

开发手册&用户指南

—— written by Superstars

1. 简介

1.1 系统概述

该系统是我们“Superstars”团队为用户设计的一款简单、开源的 S-AES 算法加解密系统，能够 满足用户简单的加解密需要（二进制/ ASCII 码字符串加解密、二重加密、三重加密、CBC 模式加密）。

1.2 主要功能和特点

该系统主要功能包括对二进制数/ ASCII 码进行 S-AES 算法的加解密。该系统的特点在于简单易懂、容易上手，可以帮助不太了解 S-AES 算法的用户快速获取加解密结果以及相应的密钥信息，也可以帮助正在学习 S-AES 算法学生更直观的接触到简单的 DES 应用，验证 S-AES 的雪崩效应特性，感受 DES 加密算法的伟大和奇妙之处。

1.3 目标用户群体

初步学习和希望了解 S-AES 算法的用户。

1.4 使用说明

该系统可以在 web 上运行，具体页面及功能详见下文

2. 开发者团队

2.1 开发者队名：Superstars

2.2 开发者姓名：

张家玮、季俊杰

3. 开发环境

3.1 开发工具和技术栈

3.1.1 开发工具

集成开发环境：Pycharm

版本控制：Github

3.1.2 技术栈

前端技术：语言：Html、Css、JavaScript

后端技术：语言：Python

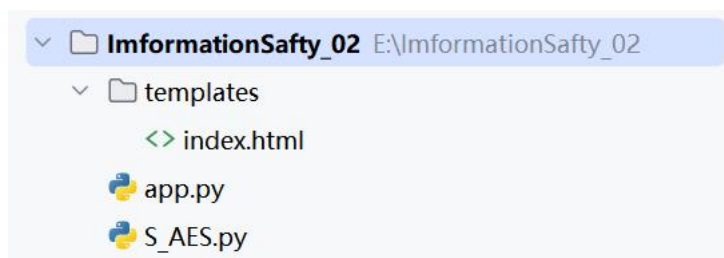
3.2 运行

运行 app.py 文件即可

4. 代码结构

4.1 项目的目录结构

项目的目录结构图如下：



S_AES.py 实现一个基于 S-AES（数据加密标准）的加密和解密算法，包括生成密钥、处理明文和密文的各种变换操作。

Templates\index.html 负责创建应用程序的主窗口，设置界面布局。

App.py 处理用户输入和与加密算法的交互。

4.2 关键组件和模块的描述

4.2.1 字节替换

```
def NS(self, input):
    output = [0, 0]
    for i in range(len(input)):
        left = input[i] >> 4
        right = input[i] & 0xF
        left_i = left >> 2
        left_j = left & 0x3
        right_i = right >> 2
        right_j = right & 0x3
        sum_value = (self.sBox[left_i][left_j] << 4) + self.sBox[right_i][right_j]
        output[i] = sum_value
    return output
```

2 个用法

```
def NS_reverse(self, input):
    output = [0, 0]
    for i in range(len(input)):
        left = input[i] >> 4
        right = input[i] & 0xF
        left_i = left >> 2
        left_j = left & 0x3
        right_i = right >> 2
        right_j = right & 0x3
        sum_value = (self.sBox_reverse[left_i][left_j] << 4) + self.sBox_reverse[right_i][right_j]
        output[i] = sum_value
    return output
```

4.2.2 行移位

```
def SR(self, input):
    output = [0, 0]
    output[0] = input[0]
    left = input[1] >> 4
    right = input[1] & 0xF
    output[1] = (right << 4) + left
    return output
```

4.2.3 列混淆

```

def MC(self, input):
    output = [0, 0]
    kernel = [[1, 4], [4, 1]]
    grid = [[0, 0], [0, 0]]
    # 将输入的每一个 int 拆分为高 4 位和低 4 位
    for i in range(2):
        left = input[i] >> 4
        right = input[i] % 16
        grid[i][0] = left
        grid[i][1] = right

    temp = [[0, 0], [0, 0]]
    # 进行矩阵运算
    for i in range(2):
        for j in range(2):
            temp[i][j] = (self.mulitiKernel[kernel[i][0]][grid[0][j]] ^ self.mulitiKernel[kernel[i][1]][grid[1][j]])
    # 将结果重新组合为输出
    for i in range(2):
        output[i] = (temp[i][0] << 4) + temp[i][1]

    return output

def MC_reverse(self, input):
    output = [0, 0]
    kernel = [[9, 2], [2, 9]]
    grid = [[0, 0], [0, 0]]
    # 将输入的每一个 int 拆分为高 4 位和低 4 位
    for i in range(2):
        left = input[i] >> 4
        right = input[i] % 16
        grid[i][0] = left
        grid[i][1] = right

    temp = [[0, 0], [0, 0]]
    # 进行矩阵运算
    for i in range(2):
        for j in range(2):
            temp[i][j] = (self.mulitiKernel[kernel[i][0]][grid[0][j]] ^ self.mulitiKernel[kernel[i][1]][grid[1][j]])
    # 将结果重新组合为输出
    for i in range(2):
        output[i] = (temp[i][0] << 4) + temp[i][1]

    return output

```

4. 2. 4 轮密钥加

```

def A_k(self, plianText, key):
    output = [0, 0]
    for i in range(2):
        output[i] = plianText[i] ^ key[i]
    return output

```

4. 2. 5 密钥扩展

```
def subKey(self,originKey):
    output = [originKey.copy()] # Initialize the output with the original key
    for i in range(1, 3):
        word = self.binary2Int(output[i - 1]) # Convert the previous subkey to an integer
        word_left = word >> 8 # Get the left 8 bits
        word_right = word & 0xFF # Get the right 8 bits
        wordRightAfterG = self.binary2Int(self.g(word_right, i)) # Apply function g to the right part
        newWordLeft = word_left ^ wordRightAfterG # XOR with the result of g
        newWordRight = newWordLeft ^ word_right # XOR to get the new right part
        newWord = (newWordLeft << 8) + newWordRight # Combine the two parts
        output.append(self.int2Binary(newWord)) # Convert back to binary and append to output
    return output
```

5. 基础功能及界面

5.1 二进制/字符串加密页面

5.1.1 功能

允许用户输入二进制数据或字符串进行加密。

5.1.2 页面展示

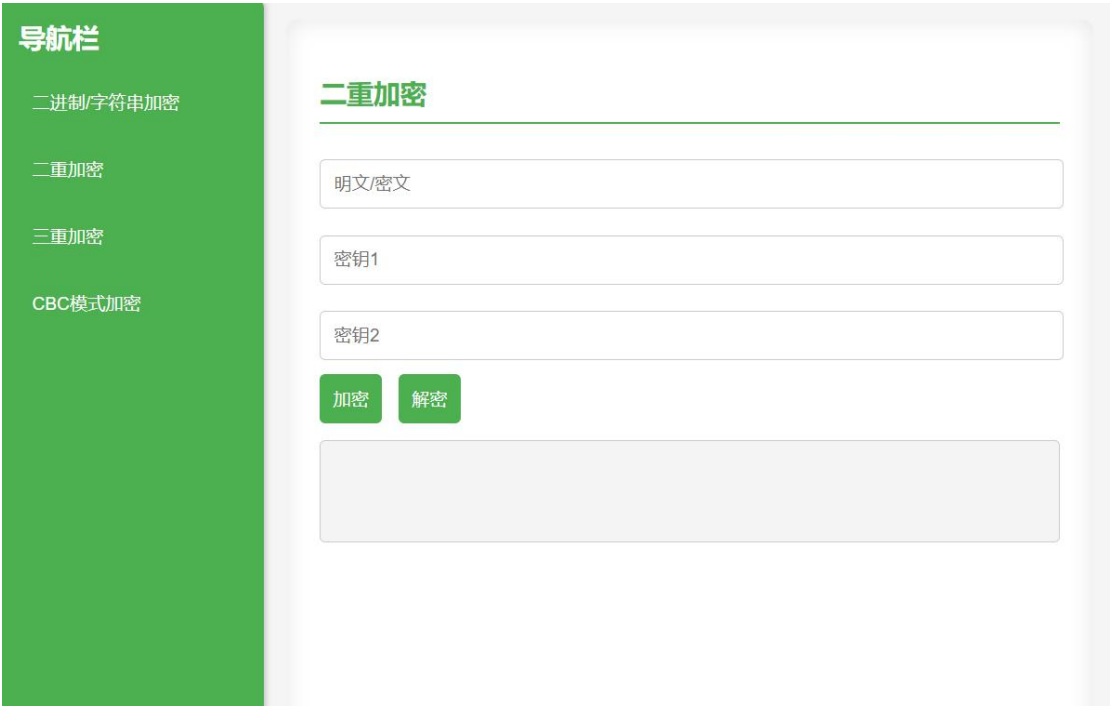
The screenshot shows a web application interface for binary/string encryption. On the left is a green sidebar with a '导航栏' (Navigation Bar) containing four items: '二进制/字符串加密' (Binary/String Encryption), '二重加密' (Double Encryption), '三重加密' (Triple Encryption), and 'CBC模式加密' (CBC Mode Encryption). The '二进制/字符串加密' item is highlighted. The main content area has a title '二进制/字符串加密' and two input fields: '明文/密文' (Plaintext/Ciphertext) and '密钥' (Key). Below these are two buttons: '加密' (Encrypt) and '解密' (Decrypt). At the bottom of the main area is a large, empty light gray box for the output.

5.2 二重加密页面

5.2.1 功能

提供二重加密功能，允许用户进行两次加密操作。

5.2.2 页面展示

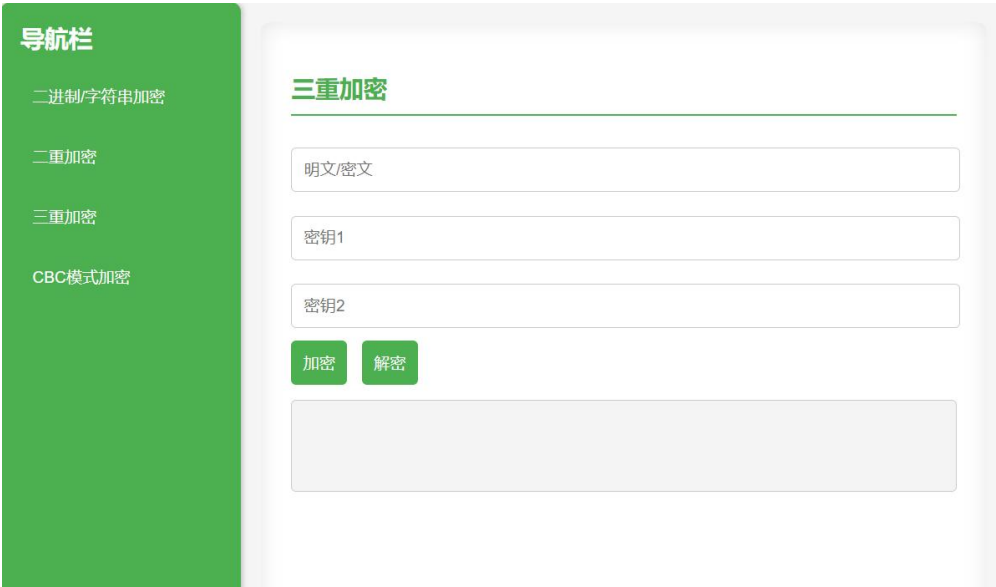


5.3 三重加密页面

5.3.1 功能

提供三重加密功能，用户进行三次加密操作，但不需要第三个密钥。

5.3.2 页面展示

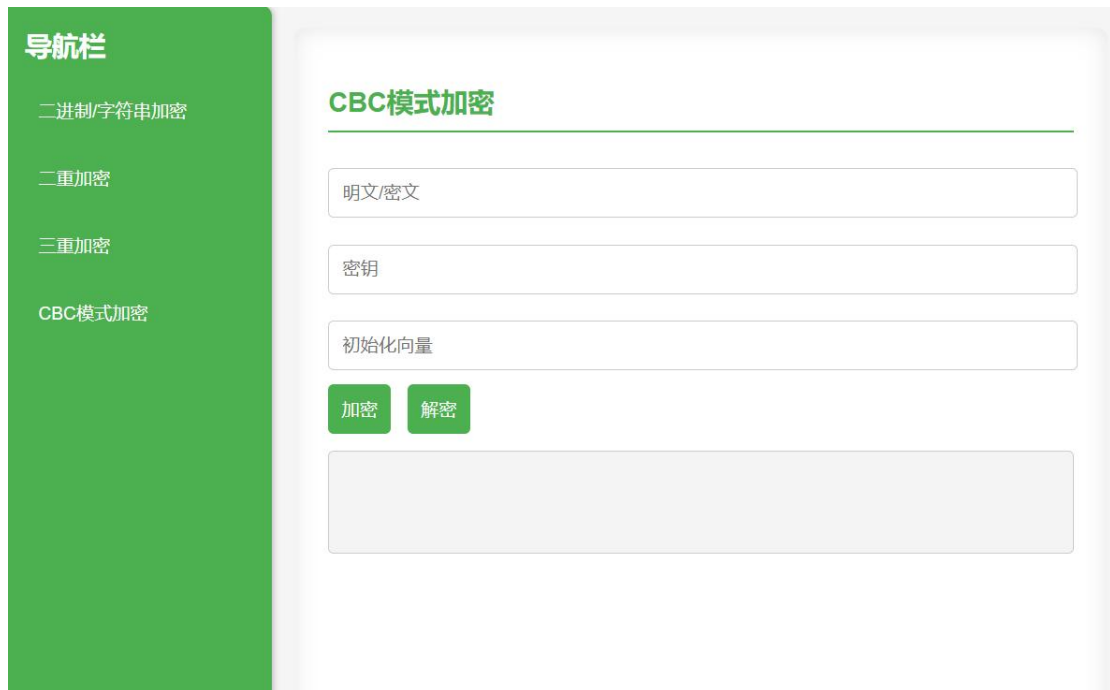


5.4 CBC 模式加密页面

5.4.1 功能

实现加密块链接模式（CBC），允许用户输入数据进行加密。

5.4.2 页面展示



The screenshot displays a web application interface for CBC mode encryption. On the left, a green sidebar contains a '导航栏' (Navigation Bar) with four menu items: '二进制/字符串加密' (Binary/String Encryption), '二重加密' (Double Encryption), '三重加密' (Triple Encryption), and 'CBC模式加密' (CBC Mode Encryption). The main content area is titled 'CBC模式加密' and features three input fields: '明文/密文' (Plaintext/Ciphertext), '密钥' (Key), and '初始化向量' (Initialization Vector). Below these fields are two green buttons labeled '加密' (Encrypt) and '解密' (Decrypt). A large, empty light-gray rectangular box is positioned at the bottom of the main area, likely for displaying the output of the encryption or decryption process.