**S-AES 加密解密系统 - 用户指南**

**1. 系统概述**

本系统基于 S-AES（简化 AES）算法实现完整加密解密功能，覆盖 5 个核心测试关卡，满足不同场景下的加密需求与安全性验证：

* 第 1 关：基础 S-AES 加解密（16 位密钥）
* 第 2 关：交叉测试验证（多平台一致性校验）
* 第 3 关：ASCII 扩展（字符串加解密）
* 第 4 关：多重加密与密码分析（双重 / 三重加密、中间相遇攻击）
* 第 5 关：CBC 工作模式（分组链加密及错误传播测试）

**2. 用户操作指南**

**2.1 启动系统**

bash

# 编译所有Java文件

javac SimpleSAES.java ASCIIExtension.java DoubleEncryption.java TripleEncryption.java MeetInMiddleAttack.java CBCMode.java CompleteSAESGUI.java

# 运行GUI主程序

java CompleteSAESGUI

**2.2 分关卡操作流程**

**2.2.1 第 1 关：基本 S-AES 加解密**

1. 关卡选择：下拉框选择「第 1 关：基本 S-AES (16 位密钥)」
2. 输入参数：
   * 16 位二进制数据：填写 1010110101010101（或自定义 16 位 0/1 组合）
   * 16 位二进制密钥：填写 1100110011001100（或自定义 16 位 0/1 组合）
3. 功能选择：点击「加密」或「解密」
4. 执行操作：点击「执行」按钮，查看输出结果

**2.2.2 第 2 关：交叉测试验证**

* 操作流程与第 1 关完全一致，核心用于与其他实现版本对比输出结果，验证算法标准性

**2.2.3 第 3 关：ASCII 扩展**

1. 关卡选择：下拉框选择「第 3 关: ASCII 扩展 (16 位密钥)」
2. 输入参数：
   * ASCII 字符串：填写文本（如 Hello、Hi 等普通字符）
   * 16 位二进制密钥：填写 1100110011001100（或自定义）
3. 功能选择：点击「加密」或「解密」
4. 执行操作：系统自动完成字符串与二进制转换，输出结果

**2.2.4 第 4 关：多重加密与密码分析**

**4.1 双重加密（32 位密钥）**

1. 关卡选择：下拉框选择「第 4 关.1: 双重加密 (32 位密钥)」
2. 输入参数：
   * 数据：支持 16 位二进制或 ASCII 字符串
   * 密钥 1：16 位二进制（如 1100110011001100）
   * 密钥 2：16 位二进制（如 1010101010101010）
3. 执行操作：点击「加密」或「解密」，系统按 E(K2, E(K1, P))/D(K1, D(K2, C)) 逻辑运算

**4.2 三重加密（48 位密钥）**

1. 关卡选择：下拉框选择「第 4 关.3: 三重加密 (48 位密钥)」
2. 输入参数：
   * 数据：支持 16 位二进制或 ASCII 字符串
   * 密钥 1 / 密钥 2 / 密钥 3：各 16 位二进制（如 CCCC、AAAA、F0F0）
3. 执行操作：点击「加密」或「解密」，系统按 E(K3, D(K2, E(K1, P)))/D(K1, E(K2, D(K3, C))) 逻辑运算

**4.3 中间相遇攻击**

1. 关卡选择：下拉框选择「第 4 关.2: 中间相遇攻击」
2. 输入参数：
   * 已知明文 1：16 位二进制（如 0000000000000000）
   * 对应密文 1：16 位二进制（如 0010000011000001）
   * 可选补充：明文 2 + 密文 2（多对明密文提升攻击准确性）
3. 执行操作：点击「执行攻击」，系统遍历密钥空间查找匹配密钥对

**2.2.5 第 5 关：CBC 工作模式**

**5.1 基础 CBC 加解密**

1. 关卡选择：下拉框选择「第 5 关: CBC 工作模式 (16 位密钥)」
2. 输入参数：
   * ASCII 文本：填写任意长度字符串（如 Hello World）
   * 16 位二进制密钥：填写 1100110011001100（或自定义）
   * 初始向量 IV：16 位二进制（如 1010101010101010）
3. 执行操作：点击「加密」或「解密」，系统自动分块处理

**5.2 CBC 错误传播测试**

1. 关卡选择：下拉框选择「第 5 关: CBC 错误传播测试」
2. 输入参数：
   * 原始密文：完整 CBC 加密输出结果
   * 篡改密文：修改原始密文中任意分组的部分比特
   * 16 位二进制密钥：与加密时一致
3. 执行操作：点击「分析」，查看错误在解密过程中的传播效果

**3. 注意事项**

**3.1 输入格式要求**

* 16 位二进制数据：必须严格为 16 个 0/1 字符组合，不可多填、少填
* ASCII 字符串：仅支持普通文本字符，不支持特殊符号、表情等
* 密钥：基础 / ASCII/CBC 模式需 16 位二进制；双重加密需 32 位（2 个 16 位）；三重加密需 48 位（3 个 16 位）

**3.2 操作建议**

* 快速测试：点击「测试示例」按钮，自动填入对应关卡的预设测试数据
* 复杂操作：建议先阅读本指南，明确输入格式和操作逻辑后再执行
* 性能提示：中间相遇攻击需遍历大量密钥组合，可能耗时较长，建议耐心等待

**3.3 错误处理**

* 输入格式错误：系统会弹出提示，说明正确格式要求（如「密钥需为 16 位二进制」）
* 运行时错误：输出面板会显示详细错误信息，可按提示排查问题（如密钥不匹配、密文篡改等）

**S-AES 加密解密系统 - 开发手册**

**1. 系统架构**

**1.1 核心组件结构**

plaintext

S-AES加密解密系统

├── SimpleSAES.java # 核心算法：基础S-AES加解密实现

├── ASCIIExtension.java # 扩展功能：ASCII字符串与二进制转换及加解密

├── DoubleEncryption.java # 多重加密：双重S-AES加密/解密

├── TripleEncryption.java # 多重加密：三重S-AES加密/解密

├── MeetInMiddleAttack.java # 密码分析：中间相遇攻击实现

├── CBCMode.java # 工作模式：CBC模式加解密及错误传播测试

└── CompleteSAESGUI.java # 交互界面：主GUI界面（整合所有功能）

**1.2 组件依赖关系**

* 主界面 CompleteSAESGUI：依赖所有其他功能模块
* 多重加密模块（DoubleEncryption/TripleEncryption）：依赖 SimpleSAES
* CBC 模式模块（CBCMode）：依赖 SimpleSAES 和 ASCIIExtension
* 中间相遇攻击模块（MeetInMiddleAttack）：依赖 SimpleSAES

**2. 核心算法接口文档**

**2.1 SimpleSAES 类（核心算法）**

**功能描述**

实现 S-AES 标准加解密逻辑，基于 GF (2⁴) 有限域运算，包含密钥扩展、轮函数（半字节代替、行移位、列混淆、密钥加）等核心操作。

**主要方法**

java

运行

/\*\*

\* 16位明文加密

\* @param plaintext 16位二进制明文（仅含0/1，长度16）

\* @param key 16位二进制密钥（仅含0/1，长度16）

\* @return 16位二进制密文

\*/

public static String encrypt(String plaintext, String key)

/\*\*

\* 16位密文解密

\* @param ciphertext 16位二进制密文（仅含0/1，长度16）

\* @param key 16位二进制密钥（与加密时一致）

\* @return 16位二进制明文

\*/

public static String decrypt(String ciphertext, String key)

**算法特性**

* 加密流程：3 轮操作（第 0 轮：密钥加；第 1 轮：完整轮；第 2 轮：简化轮）
* 基础组件：标准 S-AES S 盒、逆 S 盒，GF (2⁴) 有限域加法 / 乘法表

**2.2 ASCIIExtension 类（ASCII 扩展）**

**功能描述**

实现 ASCII 字符串与二进制的双向转换，并基于 SimpleSAES 完成字符串加解密。

**主要方法**

java

运行

/\*\*

\* ASCII字符串转二进制字符串

\* @param asciiText 普通ASCII字符串（无特殊符号）

\* @return 二进制字符串（长度为8的整数倍）

\*/

public static String asciiToBinary(String asciiText)

/\*\*

\* 二进制字符串转ASCII字符串

\* @param binaryText 二进制字符串（长度为8的整数倍）

\* @return 普通ASCII字符串

\*/

public static String binaryToAscii(String binaryText)

/\*\*

\* ASCII字符串加密

\* @param plaintext 普通ASCII字符串

\* @param key 16位二进制密钥

\* @return 二进制密文字符串

\*/

public static String encryptAscii(String plaintext, String key)

/\*\*

\* ASCII字符串解密

\* @param ciphertext 二进制密文字符串

\* @param key 16位二进制密钥

\* @return 普通ASCII明文字符串

\*/

public static String decryptAscii(String ciphertext, String key)

**2.3 DoubleEncryption 类（双重加密）**

**功能描述**

基于 SimpleSAES 实现双重加密逻辑，密钥长度 32 位（2 个 16 位密钥）。

**主要方法**

java

运行

/\*\*

\* 双重加密

\* @param plaintext ASCII明文字符串

\* @param key 32位二进制密钥（key1+key2，各16位）

\* @return 二进制密文字符串

\* @算法逻辑 E(K2, E(K1, P))

\*/

public static String encrypt(String plaintext, String key)

/\*\*

\* 双重解密

\* @param ciphertext 二进制密文字符串

\* @param key 32位二进制密钥（与加密时一致）

\* @return ASCII明文字符串

\* @算法逻辑 D(K1, D(K2, C))

\*/

public static String decrypt(String ciphertext, String key)

**2.4 TripleEncryption 类（三重加密）**

**功能描述**

基于 SimpleSAES 实现三重加密逻辑，密钥长度 48 位（3 个 16 位密钥）。

**主要方法**

java

运行

/\*\*

\* 三重加密

\* @param plaintext ASCII明文字符串

\* @param key 48位二进制密钥（key1+key2+key3，各16位）

\* @return 二进制密文字符串

\* @算法逻辑 E(K3, D(K2, E(K1, P)))

\*/

public static String encrypt(String plaintext, String key)

/\*\*

\* 三重解密

\* @param ciphertext 二进制密文字符串

\* @param key 48位二进制密钥（与加密时一致）

\* @return ASCII明文字符串

\* @算法逻辑 D(K1, E(K2, D(K3, C)))

\*/

public static String decrypt(String ciphertext, String key)

**2.5 MeetInMiddleAttack 类（中间相遇攻击）**

**功能描述**

针对双重加密的密钥恢复攻击，通过 “空间换时间” 策略遍历密钥空间，匹配中间值。

**主要方法**

java

运行

/\*\*

\* 单对明密文中间相遇攻击

\* @param plaintext 已知16位二进制明文

\* @param ciphertext 对应16位二进制密文

\* @return 找到的密钥对（格式："key1,key2"，均为16位二进制）

\*/

public static String attack(String plaintext, String ciphertext)

/\*\*

\* 多对明密文中间相遇攻击（提升准确性）

\* @param plaintext1 已知明文1（16位二进制）

\* @param ciphertext1 对应密文1（16位二进制）

\* @param plaintext2 已知明文2（16位二进制）

\* @param ciphertext2 对应密文2（16位二进制）

\* @return 唯一匹配的密钥对（格式："key1,key2"）

\*/

public static String attack(String plaintext1, String ciphertext1, String plaintext2, String ciphertext2)

**2.6 CBCMode 类（CBC 工作模式）**

**功能描述**

实现 S-AES 的 CBC（密码分组链）工作模式，支持长文本加密及错误传播测试。

**主要方法**

java

运行

/\*\*

\* CBC模式加密

\* @param plaintext ASCII明文字符串

\* @param key 16位二进制密钥

\* @param iv 16位二进制初始向量

\* @return 格式化密文（含IV+各密文块，用分隔符区分）

\*/

public static String encrypt(String plaintext, String key, String iv)

/\*\*

\* CBC模式解密

\* @param ciphertext 格式化密文（加密输出的完整结果）

\* @param key 16位二进制密钥

\* @return ASCII明文字符串

\*/

public static String decrypt(String ciphertext, String key)

/\*\*

\* CBC错误传播测试

\* @param originalCipher 原始格式化密文

\* @param tamperedCipher 篡改后的格式化密文

\* @param key 16位二进制密钥

\* @return 错误传播分析结果（文本描述）

\*/

public static String errorPropagationTest(String originalCipher, String tamperedCipher, String key)

**3. GUI 接口文档**

**3.1 CompleteSAESGUI 类（主界面）**

**功能描述**

整合所有功能模块的可视化交互界面，支持动态显示输入字段、执行操作、输出结果。

**核心界面组件**

* 关卡选择下拉框：列出 5 个核心关卡
* 操作选择按钮：加密 / 解密 / 执行攻击 / 分析
* 输入参数面板：根据所选关卡动态显示需填写的参数（如密钥、明文、IV 等）
* 输出结果面板：显示执行后的结果（密文 / 明文 / 密钥 / 分析报告）
* 辅助按钮：「测试示例」（填充预设数据）、「清空」（清空输入输出）

**主要方法**

java

运行

/\*\*

\* 根据所选关卡更新输入字段（显示/隐藏对应参数框）

\*/

private void updateInputFields()

/\*\*

\* 执行用户选择的操作（加密/解密/攻击/分析）

\*/

private void performOperation()

/\*\*

\* 填充当前关卡的预设测试数据

\*/

private void testButtonAction()

**4. 开发指南**

**4.1 编译与运行**

bash

# 编译所有Java文件（确保所有类在同一目录）

javac \*.java

# 运行主GUI程序

java CompleteSAESGUI

**4.2 扩展开发**

**4.2.1 添加新加密模式**

1. 新建 Java 类（如 ECBMode.java），参考 CBCMode.java 结构
2. 依赖 SimpleSAES 实现新模式的加解密逻辑
3. 在 CompleteSAESGUI 中添加新关卡选项，更新 updateInputFields() 方法显示对应参数
4. 在 performOperation() 中添加新模式的执行逻辑

**4.2.2 修改 GUI 界面**

1. 调整界面布局：修改 CompleteSAESGUI 中的 Swing 组件布局代码
2. 新增输入字段：在参数面板中添加新组件，并通过 updateInputFields() 控制显示逻辑
3. 优化交互体验：添加异步处理（如中间相遇攻击的后台线程），避免界面卡顿

**4.2.3 添加新攻击方法**

1. 新建 Java 类（如 BruteForceAttack.java），实现攻击逻辑
2. 依赖 SimpleSAES 完成加密 / 解密验证
3. 在 GUI 中添加对应关卡选项及操作按钮，关联攻击方法

**4.3 测试建议**

1. 功能验证：使用「测试示例」中的预设数据，验证加解密可逆性
2. 边界测试：测试空输入、格式错误输入、最大长度文本等场景
3. 标准性测试：对比 S-AES 官方测试向量，确保核心算法输出一致
4. 兼容性测试：在不同 Java 版本（JDK8+）中验证运行稳定性

**4.4 性能优化**

* 中间相遇攻击：使用哈希表（HashMap）存储正向计算结果，提升查找效率
* 大文本处理：CBC 模式中采用分块异步处理，避免长时间阻塞界面
* 重复计算：缓存密钥扩展结果，避免同一密钥多次扩展

**5. 故障排除**

**5.1 常见问题及解决方案**

| **问题类型** | **可能原因** | **解决方案** |
| --- | --- | --- |
| 编译错误 | Java 版本不兼容、类文件缺失 | 升级 JDK 至 8+、确保所有类文件在同一目录 |
| 运行时抛出异常 | 输入格式错误（如密钥长度不足） | 检查输入是否符合要求，添加格式校验逻辑 |
| 加解密不可逆 | 算法逻辑错误（如轮函数顺序错误） | 对比官方流程，调试中间结果（如密钥扩展、列混淆） |
| GUI 显示异常 | 组件布局代码错误、字段可见性设置 | 检查 updateInputFields() 中的显示逻辑 |
| 攻击无结果 | 明密文不匹配、密钥空间遍历不完整 | 验证明密文对应关系、优化遍历逻辑 |

**5.2 调试建议**

1. 控制台输出调试：在核心方法中添加 System.out.println()，打印中间结果（如密钥扩展后的轮密钥、每轮变换后的状态）
2. 单元测试：为每个核心类编写独立测试用例，验证单个方法的输出正确性
3. 对比调试：与 S-AES 官方测试向量的中间结果逐一对比，定位首次出现差异的环节
4. 日志记录：添加日志输出功能，记录操作过程和错误信息，便于追溯问题