第一关：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。

选择使用简单轻便的tkinter作为UI显示交互的框架。具体交互界面如下图所示  


明文数据：1000000000000000

密钥：1000000000000000

输入：0010000110111000

输入明文数据和密钥数据之后，点击加密按钮，输入框就会显示密文



第二关：交叉测试  
 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。

在UI界面上选择与第一关相同的密钥：1000000000000000，用第一关的密文：0010000110111000进行解密，得到与第一关明文相同的结果：1000000000000000



第三关：扩展功能

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)。

输入：Helloaaa  
 密钥：1000000000000000

密文：É0Îï¿

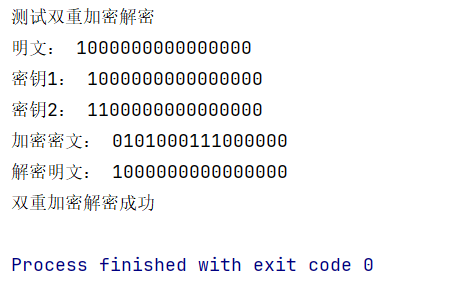


第四关：多重加密  
 4.1双重加密将S-AES算法通过双重加密进行扩展，分组长度仍然是16 bits，但密钥长度为32 bits。

代码实现：

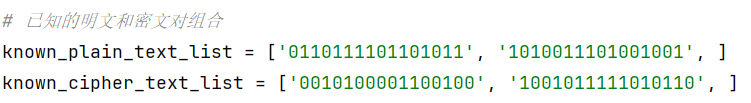
def double\_encrypt(plain\_text, key1, key2):  
 cipher\_text1 = Encrypt(plain\_text, key1)  
 cipher\_text2 = Decrypt(cipher\_text1, key2)  
 return cipher\_text2  
*# 实现双重解密*def double\_decrypt(cipher\_text, key1, key2):  
 plain\_text1 = Encrypt(cipher\_text, key2)  
 plain\_text2 = Decrypt(plain\_text1, key1)  
 return plain\_text2

得出明文与解密得出的明文相同。关键在于加密和解密的设计使得这两个不同的密钥能够在特定的顺序下完成相互抵消的作用

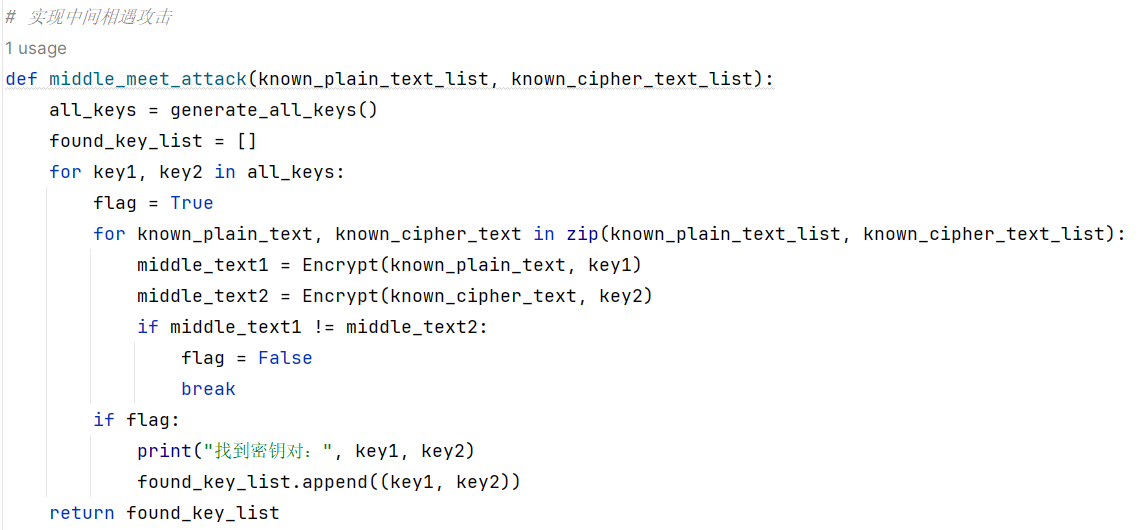


4.2假设你找到了使用相同密钥的明、密文对(一个或多个)，请尝试使用中间相遇攻击的方法找到正确的密钥Key(K1+K2)

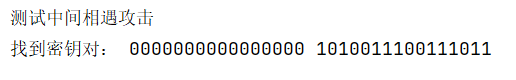
验证的正确密码为：0000000000000000 1010011100111011

已知明文对：  
 

攻击代码：



结果：



4.3三重加密，按照32 bits密钥Key(K1+K2)的模式进行三重加密解密。

加密代码：

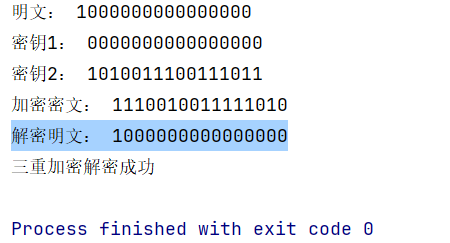
*# 实现三重加密*def triple\_encrypt(plain\_text, key1, key2):  
 cipher\_text1 = Encrypt(plain\_text, key1)  
 cipher\_text2 = Decrypt(cipher\_text1, key2)  
 cipher\_text3 = Encrypt(cipher\_text2, key1)  
 return cipher\_text3  
  
*# 实现三重解密*def triple\_decrypt(cipher\_text, key1, key2):  
 plain\_text1 = Decrypt(cipher\_text, key1)  
 plain\_text2 = Encrypt(plain\_text1, key2)  
 plain\_text3 = Decrypt(plain\_text2, key1)  
 return plain\_text3

输入明文：1000000000000000

密钥1： 0000000000000000

密钥2： 1010011100111011

测试结果：



第五关：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。

CBC 模式的对称加密和解密，它使用异或操作和密钥加密每个明文块，并通过初始向量链接每个加密操作。这种模式能够提供更强的安全性，因为相同的明文在不同的加密过程中会产生不同的密文

主要代码：

def CBC\_encrypt(plain\_text, key, IV):  
 *# 将明文分组* plain\_text\_list = [plain\_text[i:i + 16] for i in range(0, len(plain\_text), 16)]  
 *# 用于存储密文* cipher\_text\_list = []  
 *# 对每个分组进行加密* for plain\_text in plain\_text\_list:  
 *# 执行加密* cipher\_text = Encrypt(XOR(plain\_text, IV), key)  
 *# 更新初始向量* IV = cipher\_text  
 *# 将密文添加到密文列表* cipher\_text\_list.append(cipher\_text)  
 *# 将密文列表转换为字符串* cipher\_text = ''.join(cipher\_text\_list)  
 return cipher\_text  
  
*# 实现密码分组链模式的解密*def CBC\_decrypt(cipher\_text, key, IV):  
 *# 将密文分组* cipher\_text\_list = [cipher\_text[i:i + 16] for i in range(0, len(cipher\_text), 16)]  
 *# 用于存储明文* plain\_text\_list = []  
 *# 对每个分组进行解密* for cipher\_text in cipher\_text\_list:  
 *# 执行解密* plain\_text = XOR(Decrypt(cipher\_text, key), IV)  
 *# 更新初始向量* IV = cipher\_text  
 *# 将明文添加到明文列表* plain\_text\_list.append(plain\_text)  
 *# 将明文列表转换为字符串* plain\_text = ''.join(plain\_text\_list)  
 return plain\_text

测试结果：

