add round 2 key {}",AesService.BinaryToString(plusArray));* return plusArray;  
 }  
  
 public int[] encrypt(Object plainText) {  
 int[] plainTextArray=new int[16];  
 if (plainText instanceof Character) {  
 plainTextArray=*encryptChar*((char) plainText);  
 } else if (plainText instanceof String) {  
 plainTextArray=*encryptString*((String) plainText);  
 }  
  
 this.key0Array=new int[8];  
 this.key1Array=new int[8];  
 this.key2Array=new int[8];  
 this.key3Array=new int[8];  
 this.key4Array=new int[8];  
 this.key5Array=new int[8];  
 for (int i = 0; i < 8; i++) {  
 key0Array[i] = Integer.*parseInt*(this.key0.substring(i, i + 1));  
 key1Array[i] = Integer.*parseInt*(this.key1.substring(i, i + 1));  
 }  
  
 key2Array=*XOR*(key0Array,*XOR*(new int[]{1,0,0,0,0,0,0,0},*SubNib*(*RotNib*(key1Array))));  
 key3Array=*XOR*(key2Array,key1Array);  
 key4Array=*XOR*(key2Array,*XOR*(new int[]{0,0,1,1,0,0,0,0},*SubNib*(*RotNib*(key3Array))));  
 key5Array=*XOR*(key4Array,key3Array);  
 System.*arraycopy*(key0Array, 0, subKey0, 0, 8);  
 System.*arraycopy*(key1Array, 0, subKey0, 8, 8);  
 System.*arraycopy*(key2Array, 0, subKey1, 0, 8);  
 System.*arraycopy*(key3Array, 0, subKey1, 8, 8);  
 System.*arraycopy*(key4Array, 0, subKey2, 0, 8);  
 System.*arraycopy*(key5Array, 0, subKey2, 8, 8);  
 *//log.info("key0{}",subKey0);  
 //log.info("key1{}",subKey1);  
 //log.info("key2{}",subKey2);  
// Add round 0 key* plainTextArray=*XOR*(plainTextArray,subKey0);  
 *//log.info("ADD round 0 key{}",AesService.BinaryToString(plainTextArray));  
// round 1* round1(plainTextArray);  
 return round1(plainTextArray);  
 }  
  
 public static int[] encryptChar(char plainText) {  
 String binaryText = String.*format*("%16s", Integer.*toBinaryString*(plainText)).replace(' ', '0');  
 int[] binaryArray = new int[16];  
 for (int i = 0; i < 16; i++) {  
 binaryArray[i] = binaryText.charAt(i) - '0';  
 }  
 return binaryArray;  
 }  
  
 public static int[] encryptString(String binaryText) {  
 int[] binaryArray = new int[16];  
 for (int i = 0; i < 16; i++) {  
 binaryArray[i] = binaryText.charAt(i) - '0';  
 }  
 return binaryArray;  
 }  
}

## AesService的主要函数

public void setKey1(String key1) {  
 this.key1 = key1;  
 }  
  
 public void setCypherText(String cypherText) {  
 this.cypherText = cypherText;  
 }  
  
 public void setEncryption(Encryption encryption) {  
 this.encryption = encryption;  
 }  
  
 public void setKey2(String key2) {  
 this.key2 = key2;  
 }  
  
 public void setPlainText(String plainText) {  
 this.plainText = plainText;  
 }  
  
 public int[] DoubleAESEncrypt(String plainText){  
 AesService aesService=new AesService();  
 aesService.setKey1(this.key1);  
 int[] midText=aesService.SAESEncrypt(plainText);  
 aesService.setKey1(this.key2);  
 int[] finalAns=aesService.SAESEncrypt(*BinaryToString*(midText));  
 return finalAns;  
 }  
 public int[] TrippleAESEncrypt(String plainText){  
 AesService aesService= new AesService();  
 aesService.setKey1(this.key1);  
 int[] midText=aesService.SAESEncrypt(plainText);  
 aesService.setKey1(this.key2);  
 midText=aesService.SAESDecrypy(*BinaryToString*(midText));  
 aesService.setKey1(this.key1);  
 int[] finalAns=aesService.SAESEncrypt(*BinaryToString*(midText));  
 return finalAns;  
 }  
 public int[] DoubleAESDecrypt(String cypherText){  
 AesService aesService=new AesService();  
 aesService.setKey1(this.key2);  
 int[] midText=aesService.SAESDecrypy(cypherText);  
 aesService.setKey1(this.key1);  
 int[] finalAns=aesService.SAESDecrypy(*BinaryToString*(midText));  
  
 return finalAns;  
 }  
 public int[] TrippleAESDecrypt(String cypherText){  
 this.setCypherText(cypherText);  
 encryption.setKey0(this.key1.substring(0,8));  
 encryption.setKey1(this.key1.substring(8,16));  
  
 decryption.setKey0(this.key2.substring(0,8));  
 decryption.setKey1(this.key2.substring(8,16));  
  
 int[] midText1=decryption.decrypt(this.cypherText);  
 String midTextString1=*BinaryToString*(midText1);  
  
 int[] midText2=encryption.encrypt(midTextString1);  
 String midTextString2=*BinaryToString*(midText2);  
  
 return encryption.encrypt(midTextString2);  
 }  
 public int[] SAESEncrypt(String plainText){  
 this.setPlainText(plainText);  
 encryption.setKey0(this.key1.substring(0,8));  
 encryption.setKey1(this.key1.substring(8,16));  
 return encryption.encrypt(this.plainText);  
 }  
 public int[] SAESDecrypy(String cypherText){  
 this.setCypherText(cypherText);  
 decryption.setKey0(this.key1.substring(0,8));  
 decryption.setKey1(this.key1.substring(8,16));  
 return decryption.decrypt(this.cypherText);  
 }  
 public static String BinaryToString(int[] binaryArray) {  
 StringBuilder sb = new StringBuilder(binaryArray.length);  
 for (int bit : binaryArray) {  
 sb.append(bit);  
 }  
 return sb.toString();  
 }  
 public static int[] StringtoBinary(String s){  
 int num=0;  
 for(char c:s.toCharArray()){  
 num=num\*2+c-'0';  
 }  
 return *DecimalToBinary*(num);  
 }  
 public static List<String> WordsToKeys(String words) {  
 List<String> binaryList = new ArrayList<>();  
  
 *// Iterate through each character in the input text* for (char c : words.toCharArray()) {  
 *// Convert the character to its binary representation with leading zeros* String binaryString = Integer.*toBinaryString*(c);  
 while (binaryString.length() < 16) {  
 binaryString = "0" + binaryString;  
 }  
 binaryList.add(binaryString);  
 }  
 return binaryList;  
 }  
 public static int[] DecimalToBinary(int decimalValue) {  
 int[] binaryArray = new int[16];  
 for (int i = 15; i >= 0; i--) {  
 binaryArray[i] = decimalValue % 2;  
 decimalValue /= 2;  
 }  
 return binaryArray;  
 }  
 public static Set<String> Attack(String plainText,String cypherText,String plainText2,String cypherText2){  
 int n = (int) Math.*pow*(2, 16); *// 计算2的16次方* List<String>midText=new ArrayList<>();  
 Set<String> possibleKeys=new HashSet<>();  
 for (int i = 0; i < n; i++) {  
 int[] key1 = *DecimalToBinary*(i);  
 AesService aesService = new AesService();  
 aesService.setKey1(*BinaryToString*(key1));  
 int[] tmp=aesService.SAESEncrypt(plainText);  
 midText.add(i,*BinaryToString*(tmp));  
 }  
 for (int i = 0; i < n; i++) {  
 int[] key1 = *DecimalToBinary*(i);  
 AesService aesService = new AesService();  
 aesService.setKey1(*BinaryToString*(key1));  
 int[] tmp=aesService.SAESDecrypy(cypherText);  
 if(midText.contains(*BinaryToString*(tmp))){  
 int j=midText.indexOf(*BinaryToString*(tmp));  
 int[] key0=*DecimalToBinary*(j);  
 AesService aesServiceNew=new AesService();  
 aesServiceNew.setKey1(*BinaryToString*(key0));  
 aesServiceNew.setKey2(*BinaryToString*(key1));  
 int[] res=aesServiceNew.DoubleAESEncrypt(plainText2);  
 if(cypherText2.equals(*BinaryToString*(res))){  
 StringBuilder ans=new StringBuilder();  
 ans.append(*BinaryToString*(key0));  
 ans.append(*BinaryToString*(key1));  
 possibleKeys.add(ans.toString());  
 }  
 }  
 }  
 return possibleKeys;  
 }  
 public static Set<String> MidAttack(List<String> plainTexts,List<String> cypherTexts){  
 Set<String> ans=new HashSet<>();  
 while(plainTexts.isEmpty()==false){  
 String plaintext1= plainTexts.remove(0);  
 String cypherText1=cypherTexts.remove(0);  
 StringBuilder plaintext2=new StringBuilder();  
 StringBuilder cypherText2=new StringBuilder();  
 if(plainTexts.isEmpty()){  
 plaintext2.append(plaintext1);  
 cypherText2.append(cypherText1);  
 }else{  
 plaintext2.append(plainTexts.remove(0));  
 cypherText2.append(cypherTexts.remove(0));  
 }  
 if(ans.isEmpty()==true){  
 ans.addAll(*Attack*(plaintext1,cypherText1,plaintext2.toString(),cypherText2.toString()));  
 }else{  
 ans.retainAll(*Attack*(plaintext1,cypherText1,plaintext2.toString(),cypherText2.toString()));  
 }  
 if(ans.isEmpty() || ans.size()==1){  
 return ans;  
 }  
 }  
 return ans;  
 }  
  
}

## 项目初步测试结果

项目开发完成后，我们对其功能进行了测试，并对项目进行了交互测试。测试结果显示项目功能执行正常，项目功能相对完善，具有良好的实用性。其中加解密结果的一键复制功能具有较好的用户友好性和领先性。

*S-Aes项目开发手册到此结束，感谢您的阅读。*