# S-AES 算法实现

## 项目简介

基于配置文件的优化版`S-AES`实现，采用模块化架构，提高代码的可维护性和可配置性。

## 项目结构

```
s-aes-project/
├── config/
│   ├── __init__.py
│   ├── algorithm_config.py     # 算法相关配置
│   ├── gui_config.py           # GUI相关配置
│   └── constants.py            # 通用常量
├── core/
│   ├── __init__.py
│   └── s_aes_core.py           # 核心算法
├── ui/
│   ├── __init__.py
│   └── s_aes_gui.py            # GUI界面
├── utils/
│   ├── __init__.py
│   └── helpers.py              # 工具函数
├── main.py                     # 主程序入口
├── requirements.txt
└── README.md
```

## 配置优化特点

### 1. 模块化架构

- **配置分离**: 算法参数、界面设置、通用常量分别管理
- **职责清晰**: 每个模块专注于特定功能
- **易于维护**: 修改配置无需改动核心代码

### 2. 可配置性

- **算法参数**: S盒、轮常数、矩阵参数等可配置
- **界面设置**: 窗口大小、标签文本、默认值等可配置
- **错误处理**: 错误消息集中管理，支持多语言扩展

### 3. 代码复用

- **工具函数**: 通用功能封装，避免代码重复
- **工厂方法**: GUI组件创建标准化
- **验证函数**: 输入验证逻辑统一处理

# 运行方法

```
# 运行主程序
python main.py
```

# 配置修改示例

## 修改算法参数

在`config/algorithm_config.py`中:

### 1. 修改S盒

```
S_BOX = [

自定义S盒配置...

]
```

### 2. 修改轮数

```
ROUNDS = 4
```

## 修改界面文本

在`config/gui_config.py`中:

### 1. 修改窗口标题

```
WINDOW_TITLE = "我的S-AES加密工具"
```

### 2. 修改按钮文本

```
BUTTON_TEXTS = {
    "basic_encrypt": "开始加密",

... 其他按钮文本

}
```

# 扩展开发

## 1. 添加新功能

- 在对应配置类中添加配置参数
- 在核心算法或GUI中实现功能
- 通过配置文件控制功能行为

## 2. 添加新语言支持

- 在 `gui_config.py` 中添加多语言字典
- 根据语言设置返回对应文本
- 在GUI初始化时设置语言

# 技术优势

- **易于调试**: 配置集中管理，问题定位快速
- **便于测试**: 可以轻松创建不同配置的测试环境
- **可扩展性强**: 新功能通过配置即可添加
- **团队协作友好**: 配置与代码分离，减少冲突

# 许可证

本项目仅用于教学目的。

# 配置优化带来的优势

## 1. 可维护性提升

- **配置集中管理**: 所有参数在 `config` 目录中统一管理
- **修改便捷**: 调整算法参数或界面设置无需修改核心代码
- **错误定位**: 问题可以快速定位到具体配置项

## 2. 可扩展性增强

- **模块化设计**: 新功能可以独立添加到相应模块
- **配置驱动**: 通过修改配置即可改变程序行为
- **接口清晰**: 各模块间通过明确定义的接口交互

## 3. 代码质量改善

- **减少重复**: 通用功能封装在工具类中
- **类型安全**: 配置项有明确的类型和取值范围
- **文档完善**: 每个配置项都有详细的注释说明

## 4. 团队协作优化

- **配置与代码分离**: 减少版本冲突
- **标准化接口**: 不同开发者可以并行工作
- **易于理解**: 新成员可以快速理解项目结构