S-AES 加密解密系统开发手册

1. 简介

S-AES 加密解密系统是一个为资源受限环境设计的轻量级对称加密解决方案。该系统专注于提供高效且易于实现的加密和解密功能，适用于对小规模数据如消息或二进制数据进行安全处理的场景。S-AES 系统通过简化的算法实现，确保了在性能和安全性之间的平衡，优化了用户的使用体验。

本手册旨在为开发者提供全面的指南，包括系统的组件构成、接口说明、使用示例以及使用过程中的注意事项。通过本手册，开发者可以快速理解和掌握 S-AES 系统的使用方法。

2. 组件构成与接口说明

2.1 S-AES 基本加密解密

SAES_encrypt 类：提供基本的加密功能。

• encryptText(String originalText, String baseKey)：使用 S-AES 算法加密 16 位二进制文本。

参数：

originalText：待加密的二进制文本。

baseKey：加密的初始密钥。

返回值：二进制格式的加密文本。

异常处理：如果输入的原始文本或密钥格式不正确，将抛出 IllegalArgumentException。

• encryptASCII(String originalTextASCII, String baseKey)：使用 S-AES 算法加密 ASCII 字符串。

参数：

originalTextASCII：要加密的 ASCII 文本。

baseKey：加密的初始密钥。

返回值：ASCII 格式的加密文本

异常处理：同上。

SAES_decrypt 类：提供基本的解密功能。

• decryptBinary(String encodedText, String baseKey)：使用简化的 AES 算法解密 16 位二进制文本。

参数：

encodedText：待解密二进制文本。

baseKey：解密的初始密钥。

返回值：二进制格式的解密文本。

异常处理：如果输入的密文或密钥格式不正确，将抛出 IllegalArgumentException。

• decryptAsciiText(String encodedTextASCII, String baseKey)：使用简化的 AES 算法解密 ASCII 字符串。

参数：

encodedTextASCII：要解密的 ASCII 编码的加密文本。

baseKey：解密的初始密钥。

返回值：ASCII 格式的解密文本。

异常处理：同上。

2.2 多重加密解密

• doubleEncrypt(String originalText, String encryptionKey)：使用两个密钥对原始文本进行双重加密。

参数：

originalText：需要加密的原始文本。

encryptionKey：用于加密的密钥。

返回值：加密后的文本。

异常处理：如果输入的原始文本或密钥格式不正确，将抛出 IllegalArgumentException。

• doubleDecrypt(String encodedText, String decryptionKey)：使用两个密钥对加密文本进行双重解密。

参数：

encodedText：需要解密的加密文本。

decryptionKey：用于解密的密钥。

返回值：解密后的原始文本。

异常处理：同上。

•tripleEncrypt(String originalText, String encryptionKey)：使用三个密钥对原始文本进行三重加密。

参数：

originalText：需要加密的原始文本。

encryptionKey：用于加密的密钥。

返回值：加密后的文本。

异常处理：同上。

• tripleDecrypt(String encodedText, String decryptionKey)：使用三个密钥对加密文本进行三重解密。

参数：

encodedText：需要解密的加密文本。

decryptionKey：用于解密的密钥。

返回值：解密后的原始文本。

异常处理：同上。

2.3 CBC 模式加密解密

• CBCModeEncrypt(String plaintext, String initialVector, String key)：使用密码分组链模式加密明文。

参数：

plaintext：待加密的明文。

initialVector：初始向量。

key：加密密钥。

返回值：加密后的密文。

异常处理：如果输入的明文、初始向量或密钥格式不正确，将抛出IllegalArgumentException。

• CBCModeDecrypt(String ciphertext, String initialVector, String key)：使用密码分组链模式解密密文。

参数：

ciphertext：待解密的密文。

initialVector：初始向量。

key：解密密钥。

返回值：解密后的明文。

异常处理：同上。

3. 使用示例

3.1 S-AES 基本加密解密

密钥生成

```
public static void main(String[] args) {
    String baseKey = "3777";    // 初始 4 位十六进制密钥
    String[] derivedKeys = deriveKeys(baseKey);
    // 打印扩展后的密钥（二进制形式）
    for (int i = 0; i < derivedKeys.length; i++) {
        System.out.println("w" + i + ": " + derivedKeys[i]);
    }
}
```

加密示例

```
public static void main(String[] args) {
        String baseKey = "3777";    // 初始 4 位十六进制密钥

        String originalText = "0110101110100011";
        String encodedText = encryptText(originalText, baseKey);
        System.out.println("加密结果如下：");
        System.out.println(encodedText); // 输出密文结果

        String originalTextASCII = "wpk in software engineer and this is a test txt file"; // 输入 String 类型的 ASCII
        String encodedTextASCII = encryptASCII(originalTextASCII, baseKey);
        System.out.println("加密结果如下：");
        System.out.println(encodedTextASCII); // 输出 ASCII 格式的加密文本
}
```

解密示例

```
public static void main(String[] args) throws IOException {
        String baseKey = "3777";    // 初始 4 位十六进制密钥
        String ciphertext = "1011100011100001";
        String decryptedText = decryptBinary(ciphertext, baseKey);
        System.out.println("解密结果如下：");
```

System.out.println(decryptedText); // 输出解密结果

String encodedTextASCII = " ó ó TäTä0B´¶ ¡"; // ASCII 编码的加密文本

String recoveredTextASCII = decryptAsciiText(encodedTextASCII, baseKey);

System.out.println("解密结果如下：");

System.out.println(recoveredTextASCII); // 输出解密结果

}

3.2 多重加密解密

双重加解密示例

```
String key = "37773777";
String originalText = "0000000000000011";
String encodedText = "0011110011100000";
// 调用 doubleEncrypt 方法
String doubleencodedText = doubleEncrypt(originalText, key);
System.out.println("双重加密结果如下：");
System.out.println(doubleencodedText);  // 输出双重加密的密文
// 调用 doubleDecrypt 方法
String doubleDecryptedText = doubleDecrypt(encodedText, key);
System.out.println("双重解密结果如下：");
System.out.println(doubleDecryptedText);  // 输出双重解密的明文
```

三重加解密示例

```
String originalText = "0000000000000011";

String encodedText = "1100001111011100";

String keyt = "2D55A1F3B276";

String tripleCiphertext = "1010101001111000";

//  调用 tripleEncrypt 方法

String tripleEncryptedText = tripleEncrypt(originalText, keyt);

System.out.println("三重加密结果如下：");

System.out.println(tripleEncryptedText);   //  输出三重加密的密文

//  调用 tripleDecrypt 方法

String tripleRecoveredText = tripleDecrypt(tripleCiphertext, keyt);

System.out.println("三重解密结果如下：");

System.out.println(tripleRecoveredText);   //  输出三重解密的明文
```

3.3 CBC 模式加密解密

加解密示例

```
public static void main(String[] args) {

    String plaintext =

"00000000111111111111111100000000111111111111111100000000000001010";

    String initialVector = "1010101010101011";   // 16 位初始向量
```

```
        String key = "3777";    // 16 位密钥

        String encryptedTextforDecrypt =
"0100000110100001001100000010000110001011111100100110101001
1010111";            String decryptedText =
CBCModeDecrypt(encryptedTextforDecrypt, initialVector, key);
        System.out.println("解密结果如下：");
        System.out.println(decryptedText);    //  输出解密结果

        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
}
```

4. **注意**：使用时请确保输入格式正确，以实现加解密的正确运行。