\*\*1. 简明的介绍和概述：\*\*

欢迎使用S-DES加密软件用户指南。本软件是一个用于对文本进行加密和解密的工具，基于简化数据加密标准（S-DES）算法。它可以帮助您保护敏感信息，确保您的数据在传输和存储时得到安全保护。

\*\*2. 系统要求和安装：\*\*

\*\*系统要求：\*\*

- 操作系统：支持Windows、macOS和Linux。

- Python：需要安装Python解释器。

- Tkinter：需要安装Tkinter库。

- pyperclip：需要安装pyperclip库。

\*\*安装步骤：\*\*

1. 下载软件源代码到您的计算机。

2. 安装Python解释器（如果尚未安装）。

3. 安装Tkinter库（如果尚未安装）。

4. 安装pyperclip库（如果尚未安装）。

5. 运行源代码。

\*\*3. 用户界面和基本操作：\*\*

\*\*用户界面\*\*

用户界面简单易用，包括功能选择界面和操作界面。

功能选择包括二进制数加解密GUI按钮、ASCLL码加解密GUI按钮以及中文加解密GUI按钮。

操作界面包括随机生成密钥按钮、文本输入框、密钥输入框、加密和解密按钮，以及结果显示区域，其中二进制数加解密GUI还有测试按钮。

\*\*基本操作\*\*

- 在功能选择界面选择二进制数加解密GUI、ASCLL码加解密GUI或是中文加解密GUI以进入操作界面。

- 点击随机生成密钥按钮生成随机密钥。（可选）

- 在文本输入框中输入要加密或解密的文本。

- 在密钥输入框中输入10位的二进制密钥。

- 点击加密按钮执行加密操作。

- 点击解密按钮执行解密操作。

- 结果将显示在结果区域。

\*\*4. 加密和解密流程：\*\*

加密和解密流程如下：

- 输入明文文本或密文文本。

- 输入10位的二进制密钥。

- 点击加密按钮以加密文本，或点击解密按钮以解密文本。

算法描述：

加密算法：C=IP^{-1}(f\_{k\_{2}}(SW(f\_{k\_{1}}(IP(P)))))

解密算法：P=IP^{-1}(f\_{k\_{1}}(SW(f\_{k\_{2}}(IP(C)))))

密钥扩展：k\_{i}=P\_{8}(Shift^{i}(P\_{10}(K))), (i=1,2)

转换装置设定：

密钥扩展置

* P\_{10}=(3,5,2,7,4,10,1,9,8,6)
* P\_{8}=(6,3,7,4,8,5,10,9)
* Left\_Shift^1=(1,1)

初始置换盒

* IP=(2,6,3,1,4,8,5,7)

最终置换盒

* IP^{-1}=(4,1,3,5,7,2,8,6)

轮函数F

* EPBox=(4,1,2,3,2,3,4,1)
* SBox\_{1}=[(1,0,3,2);(3,2,1,0);(0,2,1,3);(3,1,0,2)]
* SBox\_{2}=[(0,1,2,3);(2,3,1,0);(3,0,1,2);(2,1,0,3)]
* SPBox=(2,4,3,1)

部分代码：

def initial\_permutation(data):  
 permuted\_data = [data[ip - 1] for ip in IP]  
 return permuted\_data  
  
# 对8位数据进行逆初始置换（IP^-1）  
def inverse\_initial\_permutation(data):  
 permuted\_data = [data[ip - 1] for ip in IP\_INV]  
 return permuted\_data  
  
# 将4位数据扩展为8位数据（使用扩展表E）  
def expand(data):  
 expanded\_data = [data[e - 1] for e in E]  
 return expanded\_data  
  
# 对两个8位数据块进行异或运算  
def xor(data1, data2):  
 result = []  
 for i in range(len(data1)):  
 result.append(data1[i] ^ data2[i])  
 return result  
  
# 将一个8位数据块分成两个4位数据块  
def split(data):  
 split\_index = len(data) // 2  
 first\_half = data[:split\_index]  
 second\_half = data[split\_index:]  
  
 return first\_half, second\_half  
  
# 使用S盒对4位数据块进行替代  
def s\_box(data, s\_box\_table):  
 row = int(''.join(map(str, [data[0], data[3]])), 2)  
 col = int(''.join(map(str, data[1:3])), 2)  
 return [int(x) for x in format(s\_box\_table[row][col], '02b')]  
  
# 使用给定的置换表对4位数据块进行置换  
def permute(data, perm\_table):  
 return [data[p - 1] for p in perm\_table]  
  
# 从10位密钥生成轮密钥  
def generate\_round\_keys(key):  
 key = permute(key, P10)  
 left, right = split(key)  
  
 round\_keys = []  
 for i in range(2):  
 left = left[LS\_1[i]:] + left[:LS\_1[i]]  
 right = right[LS\_1[i]:] + right[:LS\_1[i]]  
 round\_key = permute(left + right, P8)  
 round\_keys.append(round\_key)  
  
 return round\_keys  
  
# 执行F函数（Feistel函数）  
def feistel(data, sub\_key):  
 data = expand(data)  
 data = xor(data, sub\_key)  
 left, right = split(data)  
 left = s\_box(left, S0)  
 right = s\_box(right, S1)  
 data = permute(left + right, P4)  
 return data  
  
# 使用10位密钥加密8位明文  
def encrypt(plaintext, key):  
 round\_keys = generate\_round\_keys(key)  
 plaintext = initial\_permutation(plaintext)  
 left, right = split(plaintext)  
  
 for i in range(2):  
 left, right = right, xor(left, feistel(right, round\_keys[i]))  
  
 ciphertext = inverse\_initial\_permutation(right + left)  
 return ciphertext  
  
# 使用10位密钥解密8位密文  
def decrypt(ciphertext, key):  
 round\_keys = generate\_round\_keys(key)  
 ciphertext = initial\_permutation(ciphertext)  
 left, right = split(ciphertext)  
  
 for i in range(2):  
 left, right = right, xor(left, feistel(right, round\_keys[1 - i]))  
  
 plaintext = inverse\_initial\_permutation(right + left)  
 return plaintext  
  
# 将二进制字符串转换为整数列表  
def binary\_string\_to\_list(binary\_string):  
 return [int(bit) for bit in binary\_string]  
  
# 将整数列表转换为二进制字符串  
def list\_to\_binary\_string(data\_list):  
 return ''.join(map(str, data\_list))  
  
# 生成一个随机的10位密钥  
def generate\_random\_key():  
 random\_numbers = [random.randint(0, 1) for \_ in range(10)]  
 random\_key = ''.join(map(str, random\_numbers)) # 将随机数列表转换为字符串  
 pyperclip.copy(random\_key) # 将生成的随机数复制到剪贴板  
 return random\_key  
  
# 将ASCII字符转换为8位的二进制字符串  
def ascii\_to\_binary(text):  
 binary\_text = ''.join(format(ord(char), '08b') for char in text)  
 return binary\_text  
  
# 将8位的二进制字符串转换为ASCII字符  
def binary\_to\_ascii(binary\_text):  
 ascii\_text = ''.join(chr(int(binary\_text[i:i+8], 2)) for i in range(0, len(binary\_text), 8))  
 return ascii\_text  
  
# 将ASCII编码的字符串分组成8位数据块，然后加密  
def encrypt\_ascii\_text(ascii\_text, key):  
 binary\_text = ascii\_to\_binary(ascii\_text)  
 encrypted\_binary\_text = ''  
  
 for i in range(0, len(binary\_text), 8):  
 plaintext\_block = list(map(int, binary\_text[i:i+8]))  
 ciphertext\_block = encrypt(plaintext\_block, key)  
 encrypted\_binary\_text += ''.join(map(str, ciphertext\_block))  
  
 return binary\_to\_ascii(encrypted\_binary\_text)  
  
# 将ASCII编码的字符串分组成8位数据块，然后解密  
def decrypt\_ascii\_text(encrypted\_ascii\_text, key):  
 encrypted\_binary\_text = ascii\_to\_binary(encrypted\_ascii\_text)  
 decrypted\_binary\_text = ''  
  
 for i in range(0, len(encrypted\_binary\_text), 8):  
 ciphertext\_block = list(map(int, encrypted\_binary\_text[i:i+8]))  
 plaintext\_block = decrypt(ciphertext\_block, key)  
 decrypted\_binary\_text += ''.join(map(str, plaintext\_block))  
  
 return binary\_to\_ascii(decrypted\_binary\_text)  
  
def chinese\_to\_unicode\_binary(text):  
 binary\_text = ''.join(format(ord(char), '016b') for char in text)  
 return binary\_text  
  
# 将Unicode编码的二进制字符串转换为中文文本  
def unicode\_binary\_to\_chinese(binary\_text):  
 chinese\_text = ''.join(chr(int(binary\_text[i:i+16], 2)) for i in range(0, len(binary\_text), 16))  
 return chinese\_text  
  
# 将中文文本分组成8位数据块，然后加密  
def encrypt\_chinese\_text(chinese\_text, key):  
 unicode\_binary\_text = chinese\_to\_unicode\_binary(chinese\_text)  
 encrypted\_binary\_text = ''  
  
 for i in range(0, len(unicode\_binary\_text), 8):  
 plaintext\_block = list(map(int, unicode\_binary\_text[i:i+8]))  
 ciphertext\_block = encrypt(plaintext\_block, key)  
 encrypted\_binary\_text += ''.join(map(str, ciphertext\_block))  
  
 return unicode\_binary\_to\_chinese(encrypted\_binary\_text)  
  
# 将中文文本分组成8位数据块，然后解密  
def decrypt\_chinese\_text(encrypted\_chinese\_text, key):  
 encrypted\_binary\_text = chinese\_to\_unicode\_binary(encrypted\_chinese\_text)  
 decrypted\_binary\_text = ''  
  
 for i in range(0, len(encrypted\_binary\_text), 8):  
 ciphertext\_block = list(map(int, encrypted\_binary\_text[i:i+8]))  
 plaintext\_block = decrypt(ciphertext\_block, key)  
 decrypted\_binary\_text += ''.join(map(str, plaintext\_block))  
  
 return unicode\_binary\_to\_chinese(decrypted\_binary\_text)  
  
def encrypt\_binary\_text(binary\_text, key):  
 encrypted\_binary\_text = ''  
  
 for i in range(0, len(binary\_text), 8):  
 plaintext\_block = list(map(int, binary\_text[i:i + 8]))  
 ciphertext\_block = encrypt(plaintext\_block, key)  
 encrypted\_binary\_text += ''.join(map(str, ciphertext\_block))  
  
 return encrypted\_binary\_text  
  
  
def decrypt\_binary\_text(binary\_text, key):  
 decrypted\_binary\_text = ''  
  
 for i in range(0, len(binary\_text), 8):  
 ciphertext\_block = list(map(int, binary\_text[i:i + 8]))  
 plaintext\_block = decrypt(ciphertext\_block, key)  
 decrypted\_binary\_text += ''.join(map(str, plaintext\_block))  
  
 return decrypted\_binary\_text  
  
# 测试部分  
def test\_encrypt():  
 plain = '01010101'  
 cipher = '00000100'  
 found\_keys = []  
 key\_search = None  
  
 start\_time = time.time() # 记录开始时间  
  
 for i in range(1024):  
 cipher\_text = ''  
 key = format(i, '010b') # 将整数转换为10位的二进制字符串  
 key = list(map(int, key))  
 cipher\_text = encrypt\_binary\_text(plain, key)  
  
 if cipher\_text == cipher:  
 found\_keys.append(key)  
  
 end\_time = time.time() # 记录结束时间  
 execution\_time = end\_time - start\_time # 计算执行时间  
  
 if found\_keys:  
 print("Found Keys:")  
 for idx, key\_found in enumerate(found\_keys, start=1):  
 print(f"Key {idx}: {key\_found}")  
 else:  
 print("Keys not found")  
  
 print("Execution Time:", execution\_time, "seconds")

\*\*5. 安全性和最佳实践：\*\*

数据安全至关重要。请遵循以下最佳实践：

- 不要分享密钥或密码。

- 使用强密码来保护您的密钥。

- 将密钥存储在安全的地方。

\*\*6. 高级功能和设置：\*\*

本软件提供高级功能和设置，支持生成随机密钥，支持对二进制数、ASCII码以及中文进行加密或解密。

\*\*7. 错误处理和故障排除：\*\*

常见错误消息和问题的解决方法将在本指南中列出，以帮助您在遇到问题时解决它们。

\*\*8. 法律和合规性：\*\*

请确保您在使用本软件时遵守适用的法律和合规性要求。我们不对用户的非法或不当使用负责。

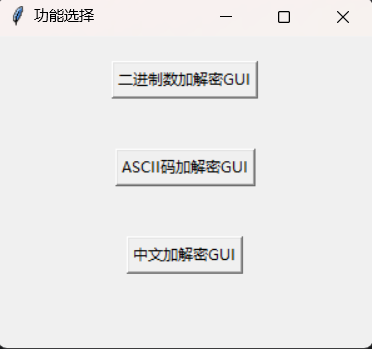
\*\*9. 更新和支持：\*\*

我们会不定期地在Github上对该软件进行更新。如果您需要技术支持或有反馈，请联系我们的支持团队。

\*\*10. 示例和案例：\*\*

为了帮助您更好地理解软件的用途，我们提供了示例和案例，以展示不同情境下的使用方式。

--运行程序，进入功能选择界面

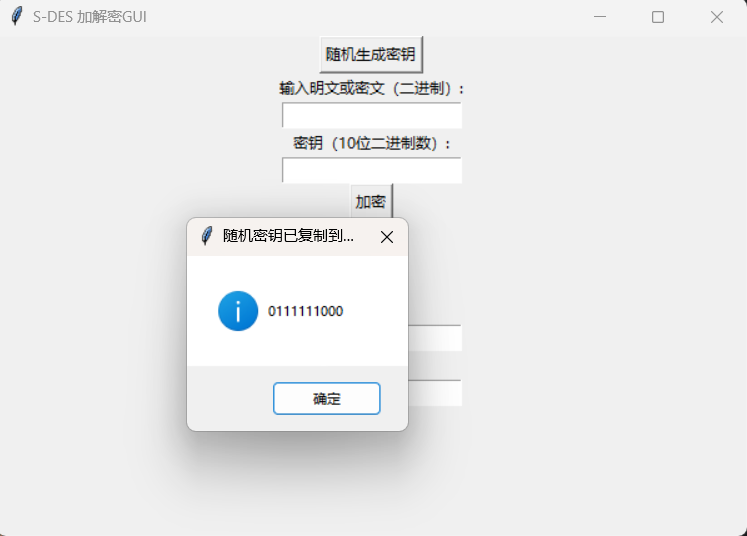


--点击二进制加解密GUI，进入操作界面（二进制加解密GUI）

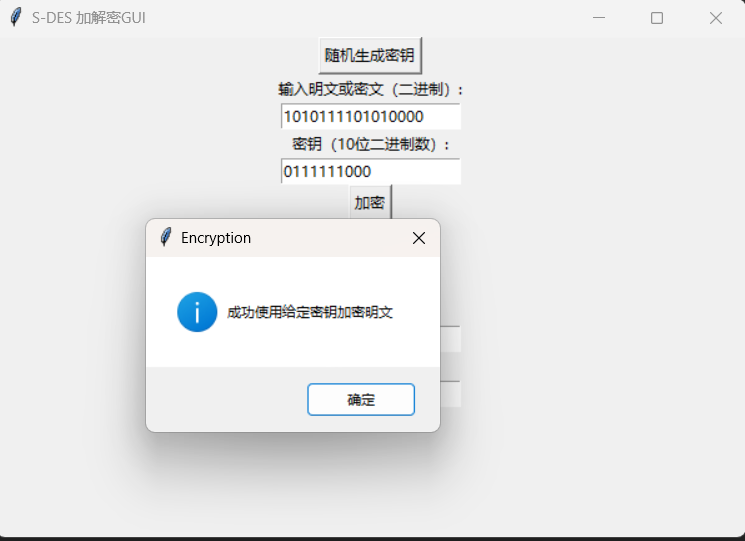
图形用户界面, 应用程序

描述已自动生成

--点击随机生成密钥（密钥已复制到剪切板，直接粘贴到密钥输入框中即可）



--将密钥粘贴到密钥输入框，在明文或密文输入框中输入将要加密的明文(位数须被8整除)，点击加密按钮

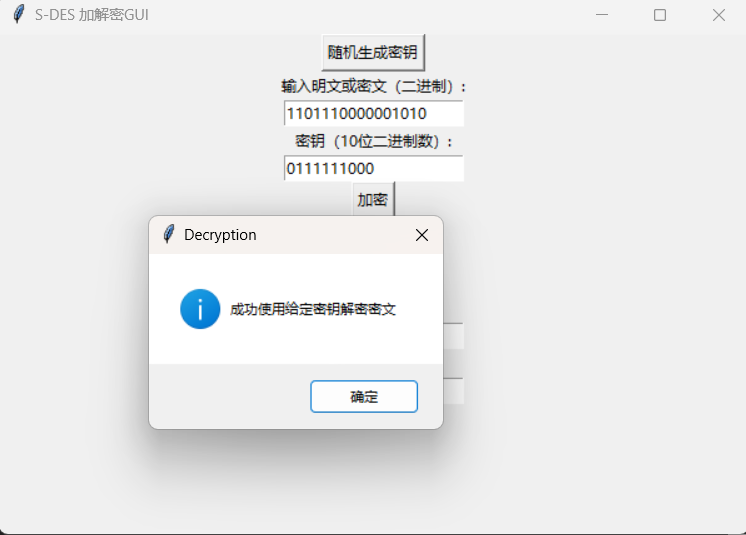


--得到加密结果，显示在密文显示框中

图形用户界面, 应用程序

描述已自动生成

--对密文进行解密同样如此，在明文或密文输入框中输入将要解密的密文，点击解密按钮



--得到解密结果，显示在明文显示框中

图形用户界面, 应用程序

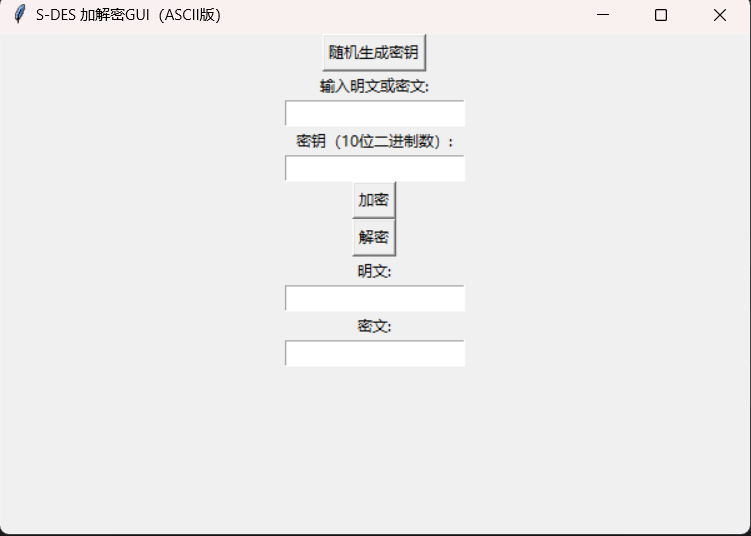
描述已自动生成

--点击测试按钮，可以找到方法内给定明密文对所对应的密钥，并显示验证密钥数量和查找所花费的时间（可在test\_encrypt()方法内修改明密文对，仍可查找其对应的密钥）

电脑萤幕

中度可信度描述已自动生成

--同理，在功能选择界面点击ASCII加解密GUI按钮，进入操作界面（ASCII加解密GUI）

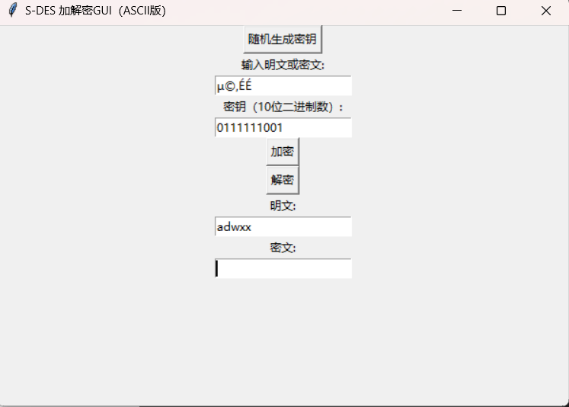


--在密钥输入框中输入密钥（可随机生成），并在明文或密文输入框中输入“adwxx”，点击加密按钮，得到密文“µ©,ÉÉ”

图形用户界面, 应用程序

描述已自动生成

--在明文或密文输入加密后得到的密文“µ©,ÉÉ”，保持密钥不变，点击解密按钮，得到明文“adwxx”显示在明文显示框中



--同理，在功能选择界面点击中文加解密GUI按钮，进入操作界面（中文加解密GUI）



--在密钥输入框中输入密钥（可随机生成），并在明文或密文输入框中输入“祝你中秋国庆快乐”，点击加密按钮，得到密文“퐽⋍앏퓁⺳僫繩없”，显示在密文显示框中

图形用户界面, 应用程序

描述已自动生成

--在明文或密文输入加密后得到的密文“퐽⋍앏퓁⺳僫繩없”，保持密钥不变，点击解密按钮，得到明文“祝你中秋国庆快乐”显示在明文显示框中

图形用户界面, 应用程序

描述已自动生成

\*\*11. 安全注意事项：\*\*

强调用户不应分享密钥、密码或加密数据，以及在使用软件时应保持警惕。

\*\*12. 测试和校对：\*\*

我们经过充分的测试和校对，确保用户指南没有拼写或语法错误。我们的目标是提供清晰和准确的指南，以帮助用户轻松使用本软件。

希望这份用户指南能够帮助您了解并正确使用S-DES加密软件。如果您有任何问题或需要进一步的帮助，请随时联系我们的支持团队。