# S-AES 加密解密程序开发手册

## 1. 简介

S-AES 加密解密程序是一种轻量级的对称加密算法，适用于资源受限的设备和应用场景。它支持 128 位密钥长度的加密和解密操作，并提供了简洁高效的加密解密算法。该程序适用于对小型数据进行加密解密，如文本消息或二进制数据。通过使用 S-AES 加密解密程序，用户可以保护敏感信息的安全性，并在需要时方便地进行加密解密操作。

在这份开发手册中，您将找到以下内容：

(1)组件及接口概述：介绍了加密解密程序的各个组件和对外提供的接口。

(2)使用示例：提供了一些示例代码，展示了如何使用加密解密程序完成加密和解密操作。

(3)注意事项：列出了在使用加密解密程序时需要注意的事项和建议。

通过阅读本开发手册，您将了解如何正确使用 S-AES 加密解密程序，以及如何保护您的数据安全。请确保遵守安全性原则，并妥善保管生成的密钥，以免数据被未授权的人员获取。

## 2. 组件及接口概述

本程序包含以下组件及其对应的接口：

### 2.1 使用 AES 加密和解密：

代码提供了一些示例用法，包括常规加解密和 ASCII 加解密。这些示例演示了如何使用 AE 加密算法对数据进行加密和解密。

### 2.2 多重加密：

提供了双重加密和三重加密的示例。这些示例演示了如何多次使用不同的 AES 密钥对数据进行加密和解密。

### 2.3 CBC 模式：

代码演示了密码分组链接模式（CBC 模式）的使用，其中明文数据被分成块并使用 AES 加密算法进行加密。CBC 模式在加密前会对每个块进行异或操作，以增加安全性。

### 2.6 GUI 界面展示：

使用 java swing 设计并实现程序的 UI 界面.

## 3. 使用示例（基于 java）

### 3.1 密钥生成

```
String initialKey = "2D55"; // Initial 16-bit key
String[] expandedKeys = expandKey(initialKey);

// Print expanded keys in binary
for (int i = 0; i < expandedKeys.length; i++) {
System.out.println("w" + i + ": " + expandedKeys[i]);
}
```

### 3.2 加解密示例

```
String initialKey = "2D55"; // 初始 16 位密钥
String plaintext = "0000000000000000";
String encryptedText = encryptText(plaintext, initialKey);
System.out.println("加密结果如下：");
System.out.println(encryptedText); // 输出秘文结果
```

```
String plaintextASCII = "wpk in software enginer and this is a test txt file";
// 输入 String 类型的 ASCII
String encryptedTextASCII = encryptASCII(plaintextASCII, initialKey);
System.out.println("加密结果如下：");
System.out.println(encryptedTextASCII); // 输出加密结果


String initialKey = "2D55"; // 初始 16 位密钥
String ciphertext = "1011100011100001";
String encryptedText = decryptText(ciphertext, initialKey);
System.out.println("解密结果如下：");
System.out.println(encryptedText); // 输出秘文结果
String encryptedTextASCII = "¯\u008C%(\u0098[ÎN·yag\u001D/"; // ASCII 编码的
加密文本
String decryptedTextASCII = decryptASCII(encryptedTextASCII, initialKey);
System.out.println("解密结果如下：");
System.out.println(decryptedTextASCII); // 输出解密结果
```

## 3.3 多重加/解密示例

```
// 定义两个密钥和明文
String key1 = "2D55";
String key2 = "A1F3";
String plaintext = "0000000000000000";
// 调用 doubleEncrypt 方法测试双重加密
String doubleEncryptedText = doubleEncrypt(plaintext, key1, key2);
System.out.println("双重加密结果如下：");
System.out.println(doubleEncryptedText); // 输出双重加密的密文
String key3 = "B276";
// 调用 tripleEncrypt 方法测试三重加密
String tripleEncryptedText = tripleEncrypt(plaintext, key1, key2, key3);
System.out.println("三重加密结果如下：");
System.out.println(tripleEncryptedText); // 输出三重加密的密文

// 定义两个密钥和密文
String key1 = "2D55";
String key2 = "A1F3";
String ciphertext = "0000001111011100";
// 调用 doubleDecrypt 方法测试双重解密
String doubleEncryptedText = doubleDecrypt(ciphertext, key1, key2);
System.out.println("双重解密结果如下：");
System.out.println(doubleEncryptedText); // 输出双重加密的密文
String key3 = "B276";
```

```
String tripleCiphertext="1010101001111001";
// 调用 tripleDecrypt 方法测试三重解密
String tripleDecryptedText = tripleDecrypt(tripleCiphertext, key1, key2,
key3);
System.out.println("三重解密结果如下：");
System.out.println(tripleDecryptedText); // 输出三重解密的明文
```

### 3.4 CBC 加/解密示例

```
String plaintext =
"0000000011111111111111110000000011111111111111110000000000000000"; //
64-bit 长明文 00FF FF00 FFFF 0000
String initialVector = "1010101010101010"; // 协商好的 16 位初始向量
String key = "2D55"; // 16-bit key
String encryptedText = CBCModeEncrypt(plaintext, initialVector, key);
System.out.println("加密结果如下：");
System.out.println(encryptedText); // 输出加密结果
```

```
String encryptedText =
"1101110011010101001111010000111100111100000001111110000110001011"; //
64-bit 长明文 00FF FF00 FFFF 0000
String initialVector = "1010101010101010"; // 协商好的 16 位初始向量
String key = "2D55"; // 16-bit key
String decryptedText = CBCModeDecrypt(encryptedText, initialVector, key);
System.out.println("解密结果如下：");
System.out.println(decryptedText); // 输出解密结果
```

## 4. 注意事项

在使用加密解密功能之前，请确保输入的密钥和数据符合算法要求。

请妥善保管密钥，不要将密钥泄露给他人。

本程序仅适用于小规模的数据加密需求，不建议用于对大量或敏感数据进行加密。