# S-DES 加密解密工具用户文档

## 1. 软件概述

S-DES（简化数据加密标准）工具是基于 Python 实现的图形化应用，完全遵循 S-DES 算法规范，支持两类核心操作：

* 基础功能：8 位二进制数据、ASCII 文本的加密与解密
* 高级功能：暴力破解（需明文 - 密文对）、封闭性测试（检测密钥碰撞）
* 定位：仅用于教学演示，因密钥空间仅 1024 种（10 位二进制密钥），**不可用于实际安全加密场景**，实际应用需选择 AES 等现代加密算法。

## 2. 系统要求

### 2.1 操作系统

* Windows：Windows 10/11（32 位 / 64 位）
* macOS：macOS 10.14 及以上版本
* Linux：Ubuntu 16.04+、CentOS 7 + 等主流发行版

### 2.2 软件依赖

* Python 版本：3.7 及以上（建议 3.8-3.11，兼容性最佳）
* 必备库：tkinter（图形化界面依赖，通常随 Python 标准库预装）

## 3. 安装与启动

### 3.1 前期准备：检查 Python 环境

1. 打开终端（Windows：CMD/PowerShell；macOS/Linux：终端）
2. 输入命令检查 Python 版本：python --version 或 python3 --version
   * 若显示 “Python 3.7.x” 及以上，直接进入下一步；
   * 若提示 “未找到命令”，需先从[Python 官网](https://www.python.org/)下载并安装对应版本（安装时勾选 “Add Python to PATH”）。

### 3.2 下载程序

从指定 GitHub 仓库下载完整程序文件：ex1-S-DES.py（确保文件未损坏，大小约几 KB）。

### 3.3 启动程序

1. 打开终端，切换到程序所在文件夹（例：cd D:\S-DES-Tool）
2. 输入启动命令：python ex1-S-DES.py 或 python3 ex1-S-DES.py
3. 成功启动后，将自动弹出图形化操作界面。

## 4. 界面功能详解

界面布局简洁直观，核心元素按功能分区排列，各组件作用如下：

|  |  |  |
| --- | --- | --- |
| 组件名称 | 位置 | 功能描述 |
| 密钥输入框 | 界面顶部 | 输入 10 位二进制密钥，右侧有 “密钥格式提示”（仅允许 0/1） |
| 明文 / 密文输入框 | 界面中部 | 输入待处理数据（8 位二进制 / ASCII 文本 / 8 的倍数二进制字符串），支持复制粘贴 |
| 功能按钮组 | 输入框下方 | 4 个核心按钮：【加密】【解密】【暴力破解】【封闭性测试】 |
| 输出区域 | 界面下部 | 显示操作结果（如密文、明文、破解到的密钥）、状态提示（如 “加密成功”）及日志 |
| 进度条 | 输出区域上方 | 仅在 “暴力破解” 时显示，实时展示破解进度（0%-100%） |

## 5. 基本操作指南

### 5.1 加密操作（分两种数据类型）

#### 5.1.1 加密 8 位二进制数据

1. 密钥输入：在 “密钥输入框” 中输入 10 位二进制（例：1010000010）
2. 明文输入：在 “明文 / 密文输入框” 中输入 8 位二进制（例：00000000）
3. 执行加密：点击【加密】按钮
4. 查看结果：输出区域将显示 “加密成功” 及对应的 8 位二进制密文（例：10001011）

#### 5.1.2 加密 ASCII 文本

1. 密钥输入：同 5.1.1（例：1010000010）
2. 明文输入：输入任意 ASCII 可打印字符（例：hello“AB”）
3. 执行加密：点击【加密】按钮
4. 查看结果：输出区域显示 “加密成功” 及完整二进制和ASCII密文（长度为 8 的倍数，例：0110000101100010对应 “AB”）

### 5.2 解密操作（分两种数据类型）

#### 5.2.1 解密 8 位二进制密文

1. 密钥输入：输入与加密时完全相同的 10 位二进制密钥（例：1010000010）
2. 密文输入：在 “明文 / 密文输入框” 中输入 8 位二进制密文（例：10001011）
3. 执行解密：点击【解密】按钮
4. 查看结果：输出区域显示 “解密成功” 及对应的 8 位二进制明文（例：00000000）

#### 5.2.2 解密二进制密文字符串

1. 密钥输入：同加密密钥（例：1010000010）
2. 密文输入：输入长度为 8 的倍数的二进制字符串（例：0110000101100010）
3. 执行解密：点击【解密】按钮
4. 查看结果：输出区域显示 “解密成功” 及对应的 ASCII 文本（例：AB）

## 6. 高级功能使用

### 6.1 暴力破解（需已知明文 - 密文对）

#### 功能原理

通过遍历所有 1024 种可能的 10 位二进制密钥，找到能将 “已知明文” 加密为 “已知密文” 的匹配密钥。

#### 操作步骤

1. 准备明文 - 密文对：
   * 先用任意密钥加密某明文（例：明文00000000，密钥1010000010，得到密文10001011）
2. 输入数据：
   * 保持 “明文 / 密文输入框” 为已知明文（例：00000000）
   * 记住对应的已知密文（无需输入，程序自动对比）
3. 启动破解：点击【暴力破解】按钮
4. 查看结果：
   * 进度条实时更新，破解完成后，输出区域显示：
     + 匹配的密钥列表（例：[1010000010]）
     + 总耗时（通常 1-5 秒，视设备性能而定）
     + 尝试的密钥总数（固定 1024 种）

### 6.2 封闭性测试（检测密钥碰撞）

#### 功能原理

密钥碰撞指 “不同密钥对同一明文加密，得到相同密文”。本功能通过随机生成明文和密钥，测试 S-DES 算法是否存在此类碰撞。

#### 操作步骤

1. 无需输入数据：无需手动输入密钥或明文
2. 启动测试：点击【封闭性测试】按钮
3. 查看结果：
   * 程序自动执行：生成随机明文→生成随机密钥→加密得到密文→遍历其他密钥验证是否碰撞
   * 输出区域显示：测试状态（如 “正在验证密钥碰撞”）、最终结论（如 “未检测到密钥碰撞，S-DES 封闭性良好”）、测试用的明文 / 密钥 / 密文详情

## 7. 输入格式规范

### 7.1 密钥格式（强制要求）

|  |  |  |  |
| --- | --- | --- | --- |
| 要求项 | 具体规则 | 正确示例 | 错误示例 |
| 长度 | 精确 10 位 | 1010000010 | 101000001（9 位） |
| 字符 | 仅允许数字 “0” 和 “1” | 0011010101 | 1234567890（含数字 2-9） |
| 空格 / 符号 | 不允许包含空格、逗号等任何非 0/1 字符 | 1100110011 | 1100 110011（含空格） |

### 7.2 数据格式（分三种类型）

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 适用场景 | 格式要求 | 正确示例 |
| 8 位二进制 | 单条短数据加密 / 解密 | 精确 8 位，仅 0/1 | 00000000 11110000 |
| ASCII 文本 | 多字符文本加密 / 解密 | 任意长度的 ASCII 可打印字符（不含控制字符） | hello S-DES 123 |
| 二进制字符串 | 长数据加密 / 解密 | 长度为 8 的倍数，仅 0/1 | 0000000011110000（16 位） |

## 8. 常见问题与解决方案

### Q1：启动程序时报错 “ModuleNotFoundError: No module named 'tkinter'”

#### 原因

tkinter 库未安装（Python 标准库默认预装，但部分 Linux 系统可能缺失）。

#### 解决方案

* Windows/macOS：重新安装 Python（勾选 “Add Python to PATH”，默认包含 tkinter）
* Ubuntu/Debian：终端输入sudo apt-get install python3-tk
* CentOS/RHEL：终端输入sudo yum install python3-tkinter
* 验证：安装后，终端输入python -m tkinter，若弹出空白 tkinter 窗口，说明安装成功。

### Q2：加密 ASCII 文本时，输出乱码或提示 “无效字符”

#### 原因

输入了非 ASCII 字符（如中文、日文、特殊符号 “±≠”）或 ASCII 控制字符（如换行符、制表符）。

#### 解决方案

* 仅输入 ASCII 可打印字符（范围：十进制 32-126，含字母、数字、常见符号!@#$%等）
* 示例：正确输入hello123!，错误输入你好 hello\n（含换行符）。

### Q3：点击【加密 / 解密】按钮无反应

#### 可能原因及解决

1. 密钥格式错误（非 10 位 0/1）：检查密钥输入框，确保符合 7.1 规范
2. 数据格式错误（如 8 位二进制输入 7 位）：按 7.2 规范修正输入数据
3. 程序卡顿：关闭其他占用资源的软件，重新启动 S-DES 工具

### Q4：暴力破解耗时超过 10 秒

#### 原因

设备性能较低（如老旧电脑、同时运行多程序），或后台进程占用 CPU 资源。

#### 建议

1. 关闭浏览器、视频软件等非必要程序
2. 确保 Python 版本为 3.8 及以上（新版本性能优化更好）
3. 教学演示时，优先使用短明文（如 8 位二进制），避免 ASCII 长文本。

### Q5：不同设备 / 小组的加密结果不一致

#### 可能原因

1. S-DES 算法实现差异（如置换表 P-Box、S-Box 不一致）
2. 密钥扩展算法错误（如轮密钥生成步骤偏差）
3. Feistel 网络轮函数实现错误（如 F 函数计算逻辑不同）

#### 解决方案

1. 核对算法实现：确保置换表、轮函数逻辑与标准 S-DES 完全一致（参考课程提供的算法文档）
2. 用 9.1 节 “基本测试用例” 验证：若输入相同密钥和明文，输出不同密文，需检查代码逻辑。

## 9. 测试用例示例

### 9.1 基本加密 - 解密测试（验证正确性）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试类型 | 密钥 | 明文（二进制） | 预期密文（二进制） | 解密后明文（二进制） |
| 加密 | 1010000010 | 00000000 | 10001011 | - |
| 解密 | 1010000010 | 10001011 | - | 00000000 |

#### 操作步骤

1. 按 “加密 8 位二进制数据” 操作，确认密文为10001011
2. 按 “解密 8 位二进制密文” 操作，确认解密后明文为00000000，说明工具正常。

### 9.2 ASCII 文本加密测试

|  |  |  |  |
| --- | --- | --- | --- |
| 密钥 | 明文（ASCII） | 预期密文（二进制，前 16 位） | 解密后明文 |
| 1010000010 | AB | 0110000101100010 | AB |
| 0011010101 | hello | 011010000110010101101100 | hello |

#### 验证方法

1. 加密 ASCII 文本后，记录二进制密文
2. 用相同密钥解密该密文，确认输出与原明文一致。

## 10. 技术支持

若遇到以下问题，可通过以下渠道获取帮助：

1. 优先自查：
   * 查看终端输出的错误日志（程序启动时的终端窗口，报错信息通常在此显示）
   * 核对 “输入格式规范”（7 章）和 “常见问题”（8 章）
2. 外部支持：
   * 课程答疑：联系授课老师或助教，提供错误截图和操作步骤
   * GitHub Issue：在程序所在 GitHub 仓库提交 Issue，描述问题（含设备系统、Python 版本、错误信息）
   * 技术交流：加入课程指定的技术交流群，寻求同学或老师帮助