开发手册

**1.SAES加解密**

**1.1 系统概述**

S-AES加解密程序采用Java语言实现，基于Java Swing库构建用户界面，分为加解密核心算法和图形界面两大部分。核心算法基于简化版的AES（S-AES）算法，通过字节代换、行移位、列混合和密钥扩展等步骤，实现加解密功能。图形界面模块提供用户输入、操作选择和结果展示功能，支持基本加密、双重加密和三重加密模式。

**1.2 系统架构**

该程序主要由两个核心部分组成：

S-AES加解密逻辑：负责加密和解密的算法实现。

图形用户界面（GUI）：通过Java Swing实现，负责用户交互和结果展示。

代码模块设计如下：

SAES类：加解密算法核心类。

GUI类：图形用户界面类，用于接收用户输入、触发加解密操作、展示结果。

**1.3 核心算法实现**

1.3.1 字节代换（SubBytes）

在S-AES中，字节代换通过S-盒（S-Box）完成。S-盒是一种固定的查找表，用于将一个字节映射到另一个字节。加密时使用正向S-盒（S\_BOX），解密时使用逆向S-盒（INVERSE\_S\_BOX）。

实现流程：

加密：在subBytes()方法中，遍历每个字节，使用S\_BOX进行代换。

解密：在inverseSubBytes()方法中，使用INVERSE\_S\_BOX进行逆代换。

1.3.2 行移位（ShiftRows）

行移位用于对状态矩阵的行进行循环移位，增强数据的混合性。在S-AES中，行移位只涉及单字节移位。

实现流程：

加密：通过shiftRows()方法，将状态矩阵的行循环移位。

解密：通过inverseShiftRows()方法，实现逆方向移位。

1.3.3 列混合（MixColumns）

列混合通过伽罗瓦域GF(2^4)实现多项式运算，以增加数据的扩散性。

实现流程：

加密：mixColumns()方法通过矩阵乘法实现列混合。

解密：inverseMixColumns()方法实现逆列混合。

1.3.4 密钥扩展（Key Expansion）

密钥扩展将用户输入的16位密钥扩展为多轮密钥，供各轮加解密使用。扩展过程中使用轮常数（Rcon）来增加密钥的非线性。

实现流程：

expandKey()方法生成轮密钥，rcon()方法生成轮常数。

**1.4 GUI 实现**

1.4.1 界面布局

图形界面通过GridLayout布局，实现输入框、操作按钮、结果区域的清晰分布。

界面主要元素包括：

输入文本框：用于输入需要加解密的数据。

密钥输入框：用于输入16位密钥。

操作选择区域：加密或解密选择。

加密类型选择框：选择基本加密、双重加密或三重加密。

结果显示区域：不可编辑的文本框，用于显示加解密后的结果。

1.4.2 输入验证

为了确保用户输入的正确性，GUI界面在按钮点击时会进行输入检测，包括：

检查数据是否为二进制或ASCII字符串。

检查密钥格式是否为16位整数。

如果输入无效，将在界面中弹出错误提示。

1.4.3 功能按钮实现

开始按钮：根据选择的操作模式，调用相应的加密或解密函数。

1.4.4 加解密模式选择

根据用户的选择，通过SAES类的不同方法执行基本加密、双重加密或三重加密。GUI界面通过组合框（JComboBox）提供加密模式的选择。

**1.5. 扩展功能**

1.5.1 加密模式扩展

开发者可以在SAES类中增加更多加密模式，或修改现有模式的加密轮数。例如：

增加四重加密模式。

扩展密钥的位数，增强加密强度。

1.5.2 支持更多输入格式

可以增加对十六进制或Base64输入的支持，以适应不同场景的需求。此功能可以通过扩展parseKey()和输入验证方法来实现。

1.5.3 错误处理改进

增强输入验证与错误处理模块，以便为用户提供更加清晰的错误提示。例如，检查密钥格式并提示“密钥必须为16位整数”。

**2.MiddleMeetAttackGUI 类及密钥查找程序**

**2.1 概述**

MiddleMeetAttackGUI 类提供了一个基于 Java Swing GUI 的应用，执行 S-AES（简化高级加密标准）算法的中间相遇攻击。通过使用用户输入的明密文对组合，程序尝试寻找匹配的密钥对 key1 和 key2。

**2.2 类和方法概览**

MiddleMeetAttackGUI：主类，继承自 JFrame，构建 GUI 和相关功能。

PCPair：嵌套类，用于存储和管理明密文对数据。

核心方法：

updateInputPairsPanel()：动态更新明密文对输入面板。

findKey()：执行中间相遇攻击算法，查找密钥对。

formatTime(long nanos)：格式化和显示已用时间。

stringToArray(String str)：将二进制字符串转换为数组。

arrayToString(int[] arr)：将数组转换为二进制字符串。

intToBits(int n)：将整数转换为 16 位二进制数组。

**2.3功能详细描述**

1. MiddleMeetAttackGUI 构造函数

初始化 GUI 组件和布局，包括主面板、输入明密文对的面板和结果显示面板。

设置“查找密钥”按钮的事件监听器，并创建计时器组件。

2. updateInputPairsPanel() 方法

作用：根据用户选择的明密文对数量，动态生成输入面板。

逻辑：获取 pairCountComboBox 中的值（明密文对数量），然后在 inputPairsPanel 中生成相应数量的输入文本框。

流程：用户每次更改明密文对数量时，调用此方法清空并重新绘制输入组件。

3. findKey() 方法

作用：核心功能，通过多线程执行中间相遇攻击算法，尝试查找匹配的密钥对。

流程：

从用户界面获取每对明密文并存储到 PCPair 列表中。

创建线程池以实现并行处理。

对每个可能的 key1 和 key2 组合执行加密解密操作，并与用户输入的明密文对进行比较。

若找到匹配的密钥对，将其添加到 foundKeys1 和 foundKeys2 列表中。

若找到了至少一个匹配对，将密钥对输出到 resultArea 中，否则显示“未找到密钥”。

4. formatTime(long nanos) 方法

作用：将以纳秒计的时间格式化为秒、毫秒、微秒和纳秒的组合，以提升显示易读性。

逻辑：纳秒转换为微秒、毫秒和秒，并在 timerLabel 中显示已用时间。

5. stringToArray(String str) 和 arrayToString(int[] arr)

stringToArray(String str)：将用户输入的二进制字符串转换为整数数组，并检查是否为有效二进制格式。

arrayToString(int[] arr)：将整数数组转换为二进制字符串表示形式。

6. intToBits(int n) 方法

作用：将整数转换为对应的 16 位二进制数组表示，用于密钥位处理。

流程：利用位移和按位操作，将整数转换为二进制数组。

7. PCPair 类

作用：用于表示和存储单对明密文数据。

字段：

plaintext：明文的 16 位二进制数组。

ciphertext：对应密文的 16 位二进制数组。

方法：

getPlaintext()：返回明文数组。

getCiphertext()：返回密文数组。

**2.4 GUI 结构和设计**

1.组件和布局：

mainPanel：主面板，包含明密文对数量面板、输入面板和结果显示面板。

pairCountPanel：显示明密文对数量选择的下拉菜单。

inputPairsPanel：用于输入明密文对的动态生成面板。

buttonPanel：包含“查找密钥”按钮和计时器标签。

resultArea：显示找到的密钥对的文本区域。

2.事件处理：

通过 ActionListener 实现对按钮和计时器的事件监听。

“查找密钥”按钮会启动 findKey() 方法，而计时器每秒更新一次显示。

**2.5 线程池和并行计算**

使用 ExecutorService 实现线程池管理，以便在多核环境下并行处理 key1 和 key2 的密钥查找，提升效率。

设置线程池大小为系统可用核心数，以优化资源使用。

**2.6性能和注意事项**

线程管理：确保 executorService 在密钥查找完成后被正确关闭，避免资源泄露。

计时器：在用户点击“查找密钥”后启动计时器，在查找结束后关闭。

验证输入：检查每对明密文是否为有效 16 位二进制字符串，并在无效时提示用户重新输入。

**3.CBC加解密**

**3.1概述**

本程序是一个采用 Java Swing 构建的图形化用户界面（GUI）应用，用于实现基于分组密码的 CBC（Cipher Block Chaining）模式加解密功能。主要功能包括：二进制和 ASCII 格式的输入，16 位密钥的加密解密选择，生成初始向量（IV）以及显示最终结果。

**3.2 代码结构**

CBC.java：实现 CBC 加解密功能的核心逻辑，包括 generateIV、encrypt\_CBC 和 decrypt\_CBC 方法。

CBC\_GUI.java：图形化用户界面，用户可以通过界面输入加密/解密信息，选择输入方式，执行 CBC 操作，并在界面上显示结果。

**3.3 详细说明**

3.3.1 CBC 类

IV生成:

generateIV 方法用于生成 16 位随机 IV（初始向量），将其作为加密链的第一个密文分组。

加密方法:

encrypt\_CBC 方法接收明文、密钥和 IV，通过 CBC 模式对分组进行加密，将明文分成 16 位长度的分组，对每组应用异或操作并使用 S-AES 加密。

解密方法:

decrypt\_CBC 方法接收密文、密钥和 IV，对分组逐一进行解密，然后通过异或还原明文。

3.3.2 CBC\_GUI 类

构造函数:

CBC\_GUI 构造函数生成 GUI 界面，定义了输入、选择和输出区域。使用 JRadioButton 实现单选输入方式和操作类型选择，JTextArea 用于明文、密文和结果显示。

核心操作逻辑:

handle 方法处理用户的加密/解密请求。它根据用户选择的输入格式（ASCII 或二进制）解析输入，进行分组，调用相应的加解密方法，并将结果显示在 GUI 中。

结果显示: resultContent.setText(result); 用于将加密/解密结果更新至结果文本区域。

**3.4运行流程**

初始化：运行主方法生成一个随机 IV 并调用 CBC\_GUI 实例化 GUI。

输入数据：用户在 GUI 输入框中填写明文/密文和密钥。

执行操作：点击“开始执行”按钮，触发 handle 方法。