S-AES 加密解密系统

1.用户指南

1.1 基础加解密（第 1 关、第 3 关）

1. 功能：16 位数据块的 S-AES 标准加解密，支持二进制和 ASCII 文本输入
2. 操作步骤：

选择"基础加解密"标签页

选择输入类型：16 位二进制或 ASCII 文本
输入 16 位密钥（二进制格式）
输入要加密/解密的数据
点击"加密"或"解密"按钮
示例图片：

1.2 多重加密（第 4 关）

1. 双重加密：
使用两个 16 位密钥（K1+K2），总密钥长度 32 位
加密流程：E(K2, E(K1, P))
2. 三重加密：
使用三个 16 位密钥（K1+K2+K3），总密钥长度 48 位
加密流程：E(K3, D(K2, E(K1, P)))
3. 操作步骤：
选择"多重加密"标签页
选择加密模式：双重或三重加密
输入对应的密钥
输入要加密/解密的数据
点击相应按钮执行操作
示例图片：

1.3 CBC 模式（第 5 关）

1. 功能：密码分组链模式，支持错误传播分析
2. 操作步骤：
选择"CBC 模式"标签页
输入 16 位密钥和初始向量 IV
输入明文文本

点击"CBC 加密"进行加密
可进行篡改实验，观察错误传播效果
示例图片：

1.4 密码分析（第 4 关）
1. 中间相遇攻击：
针对双重加密的密钥恢复攻击
需要至少一对明密文
2. 操作步骤：
选择"密码分析"标签页
输入已知的明密文对
点击"执行攻击"开始分析
查看可能的密钥对结果
示例图片：

中间相遇攻击

攻击原理： 针对双重加密 E(K2, E(K1, P))，通过构建加密表和解密表

第一对明密文：

Hello

%SÑÇ i

第二对明密文（可选，提高准确性）：

第二明文

第二密文

执行攻击 清空

攻击结果：

找到 1 个可能的密钥对：

密钥对 1：
K1: 1010111100110001 (AF31H)
K2: 1100110011001100 (CCCCH)
完整密钥: 10101111001100011100110011001100

验证第一个密钥对：
明文: Hello
预期密文: %SÑÇ□i
实际密文: %SÑÇ□i
匹配: 是

2. 开发手册

2.1 系统架构

项目结构：

```
├──── app.py                 # Flask 后端主程序
├──── S_AES.py               # 基础 S-AES 算法实现
├──── S_AES_3.py             # 扩展功能实现（GUI 版本）
├──── work.py                # CBC 模式实现
├──── templates/
│     └──── index.html       # 前端界面
└──── 说明文档.md            # 项目文档
```

2.2 核心组件接口

2.2.1 SAES 类（S_AES.py）

构造函数：

```
def __init__(self)
```

初始化 S-AES 算法，包括 S 盒、逆 S 盒、轮常数等。

核心方法：

1. 密钥扩展：

2. def key_expansion(self, key)

输入：16 位整数密钥

输出：三个轮密钥(k0, k1, k2)

2.数据块加密：

```
def encrypt_block(self, plaintext, key)
```

输入：16 位明文，16 位密钥

输出：16 位密文

3.数据块解密：

```
def decrypt_block(self, ciphertext, key)
```

输入：16 位密文，16 位密钥

输出：16 位明文

4.文本加解密：

```
def encrypt_text(self, plaintext, key)
def decrypt_text(self, ciphertext, key)
```

输入：字符串明文/密文，16 位密钥

输出：字符串密文/明文

2.2.2 扩展功能接口

1.多重加密：

```
def double_encrypt_block(self, plaintext, key1, key2)
def triple_encrypt_block(self, plaintext, key1, key2, key3)
```

2.CBC 模式：

```
def cbc_encrypt(self, plaintext_blocks, key, iv)
def cbc_decrypt(self, ciphertext_blocks, key, iv)
```

3.：中间相遇攻击

```
def attack(self, plaintexts, ciphertexts, max_keys=100)
```

2.3 API 接口

2.3.1 基础加解密 API

1.加密

端点：POST /api/basic/encrypt

参数：

json

```
{
  "key": "16 位二进制密钥",
  "input": "输入数据",
  "type": "binary|text"
}
```

响应：

json

```
{
  "success": true,
  "result": {
    "binary": "二进制密文",
    "hex": "十六进制密文",
    "text": "文本密文"
  }
}
```

2.解密

端点：POST /api/basic/decrypt

参数同上；响应类似

2.3.2 多重加密 API

1. 端点：POST /api/multiple/encrypt 和 /api/multiple/decrypt

2. 参数：

json

```
{
  "mode": "double|triple",
  "keys": ["key1", "key2", ...],
  "input": "输入数据",
  "type": "binary|text"
}
```

2.3.3 CBC 模式 API

1.端点：POST /api/cbc/encrypt 和 /api/cbc/decrypt

2.参数：

json

```
{
  "key": "16 位密钥",
  "iv": "16 位初始向量",
  "input": "明文文本"
}
```

2.3.4 中间相遇攻击 API

1.端点：POST /api/attack/meet-middle
2.参数：
json
```
{
  "plaintexts": ["明文 1", "明文 2"],
  "ciphertexts": ["密文 1", "密文 2"]
}
```

3. 总结

本系统成功实现了 S-AES 算法的所有要求功能。系统设计合理，代码结构清晰，测试充分，具有良好的可用性。