AES 算法 UI 开发手册

1. 项目概述

1.1 项目目标和目的

本项目旨在开发一个基于 AES（高级加密标准）算法的加密系统，提供安全可靠的数据加密和解密服务。系统将支持多种加密模式，包括一重、二重、三重加密以及针对字符的加密，以满足不同场景下的安全需求。目的是为用户提供一个简单、直观且高效的工具，以保护他们的敏感数据免受未授权访问和泄露。

1.2 项目范围

加密功能: 提供对文本和文件的 AES 加密功能，支持多种加密模式和密钥长度。
解密功能: 提供对使用本系统加密的文本和文件的解密功能。
用户界面: 提供一个简洁且用户友好的界面，让用户能够轻松地进行加密和解密操作。
性能: 优化算法性能，确保加密和解密操作的高效进行。

1.3 目标用户或受众

个人用户: 希望保护其个人文件和通信安全的用户。
企业用户: 需要保护商业机密和客户数据的小型和中型企业。
开发者: 需要在自己的项目中集成加密功能的软件开发者。

2. 环境设置

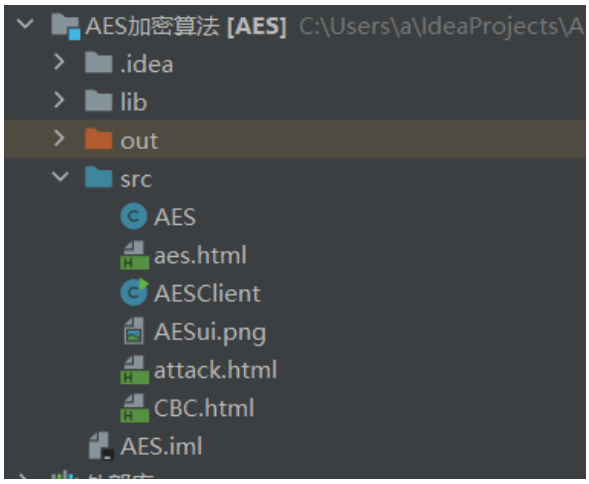
本项目使用 IntelliJ IDEA 2023.1.3 进行开发，前端界面使用 HTML 搭建，算法逻辑使用 Java 编写，并利用 JavaFX 工具实现在 HTML 中调用 Java 方法。

2.1 开发工具和软件

IntelliJ IDEA 2023: IntelliJ IDEA 用于 Java 代码的编写和项目管理。
Java JDK: java8
JavaFX SDK: 用于在 HTML 界面中嵌入和调用 Java 方法。
任意现代 Web 浏览器: 用于展示和交互 HTML 界面，如 Google Chrome、Mozilla Firefox 或 Microsoft Edge。

2.2 项目结构

项目结构如图所示，AES.java 中实现了 AES 加密的所有相关方法，aes.html 是 AES 算法的主 ui，attack.html、CBC.html 分别是暴力破解密钥、CBC 加密算法的界面，AESClient 则是使用 javaFX 以便在 html 界面中调用 java 编写的加密算法，需要运行的就是该文件



3. 代码管理

aes.html:该文件为 aes 主界面的 html 代码，以下是其 html、css、js 的一部分

```
<div class="aes-box">
    <h2>AES加密</h2>
    <form>
        <div class="user-box">
            <input type="text" id="before" required="">
            <label>请输入加密/解密的内容</label>
        </div>
        <div class="user-box">
            <input type="text" name="key" id="key" required="">
            <label>请输入密钥</label>
        </div>
        <div class="user-box">
            <input type="text" name="result" required="" readonly>
            <label>结果展示</label>
        </div>
        <div class="encryption-level">
            <input type="radio" id="single" name="encryption" value="single" checked>
            <label for="single">一重加密</label>
            <input type="radio" id="double" name="encryption" value="double">
            <label for="double">二重加密</label>
            <input type="radio" id="triple" name="encryption" value="triple">
            <label for="triple">三重加密</label>
        </div>
        <a href="#">
            <span></span>
```

```
<script>
    1 个用法
    function singleEncrypt() {
        var before = document.getElementById("before").value;
        var key0 = document.getElementById("key").value;
        return "加密结果: " + java.encryption(before, key0);
    }

    1 个用法
    function singleDecrypt() {
        var before = document.getElementById("before").value;
        var key0 = document.getElementById("key").value;
        return "解密结果: " + java.decryption(before, key0);
    }

    1 个用法
    function doubleEncrypt() {
        var before = document.getElementById("before").value;
        var key0 = document.getElementById("key").value;
        return "加密结果: " + java.double_encryption(before, key0);
    }

    1 个用法
    function doubleDecrypt() {
        var before = document.getElementById("before").value;
        var key0 = document.getElementById("key").value;
        return "解密结果: " + java.double_decryption(before, key0);
    }
```

```
<style>
    html {
        height: 100%;
    }
    body {
        margin:0;
        padding:0;
        font-family: sans-serif;
        background-image: url('AESui.png');
    }

    .aes-box {
        position: absolute;
        top: 50%;
        left: 50%;
        width: 400px;
        padding: 40px;
        transform: translate(-50%, -50%);
        background: rgba(0,0,0,.7);
        box-sizing: border-box;
        box-shadow: 0 15px 25px rgba(0,0,0,0.6);
        border-radius: 10px;
    }

    .aes-box h2 {
        margin: 0 0 30px;
        padding: 0;
        color: #fff;
        text-align: center;
```

AES.java:此为实现 AES 算法逻辑的部分，以下是加密、解密的核心部分

```java
public String encryption(String ming, String key) {
    //获得密钥
    char[][] keys = get_key(key);
    //存储当前状态
    char[][] state = new char[2][8];
    //轮密钥加（异或）
    state[0] = XOR(ming.substring(0,8).toCharArray(), keys[0]);
    state[1] = XOR(ming.substring( beginIndex: 8).toCharArray(), keys[1]);

    //半字节代替,与SubNib操作一样。
    state[0] = SubNib(state[0]);
    state[1] = SubNib(state[1]);

    //行移位,
    state[1] = RotNib(state[1]);

    //列混淆
    state = mix(martrix,state);

    //轮密钥加
    state[0] = XOR(state[0], keys[2]);
    state[1] = XOR(state[1], keys[3]);

    //半字节代替。
    state[0] = SubNib(state[0]);
    state[1] = SubNib(state[1]);

    //行移位,
    state[1] = RotNib(state[1]);

    //轮密钥加
    state[0] = XOR(state[0], keys[4]);
    state[1] = XOR(state[1], keys[5]);


    //格式转换
    String state0 = String.valueOf(state[0]);
    String state1 = String.valueOf(state[1]);
    return state0.concat(state1);
}
```

```java
public String decryption(String mi, String key){
    //获得密钥
    char[][] keys = get_key(key);
    //存储当前状态
    char[][] state = new char[2][8];
    //轮密钥加（异或）
    state[0] = XOR(mi.substring(0,8).toCharArray(), keys[4]);
    state[1] = XOR(mi.substring( beginIndex: 8).toCharArray(), keys[5]);

    //行移位,
    state[1] = RotNib(state[1]);

    //半字节代替,与SubNib操作一样。
    state[0] = I_SubNib(state[0]);
    state[1] = I_SubNib(state[1]);

    //轮密钥加
    state[0] = XOR(state[0], keys[2]);
    state[1] = XOR(state[1], keys[3]);

    //列混淆,使用域内逆矩阵
    state = mix(I_martrix,state);

    //行移位,
    state[1] = RotNib(state[1]);

    //半字节代替。
    state[0] = I_SubNib(state[0]);
    state[1] = I_SubNib(state[1]);

    //轮密钥加
    state[0] = XOR(state[0], keys[0]);
    state[1] = XOR(state[1], keys[1]);

    String state0 = String.valueOf(state[0]);
    String state1 = String.valueOf(state[1]);
    return state0.concat(state1);
}
```

AESClient:这个文件使用 javaFX 引入并展示 html 界面，也实现了在 html 中调用 java 编写的方法。

```java
import javafx.application.Application;
import javafx.concurrent.Worker;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import javafx.stage.Stage;
import netscape.javascript.JSObject;


public class AESClient extends Application {


    public static void main(String[] args) { launch(args); }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("AES加密算法");

        WebView aesWebView = new WebView();
        WebEngine aesEngine = aesWebView.getEngine();
        aesEngine.load(getClass().getResource( name: "aes.html").toExternalForm());
        StackPane aesRoot = new StackPane(aesWebView);
        Scene aesScene = new Scene(aesRoot,  width: 800,  height: 600);

        WebView attackWebView = new WebView();
        WebEngine attackEngine = attackWebView.getEngine();
        attackEngine.load(getClass().getResource( name: "attack.html").toExternalForm());
        StackPane attackRoot = new StackPane(attackWebView);
        Scene attackScene = new Scene(attackRoot,  width: 800,  height: 600);
```

```java
        attackEngine.load(getClass().getResource( name: "attack.html").toExternalForm());
        StackPane attackRoot = new StackPane(attackWebView);
        Scene attackScene = new Scene(attackRoot,  width: 800,  height: 600);

        WebView CBCWebView = new WebView();
        WebEngine CBCEngine = CBCWebView.getEngine();
        CBCEngine.load(getClass().getResource( name: "CBC.html").toExternalForm());
        StackPane CBCRoot = new StackPane(CBCWebView);
        Scene CBCScene = new Scene(CBCRoot,  width: 800,  height: 600);

        primaryStage.setScene(aesScene);
        primaryStage.show();

        AES aes = new AES(primaryStage, aesScene, attackScene, CBCScene);

        aesEngine.getLoadWorker().stateProperty().addListener((observable, oldValue, newValue) -> {
            if (newValue == Worker.State.SUCCEEDED) {
                JSObject window = (JSObject) aesEngine.executeScript("window");
                window.setMember( name: "java", aes);
            }
        });

        attackEngine.getLoadWorker().stateProperty().addListener((observable, oldValue, newValue) -> {
            if (newValue == Worker.State.SUCCEEDED) {
                JSObject window = (JSObject) attackEngine.executeScript("window");
                window.setMember( name: "java", aes);
            }
        });

        CBCEngine.getLoadWorker().stateProperty().addListener((observable, oldValue, newValue) -> {
            if (newValue == Worker.State.SUCCEEDED) {
```

4. 组件封装及接口文档

考虑到模块化和复用性，我们应该封装一些常用的功能。以下是接口文档：

类名：AES
方法：
encryption

参数：
ming (String): 要加密的文本
key (String): 密钥
返回：加密后的文本 (String)

decrypt
参数：
mi (String): 要解密的文本
key (String): 密钥
返回：解密后的文本 (String)

5. 开发建议

请确保用户输入的密钥和文本都经过合法性校验，以避免错误或恶意输入。
保证 UI 的响应速度，特别是加密和解密操作。