**S-AES原理解释**

**第一关：基本测试**

1. **加密测试**

1.1步骤：

分组和密钥：S-AES处理16位的数据块，使用一个16位的密钥。输入的数据块被分为4个4位的字节。

轮次：S-AES通常包含3轮加密。每轮包括几个步骤。

·步骤：

字节替代：通过查找一个S-Box（替代盒），将每个字节替换为另一个字节。

行移位：对每一行的字节进行循环左移，增加扩散。

列混合：对每列的字节进行线性变换，进一步增加扩散（在最后一轮中省略此步骤）。

轮密钥加：将当前状态与轮密钥进行按位异或操作。

1.2测试样例：

（1）测试样例，如图1所示：

明文（P）：1010010111110000

密钥（K）：1100111000101010



图1：加密测试

**2、解密测试**

2.1步骤：

·解密过程与加密相反，主要步骤包括：

逆轮密钥加：首先对密文进行轮密钥加操作。

逆列混合：对每列进行逆变换（除最后一轮）。

逆行移位：对行进行逆向循环移位。

逆字节替代：使用逆S-Box进行字节替代。

2.2测试样例：

示例说明，如图2所示：

·密文（C）：0100010100111000

·密钥（K）：1100111000101010



图2：解密测试

**3、总结**

**3.1相同点**

1、算法结构：加密和解密都使用相同的基本结构，包括多个轮次（Rounds）和操作步骤（如字节替代、行移位等）。

2、密钥使用：两者都依赖于相同的密钥调度过程，以生成不同轮次所需的轮密钥。

3、数据处理单元：加密和解密都是针对相同的数据块（如128位），并以相同的方式处理。

4、安全性目标：两者的目标都是保护数据的机密性和安全性，确保数据在传输或存储过程中的安全。

**3.2不同点**

1、操作顺序：

·加密：遵循特定的顺序进行字节替代、行移位、列混合和轮密钥加。

·解密：步骤顺序相反，使用逆操作（如逆字节替代、逆行移位、逆列混合）。

2、使用的函数：

·加密：使用S-Box进行字节替代。

·解密：使用逆S-Box进行字节替代。

3、轮次处理：在解密过程中，最后一轮省略列混合，而加密的最后一轮则包括列混合。

4、目的：

·加密：将明文转换为密文，防止未授权访问。

·解密：将密文恢复为明文，确保合法用户能够访问原始数据。

**第二关：交叉测试**

1. **我方加密结果，如图3所示**

****

图3：我方加密结果

1. **对方加密结果，如图4所示**

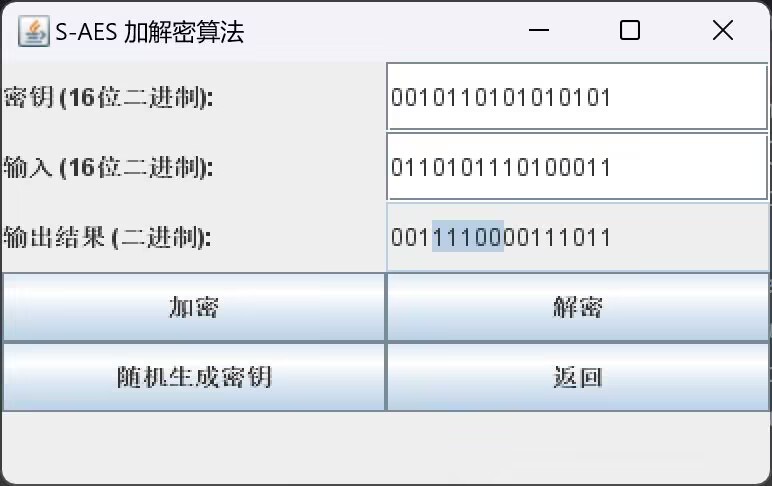
****

图4：对方加密结果

1. **总结**

**5.1原理与操作步骤**

通过组间的合作与验证，测试简化数据加密标准（S-DES）加密和解密的有效性与一致性。通过设定同样的密钥 KKK，我方和对方可以分别验证：

·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。

**5.2常见问题与解决方案**

密钥选择不一致：若密钥选择不一致，即使程序正确，密文和解密结果也不会一致。需要确保两组使用的密钥完全相同。

加密算法实现差异：不同的实现可能导致加密输出不同。

程序调试：在解密阶段，若无法还原明文，可能需要检查密文传递、密钥使用或S-AES解密过程的细节。

**第三关:扩展功能（ASCII编码字符串）**

**1、相关基本原理**

1. 数据准备

ASCII转换：将明文字符转换为ASCII码，通常是将每个字符的ASCII值表示为8位二进制数。这意味着每个字符（如字母、数字和符号）都会转换成相应的数字形式。

分组：AES处理的单位是128位（16字节），因此在加密之前，需要将ASCII码数据填充到16字节的长度。如果数据长度不足16字节，可以使用填充方案（如PKCS#7）来填充。

2. 加密过程

加密过程遵循AES算法的步骤：

轮密钥加（AddRoundKey）：将明文与轮密钥进行按位异或操作。

字节替代（SubBytes）：使用S-Box将每个字节替换为另一个字节。

行移位（ShiftRows）：对状态矩阵的行进行循环左移。

列混合（MixColumns）：对每列进行线性变换，增加数据的扩散。

这些步骤会重复进行多个轮次（通常为10轮、12轮或14轮，具体取决于密钥长度）。

3. 解密过程

解密过程与加密相反，步骤如下：

逆轮密钥加：首先对密文进行逆轮密钥加。

逆列混合（Inverse MixColumns）：对每列进行逆变换（在最后一轮省略此步骤）。

逆行移位（Inverse ShiftRows）：对行进行逆向循环移位。

逆字节替代（Inverse SubBytes）：使用逆S-Box进行字节替代。

4. 结果转换

ASCII恢复：解密后得到的明文以二进制形式呈现，需要将其转换回ASCII字符，恢复原始文本。

**2、加密测试结果展示，如图5所示**



图5：ASCII编码的加密测试

**3、解密测试结果展示，如图6所示**



图6：ASCII编码的解密测试

**第四关：多重加密**

**1. 双重加密**

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。

**1.1原理：**

双重加密是指对同一明文进行两次AES加密，通常使用两个不同的密钥。

加密过程如下：

·使用第一个密钥对明文进行AES加密，得到中间密文。

·使用第二个密钥对中间密文进行AES加密，得到最终密文。

**1.2优势：**

·提高安全性。即使第一个密钥被破解，第二个密钥仍然可以保护数据。

·适用于对安全要求较高的场合。

**1.3成果展示**

·双重加密测试，如图7所示：



图7：双重加密测试

·双重解密测试，如图8所示：

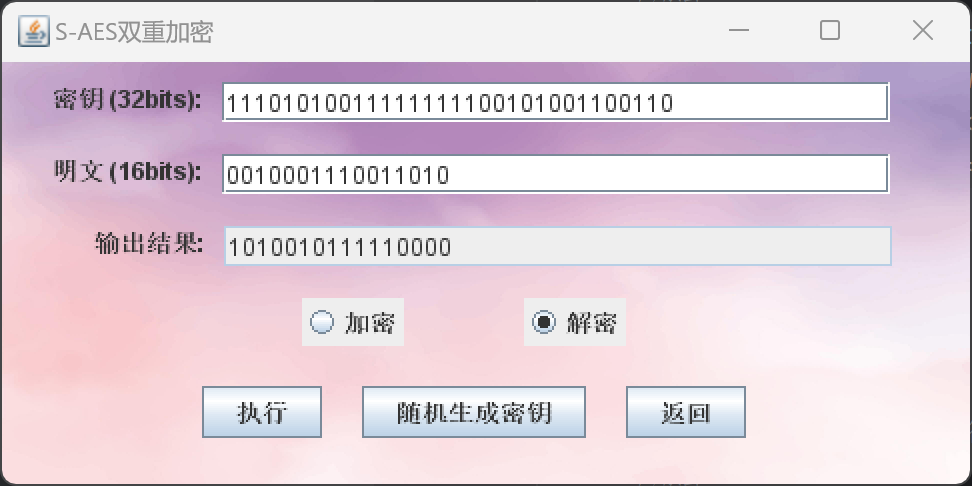


图8：双重解密测试

**2. 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。

**2.1原理：**

三重加密进一步增加了安全性，对同一明文进行三次AES加密，通常使用三个密钥。

加密过程如下：

·使用第一个密钥对明文进行加密。

·使用第二个密钥对第一次加密的结果进行加密。

·使用第三个密钥对第二次加密的结果进行加密，得到最终密文。

**2.2优势：**

·提供更高的安全性，抵抗更强的攻击。

·常用于非常敏感的数据传输和存储。

**2.3成果展示**

·三重加密测试，如图9所示：



图9：三重加密测试

·三重解密测试，如图10所示：



图10：三重解密测试

**3、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)。

**3.1原理：**

中间相遇攻击（Meet-in-the-Middle Attack）是一种针对双重加密的攻击方式。

攻击者尝试通过同时计算两个方向的中间值来减少需要的时间和计算量。具体步骤如下：

对已知明文进行第一次AES加密，计算出所有可能的中间密文。

对相同的已知密文进行反向AES解密，生成可能的中间明文。

比较两个步骤得到的中间值，找到匹配的值，从而推断出密钥。

**3.2影响：**

中间相遇攻击通过利用双重加密的结构来显著降低所需的计算复杂度，理论上将复杂度从22n2^{2n}22n（单个密钥的强度）降低到2n2^n2n（针对两个密钥的复杂度）。

**3.3成果展示**

·中间相遇攻击，如图11所示

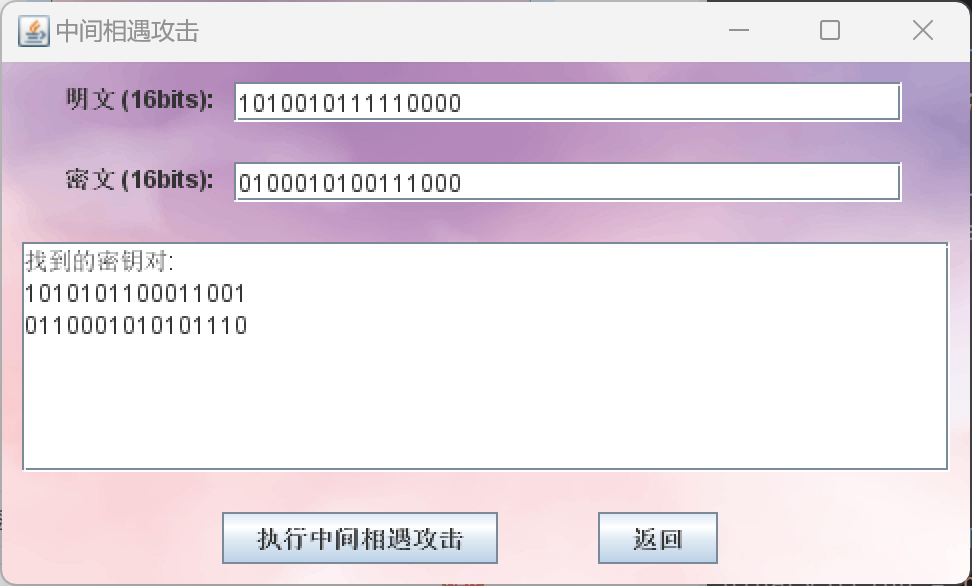


图11：中间相遇攻击测试

**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。

**1. 初始向量 (IV) 的生成**

在CBC模式中，初始向量（IV）是随机生成的，并且需要加解密双方共享。IV的长度通常与块长度相同，对于S-AES来说为16位（2字节）。

**2. 加密过程**

在CBC模式中，加密过程如下：

将明文分为多个块，如果最后一个块不足16位，则进行填充。

对于第一个明文块，将其与IV进行异或操作，然后用S-AES加密。

对于后续块，将其与前一个加密结果（密文块）进行异或操作后加密。

2.1加密测试，成果展示

·CBC模式32bit加密，如图12所示



图12：CBC模式32bit加密

·CBC模式48bit加密，如图13所示



图13：CBC模式48bit加密

·CBC模式64bit加密，如图14所示



图14：CBC模式64bit加密

**3. 解密过程**

解密过程与加密相反。将密文分为块，对于每个密文块，先解密，然后将结果与前一个密文块（对于第一个块则是IV）进行异或操作，得到明文。

**3.1解密测试**，成果展示：

·CBC模式64bit解密，如图15所示



图15：CBC模式64bit解密

解密结果前：如果没有篡改，解密后得到的结果应该是原始明文。

解密结果后：篡改密文后，解密可能会导致出现乱码或者部分正确的字符，因为只篡改了一个密文块，导致其后续块的解密结果受到影响。

**3.2总结：**

在CBC模式下，密文的篡改会对解密后的结果产生显著影响。这表明CBC模式对密文的完整性是敏感的，篡改任何一个密文块都会导致解密错误或产生不可预测的结果。这也强调了在实际应用中使用完整性保护措施（如HMAC）的重要性。