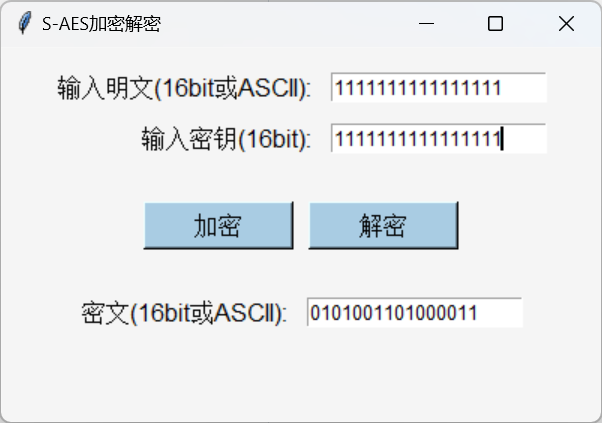
**3.1 第1关：基本测试**

根据S-AES算法编写和调试程序，提供GUI解密支持用户交互。输入可以是16bit的数据和16bit的密钥，输出是16bit的密文。

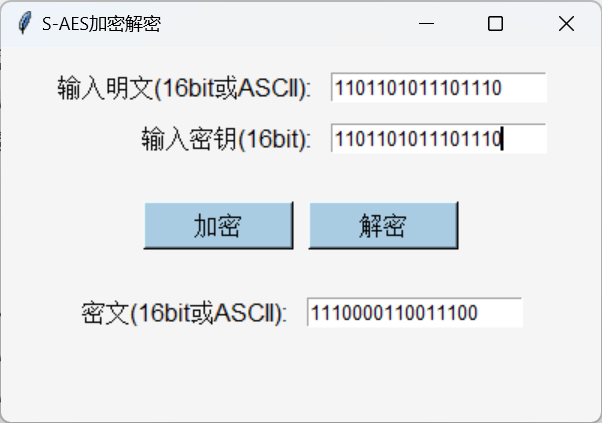


**3.2 第2关：交叉测试**

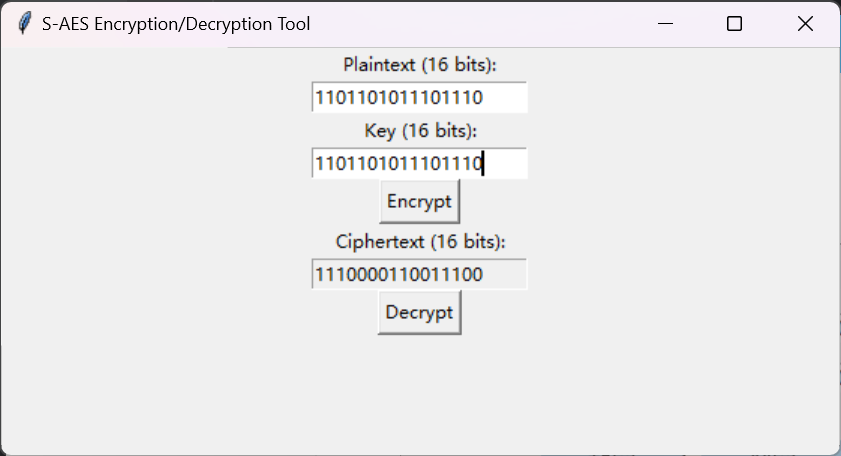
考虑到是"**算法标准"**，所有人在编写程序的时候需要使用相同算法流程和转换单元(替换盒、列混淆矩阵等)，以保证算法和程序在异构的系统或平台上都可以正常运行。

设有A和B两组位同学(选择相同的密钥K)；则A、B组同学编写的程序对明文P进行加密得到相同的密文C；或者B组同学接收到A组程序加密的密文C，使用B组程序进行解密可得到与A相同的P。

赵浚杰组：

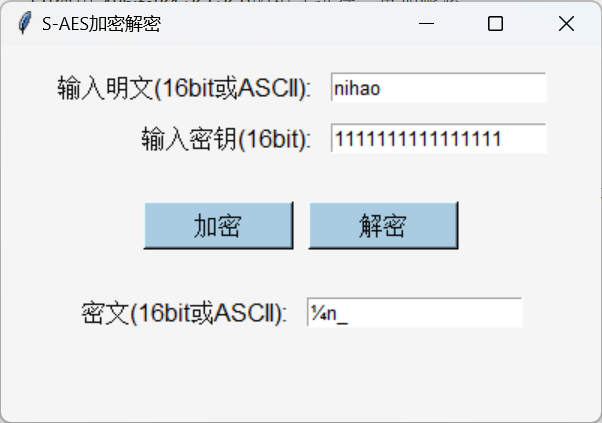


李殊斐，周琦组：



**3.3 第3关：扩展功能**

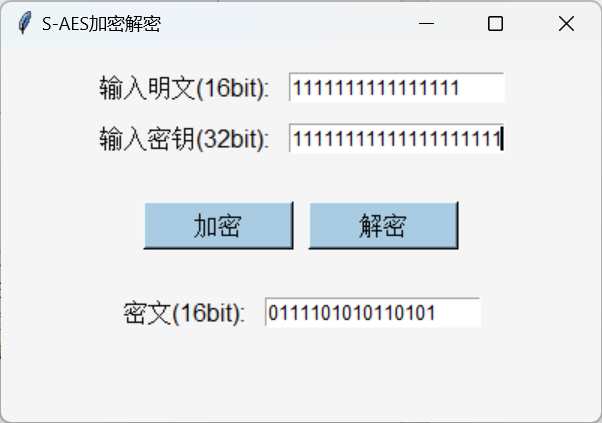
考虑到向实用性扩展，加密算法的数据输入可以是ASII编码字符串(分组为2 Bytes)，对应地输出也可以是ACII字符串(很可能是乱码)。



**3.4 第4关：多重加密**

3.4.1 双重加密

将S-AES算法通过双重加密进行扩展，分组长度仍然是16 bits，但密钥长度为32 bits。



3.4.2 中间相遇攻击

假设你找到了使用相同密钥的明、密文对(一个或多个)，请尝试使用中间相遇攻击的方法找到正确的密钥Key(K1+K2)。



3.4.3 三重加密

将S-AES算法通过三重加密进行扩展，下面两种模式选择一种完成：

(1)按照32 bits密钥Key(K1+K2)的模式进行三重加密解密，

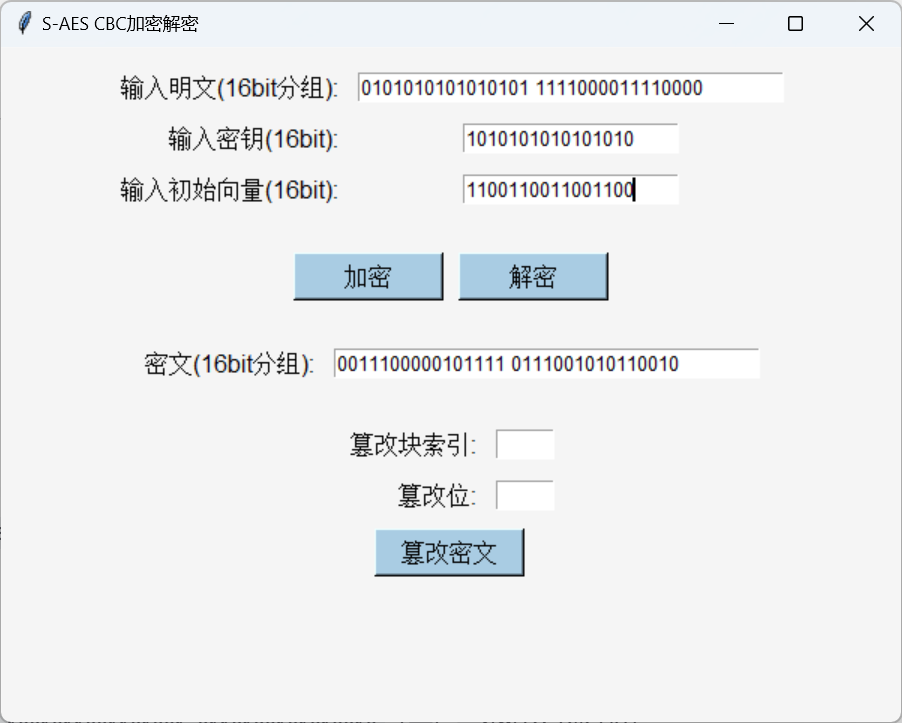
(2)使用48bits(K1+K2+K3)的模式进行三重加解密。



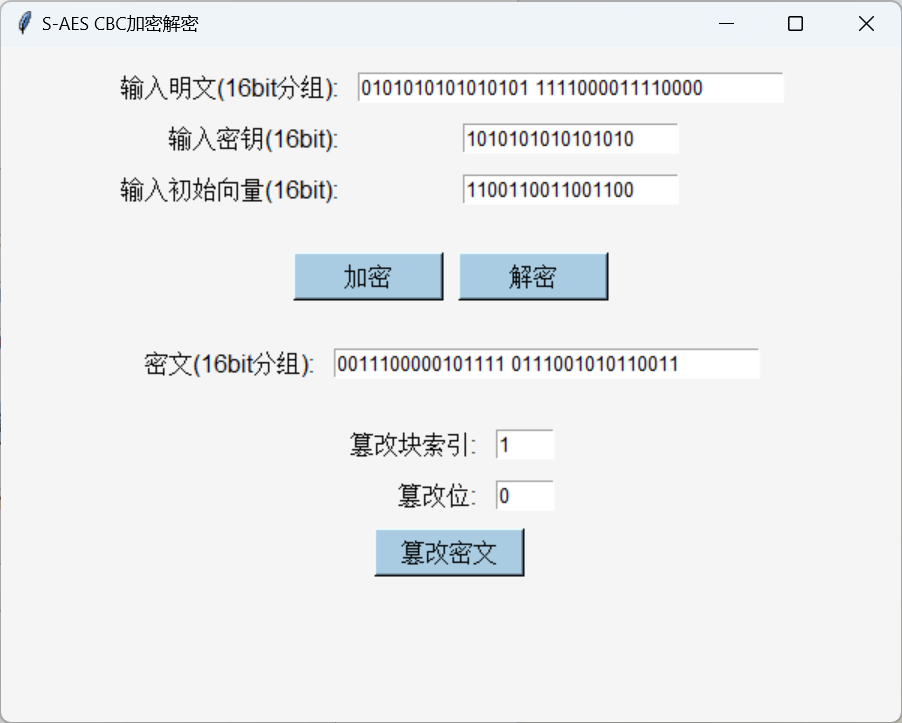
**3.5 第5关：工作模式**

基于S-AES算法，使用密码分组链(CBC)模式对较长的明文消息进行加密。注意初始向量(16 bits) 的生成，并需要加解密双方共享。

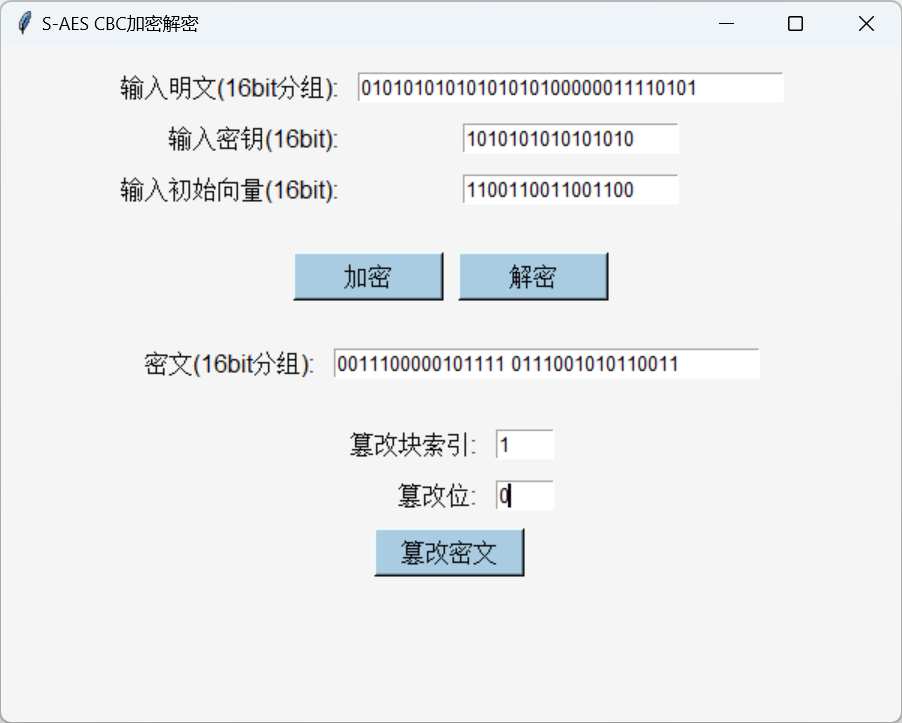
在CBC模式下进行加密，并尝试对密文分组进行替换或修改，然后进行解密，请对比篡改密文前后的解密结果。



篡改



解密：



用户指南

1. 程序概述

本程序实现了S-AES算法的加密和解密功能，并支持多种扩展功能，包括双重加密、三重加密和CBC模式。程序提供了图形用户界面（GUI），方便用户进行交互操作。

2. 系统要求

Python 3.x

Tkinter库（通常随Python安装）

任意文本编辑器或IDE（如PyCharm、VSCode等）

3. 安装步骤

确保已安装Python 3.x。

打开命令行或终端，运行以下命令安装所需的库（如果尚未安装）：

sh

深色版本

pip install tk

4. 使用说明

4.1 基本测试

运行程序：

sh

深色版本

python saes\_gui.py

在“输入明文(16bit)”输入框中输入16位的二进制数据，例如 0101010101010101。

在“输入密钥(16bit)”输入框中输入16位的二进制密钥，例如 1010101010101010。

点击“加密”按钮，加密后的密文会显示在“密文(16bit)”输入框中。

点击“解密”按钮，解密后的明文会显示在“输入明文(16bit)”输入框中。

4.2 交叉测试

A组和B组使用相同的密钥进行加密和解密。

A组使用自己的程序对明文进行加密，得到密文。

B组使用自己的程序对接收到的密文进行解密，应得到与A组相同的明文。

4.3 扩展功能

4.3.1 ASCII编码字符串

在“输入明文(ASCII)”输入框中输入ASCII字符串，例如 Hello。

点击“加密(ASCII)”按钮，加密后的密文会显示在“密文(ASCII)”输入框中。

点击“解密(ASCII)”按钮，解密后的明文会显示在“输入明文(ASCII)”输入框中。

4.3.2 双重加密

在“输入明文(16bit)”输入框中输入16位的二进制数据，例如 0101010101010101。

在“输入密钥1(16bit)”和“输入密钥2(16bit)”输入框中分别输入两个16位的二进制密钥，例如 1010101010101010 和 0101010101010101。

点击“双重加密”按钮，双重加密后的密文会显示在“密文(16bit)”输入框中。

点击“双重解密”按钮，解密后的明文会显示在“输入明文(16bit)”输入框中。

4.3.3 三重加密

在“输入明文(16bit)”输入框中输入16位的二进制数据，例如 0101010101010101。

在“输入密钥1(16bit)”、“输入密钥2(16bit)”和“输入密钥3(16bit)”输入框中分别输入三个16位的二进制密钥，例如 1010101010101010、0101010101010101 和 1111000011110000。

点击“三重加密”按钮，三重加密后的密文会显示在“密文(16bit)”输入框中。

点击“三重解密”按钮，解密后的明文会显示在“输入明文(16bit)”输入框中。

4.4 工作模式

4.4.1 CBC模式

在“输入明文(16bit分组)”输入框中输入多组16位的二进制数据，例如 0101010101010101 1111000011110000。

在“输入密钥(16bit)”输入框中输入16位的二进制密钥，例如 1010101010101010。

在“输入初始向量(16bit)”输入框中输入16位的二进制初始向量，例如 1100110011001100。

点击“CBC加密”按钮，加密后的密文会显示在“密文(16bit分组)”输入框中。

点击“篡改密文”按钮，输入要篡改的块索引和位位置，例如 1 和 0，点击“篡改密文”按钮，篡改后的密文会显示在“密文(16bit分组)”输入框中。

点击“CBC解密”按钮，解密后的明文会显示在“输入明文(16bit分组)”输入框中。

开发手册

1. 组件封装

1.1 S-AES核心算法

python

深色版本

# 密钥扩展函数

def key\_expansion(key):

w = [0] \* 6

w[0] = (key & 0xFF00) >> 8

w[1] = key & 0x00FF

w[2] = w[0] ^ 0x80 ^ sub\_nibble(w[1])

w[3] = w[2] ^ w[1]

w[4] = w[2] ^ 0x30 ^ sub\_nibble(w[3])

w[5] = w[4] ^ w[3]

return w

def sub\_nibble(nibble):

return (S\_BOX[nibble & 0x0F] << 4) | S\_BOX[(nibble & 0xF0) >> 4]

# 字节替代和逆替代

def sub\_bytes(state):

return ((S\_BOX[(state & 0xF000) >> 12] << 12) |

(S\_BOX[(state & 0x0F00) >> 8] << 8) |

(S\_BOX[(state & 0x00F0) >> 4] << 4) |

S\_BOX[state & 0x000F])

def inv\_sub\_bytes(state):

return ((INV\_S\_BOX[(state & 0xF000) >> 12] << 12) |

(INV\_S\_BOX[(state & 0x0F00) >> 8] << 8) |

(INV\_S\_BOX[(state & 0x00F0) >> 4] << 4) |

INV\_S\_BOX[state & 0x000F])

# 轮密钥加函数

def add\_round\_key(state, key):

return state ^ key

# 行移位操作

def shift\_rows(state):

row0 = state & 0xF0F0

row1 = ((state & 0x0F00) >> 8) | ((state & 0x000F) << 8)

return row0 | row1

# 逆行移位操作

def inv\_shift\_rows(state):

row0 = state & 0xF0F0

row1 = ((state & 0x000F) << 8) | ((state & 0x0F00) >> 8)

return row0 | row1

# 列混合和逆列混合操作

def mix\_columns(state):

t0, t2, t1, t3 = (state & 0xF000) >> 12, (state & 0x0F00) >> 8, (state & 0x00F0) >> 4, state & 0x000F

return ((t0 ^ mul4(t2)) << 12) | ((t2 ^ mul4(t0)) << 8) | ((t1 ^ mul4(t3)) << 4) | (t3 ^ mul4(t1))

def inv\_mix\_columns(state):

t0, t2, t1, t3 = (state & 0xF000) >> 12, (state & 0x0F00) >> 8, (state & 0x00F0) >> 4, state & 0x000F

return ((mul9(t0) ^ mul2(t2)) << 12) | ((mul2(t0) ^ mul9(t2)) << 8) | ((mul9(t1) ^ mul2(t3)) << 4) | (mul2(t1) ^ mul9(t3))

# 有限域运算辅助函数

def mul2(nibble):

return ((nibble << 1) & 0xF) ^ 0x3 if (nibble & 0x8) else (nibble << 1) & 0xF

def mul4(nibble):

return mul2(mul2(nibble)) & 0xF

def mul9(nibble):

return (mul4(mul2(nibble)) ^ nibble) & 0xF

# 加密和解密函数

def s\_aes\_encrypt(plaintext, key):

w = key\_expansion(key)

state = add\_round\_key(plaintext, (w[0] << 8) | w[1])

state = mix\_columns(shift\_rows(sub\_bytes(state)))

state = add\_round\_key(state, (w[2] << 8) | w[3])

return add\_round\_key(shift\_rows(sub\_bytes(state)), (w[4] << 8) | w[5])

def s\_aes\_decrypt(ciphertext, key):

w = key\_expansion(key)

state = add\_round\_key(ciphertext, (w[4] << 8) | w[5])

state = inv\_shift\_rows(inv\_sub\_bytes(state))

state = add\_round\_key(state, (w[2] << 8) | w[3])

state = inv\_mix\_columns(state)

state = inv\_shift\_rows(inv\_sub\_bytes(state))

return add\_round\_key(state, (w[0] << 8) | w[1])

1.2 双重加密

python

深色版本

def double\_s\_aes\_encrypt(plaintext, key1, key2):

state = s\_aes\_encrypt(plaintext, key1)

return s\_aes\_encrypt(state, key2)

def double\_s\_aes\_decrypt(ciphertext, key1, key2):

state = s\_aes\_decrypt(ciphertext, key2)

return s\_aes\_decrypt(state, key1)

1.3 三重加密

python

深色版本

def triple\_s\_aes\_encrypt(plaintext, key1, key2, key3):

state = s\_aes\_encrypt(plaintext, key1)

state = s\_aes\_encrypt(state, key2)

return s\_aes\_encrypt(state, key3)

def triple\_s\_aes\_decrypt(ciphertext, key1, key2, key3):

state = s\_aes\_decrypt(ciphertext, key3)

state = s\_aes\_decrypt(state, key2)

return s\_aes\_decrypt(state, key1)

1.4 CBC模式

python

深色版本

def cbc\_encrypt(plaintext, key, iv):

ciphertext\_blocks = []

prev\_ciphertext = iv

for i in range(0, len(plaintext), 16):

block = plaintext[i:i+16]

if len(block) < 16:

block += '0' \* (16 - len(block))

block = int(block, 2)

block ^= prev\_ciphertext

ciphertext\_block = s\_aes\_encrypt(block, key)

ciphertext\_blocks.append(ciphertext\_block)

prev\_ciphertext = ciphertext\_block

return ciphertext\_blocks

def cbc\_decrypt(ciphertext\_blocks, key, iv):

plaintext\_blocks = []

prev\_ciphertext = iv

for ciphertext\_block in ciphertext\_blocks:

block = s\_aes\_decrypt(ciphertext\_block, key)

block ^= prev\_ciphertext

plaintext\_blocks.append(format(block, '016b'))

prev\_ciphertext = ciphertext\_block

return ''.join(plaintext\_blocks)

1.5 ASCII编码支持

python

深色版本

def ascii\_to\_binary(ascii\_str):

return ''.join(format(ord(char), '016b') for char in ascii\_str)

def binary\_to\_ascii(binary\_str):

return ''.join(chr(int(binary\_str[i:i+16], 2)) for i in range(0, len(binary\_str), 16))

2. 接口文档

2.1 S-AES核心算法

key\_expansion(key: int) -> List[int]: 密钥扩展函数，返回扩展后的密钥列表。

sub\_bytes(state: int) -> int: 字节替代函数。

inv\_sub\_bytes(state: int) -> int: 逆字节替代函数。

add\_round\_key(state: int, key: int) -> int: 轮密钥加函数。

shift\_rows(state: int) -> int: 行移位操作。

inv\_shift\_rows(state: int) -> int: 逆行移位操作。

mix\_columns(state: int) -> int: 列混合操作。

inv\_mix\_columns(state: int) -> int: 逆列混合操作。

s\_aes\_encrypt(plaintext: int, key: int) -> int: S-AES加密函数。

s\_aes\_decrypt(ciphertext: int, key: int) -> int: S-AES解密函数。

2.2 双重加密

double\_s\_aes\_encrypt(plaintext: int, key1: int, key2: int) -> int: 双重加密函数。

double\_s\_aes\_decrypt(ciphertext: int, key1: int, key2: int) -> int: 双重解密函数。

2.3 三重加密

triple\_s\_aes\_encrypt(plaintext: int, key1: int, key2: int, key3: int) -> int: 三重加密函数。

triple\_s\_aes\_decrypt(ciphertext: int, key1: int, key2: int, key3: int) -> int: 三重解密函数。

2.4 CBC模式

cbc\_encrypt(plaintext: str, key: int, iv: int) -> List[int]: CBC模式加密函数。

cbc\_decrypt(ciphertext\_blocks: List[int], key: int, iv: int) -> str: CBC模式解密函数。

2.5 ASCII编码支持

ascii\_to\_binary(ascii\_str: str) -> str: 将ASCII字符串转换为二进制字符串。

binary\_to\_ascii(binary\_str: str) -> str: 将二进制字符串转换为ASCII字符串。

总结

通过以上用户指南和开发手册，您应该能够顺利地使用和开发S-AES算法的各种功能。如果您有任何问题或需要进一步的帮助，请随时联系。祝您开发愉快！