**S-DES原理解释**

**第一关：基本测试**

**1、加密测试**

1.1步骤：

（1）密钥生成：S-DES使用一个10位的密钥来生成两个子密钥K1和K2。密钥生成分为以下几个步骤：

·从10位密钥中选择8位作为有效密钥。

·将密钥分成两部分，分别进行循环左移。

·通过选择变换生成子密钥K1和K2。

（2）初始置换：将明文进行初始置换（IP），此步骤对明文的位顺序进行改变，以增加安全性。

（3）轮函数：S-DES的加密过程包含两个主要轮次（Round 1和Round 2），每个轮次都使用不同的子密钥：

·每个轮次将当前数据分为两半，左半部分与右半部分通过轮函数进行组合。

·轮函数包括扩展置换、S-Box替换和P-Box置换。

（4）逆初始置换：最后将经过轮函数处理的数据再经过逆初始置换（IP逆）以得到最终的密文。

1.2测试样例：

（1）测试样例：

明文（P）：10101010（假设的8位明文）

密钥（K）：1010000010（10位密钥）

（2）步骤：

·密钥生成：

从K中生成K1和K2，例如K1=10100000，K2=10001000。

·初始置换：

对明文P进行初始置换（IP），假设结果为11100000。

·加密过程：

轮1：使用K1处理数据，执行扩展置换、S-Box替换和P-Box替换。

轮2：使用K2处理经过轮1的结果。

·逆初始置换：

最终结果经过逆初始置换，得到密文。

**2、解密测试**

2.1步骤：

1. 密钥生成：

解密过程中使用的密钥与加密时生成的子密钥K1和K2相同，但使用顺序相反：首先使用K2，然后使用K1进行解密。

1. 初始置换：

解密过程的第一步是对密文进行初始置换（IP），与加密时相同。

（3）轮函数：

·解密过程同样包括两个轮次（Round 1和Round 2），但此时的操作使用的是反向的子密钥：

·数据被分为左右两部分，左半部分与右半部分通过轮函数进行结合。

·轮函数的具体步骤同加密过程，包括扩展置换、S-Box替换和P-Box置换。

（4）逆初始置换：

·最后，将经过两轮处理的数据进行逆初始置换（IP逆），得到原始明文。

2.2测试样例：

示例说明：

·密文（C）：11011001

·密钥（K）：1010000010

2.3步骤：

（1）密钥生成：

·从10位密钥生成两个子密钥K1和K2。例如：

K1：10100000

K2：10001000

（2）初始置换：

·密文C经过初始置换（IP），假设结果为：01001100

（3）解密过程：

·轮1（使用K2）：

·将数据分为左右两部分，假设左半部分为0100，右半部分为1100。

·对右半部分进行扩展置换（得到11000000），然后与K2进行异或运算。

·经过S-Box替换和P-Box置换，得到新的右半部分。

·将新的右半部分与左半部分结合，形成新的数据。

·轮2（使用K1）：

·使用K1处理新的数据，重复轮1的过程。

（4）逆初始置换：

·最终结果经过逆初始置换，得到原始明文。例如：明文可能为 10101010。

**3、总结**

3.1相同点

（1）结构和步骤：

加密和解密过程都包含相似的基本步骤，包括初始置换（IP）、轮函数处理和逆初始置换（IP逆）。无论是加密还是解密，数据都经过相同的逻辑结构进行处理。

（2）密钥使用：

两个过程都使用相同的密钥生成机制，即从10位密钥生成两个子密钥K1和K2，密钥生成的算法相同。

（3）数据处理方法：

两者都采用扩展置换、S-Box替换和P-Box置换等方法来处理数据，只是在轮函数的具体实现中对密钥的使用顺序有所不同。

（4）对称性：

S-DES作为对称加密算法，加密和解密使用相同的密钥，从而保持了算法的对称性。

3.2不同点

（1）密钥顺序：

加密过程中，子密钥K1在第一轮使用，K2在第二轮使用；而解密过程中则相反，首先使用K2，接着使用K1。

（2）输入数据：

加密过程的输入是明文，而解密过程的输入是密文。输出结果分别是密文和明文。

（3）目的和方向：

加密的目的是将明文转换为密文以保护数据，而解密的目的是将密文恢复为原始明文。因此，它们的功能和最终目标截然不同。

（4）轮函数的执行顺序：

加密和解密使用相同的轮函数，但在执行顺序和具体处理的上下文中，存在关键的差异。这影响了如何将输入数据与密钥结合的过程。

**第二关：交叉测试**

1. **我方加密结果，如图1所示**

****

图1：我方加密结果

1. **对方加密结果，如图2所示**

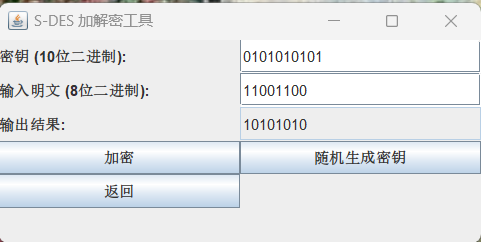
****

图2：对方加密结果

1. **我方解密结果，如图3所示**

****

图3：我方解密结果

1. **对方解密结果，如图4所示**

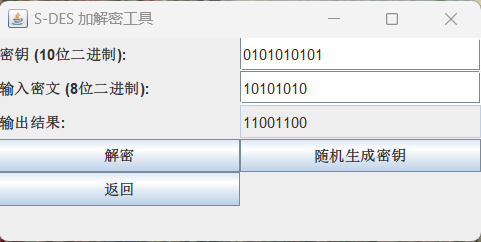
****

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：

·A、B两组同学编写的程序对相同的明文 PPP 进行加密时，能否生成相同的密文 CCC。

·B组接收到A组加密后的密文 CCC 后，能否通过自身的程序和密钥 KKK 成功解密出与A组相同的明文 PPP。

**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。建议对照S-DES算法的具体步骤，尤其是初始置换（IP）、扩展置换（EP）、S盒替换等部分，确保算法实现一致。

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-DES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**

在加密算法中，输入可以是ASCII字符串，这些字符串首先被转换为对应的二进制数进行加密，输出的结果可能仍然是ASCII格式，但由于加密后的数据通常是非可读的，因此会出现乱码现象。

（1） ASCII编码加密的原理

加密算法通常将明文（ASCII字符串）进行分组并按位或字节操作。以下是加密流程中使用ASCII编码的常见步骤：

·将字符串转为二进制：将输入的每个ASCII字符转换为8位二进制（字节）表示。例如，字符'A'的ASCII码是65，它的二进制表示是01000001。

·加密操作：加密算法会对这些二进制数进行位操作、轮函数、替换（如S盒替换）等。对于分组加密算法（如S-DES、DES），字符串会被分成固定长度的块（例如64位或8字节），逐块进行加密。

·加密后的输出：加密后的二进制数再转换为ASCII字符输出。如果这些二进制数对应的ASCII字符不是可读的字符（例如0x00到0x1F之间的控制字符），输出的内容就会是乱码。对于加密输出，如果需要保持可读性，通常会对结果进行编码（如Base64编码）处理。

（2） ASCII字符串的分组加密

在加密算法中，输入数据通常会被分成固定长度的分组。对于分组长度小于数据块长度的字符串（例如S-DES的64位块），会使用\*\*填充（padding）\*\*策略，常见的填充方式是PKCS#5或PKCS#7填充，将剩余的字节填充到分组大小。

对于ASCII字符串加密，可以使用如下流程：

·明文分组：假设每个块是64位（8字节），那么"Hello"这样的字符串需要被分组为[H, e, l, l, o, (padding)]，并转换为对应的二进制。

·加密分组：每个分组分别进行加密，得到加密后的二进制。

·输出字符：将加密后的二进制转换为对应的ASCII字符。

（3）乱码输出的原因

加密后的输出结果大多为随机的字节流，这些字节流直接作为ASCII字符时，可能会映射到不可读或控制字符，因此呈现乱码。这是因为加密的目的是让加密后的数据看起来像是无序的随机数据，而不是可读的明文。常见解决方案有：

Base64编码：将加密后的二进制数据转换为可读的Base64格式，常用于传输加密数据。

Hex编码：将每个字节转换为两位的十六进制字符串，便于显示和传输。

**2、加密测试**

（1）加密的一般步骤

ASCII加密测试步骤,以下是如何使用ASCII编码进行加密测试的步骤：

·准备明文：选择一段ASCII字符串作为明文输入。

·转换为二进制：将ASCII字符串转换为对应的二进制编码。

·加密：对二进制数据应用加密算法，例如S-DES。

·输出结果：将加密后的结果再转换为ASCII字符。

·检查结果：观察加密结果，如果出现乱码，意味着加密后的二进制数据无法直接映射为可读的字符。

·解密测试：用相同的密钥进行解密，验证解密后的结果是否能还原成原始的明文。

（2）成果展示（如图5所示）



图5：S-DES的ASCII编码的加密测试

**3、解密测试**

（1）解密的基本原理

解密是加密的逆过程，它使用与加密相同的密钥和加密算法的逆步骤。具体流程包括：

·读取密文（可能是ASCII编码的乱码）。

·使用正确的密钥执行加密算法的逆操作。

·最终将得到的二进制数据还原为ASCII明文。

对称加密算法如S-DES，使用相同的密钥进行加密和解密，因此如果密钥错误，解密后的数据将无法恢复为正确的明文。

（2）ASCII编码在解密中的作用

在加密解密测试中，输入和输出是以ASCII编码表示的。解密过程中的重点在于将加密后的乱码正确解密，并将其转换为ASCII字符。

ASCII编码的作用：ASCII是一个字符编码标准，规定了从二进制（0和1的组合）到可读字符（如字母、数字和符号）的映射。解密后的数据是二进制数，这些二进制数将需要转换回对应的ASCII字符。

乱码现象：加密后的输出通常是随机的二进制流，当直接转为ASCII字符时，由于二进制数可能不对应有效的ASCII字符，结果常常显示为乱码。在解密过程中，需要通过正确的密钥恢复出原始的二进制流，再转换为正确的ASCII编码，才能得到可读的明文。

（3）一般步骤

解密测试主要围绕如何将加密后的ASCII编码（可能是乱码）正确解密并转换回明文.以下是解密操作的大概步骤：

·密文输入：输入的密文可以是ASCII编码的字符串，可能含有乱码。如果是经过Base64或Hex编码处理的密文，则需要先将其转换回二进制格式。

·解密过程：使用正确的密钥，通过加密算法的逆操作将密文转换回原始的二进制明文表示。

·二进制到ASCII转换：解密后，将二进制数据按每8位（1字节）切分，并将其转换回对应的ASCII字符，得到解密后的明文。

（4）结果验证

正确解密的结果：如果使用正确的密钥解密，解密后的二进制数据应能完全恢复为原始明文的ASCII编码。

解密失败的表现：若使用错误的密钥解密，输出的二进制数据可能无法转换为可读的明文字符，表现为乱码或不可见字符。

（5）成果展示（如图6所示）



图6：S-DES的ASCII编码的解密测试

**4、总结**

加密时的乱码现象是正常的，因为加密后的二进制数据通常映射到不可读的字符。

解密测试的关键在于密钥的正确性。只有使用正确的密钥才能将密文还原为明文。

加密使用的是密钥的生成过程，解密是加密过程的逆操作，依赖于密钥的正确性才能解密成功。

加密测试侧重于将明文字符转换为乱码式的密文，解密测试则验证是否能通过密钥正确还原明文。

1. 加解密测试的相同点

输入输出格式：都使用 ASCII 编码 的字符串，输入时将其转换为二进制进行处理。

处理流程：二者都依赖相同的S-DES算法，涉及分组操作、置换、S盒替换、密钥生成等。

二进制转换：无论加密还是解密，最终都是二进制数据的转换：加密时将明文字符转换为二进制；解密时还原二进制为明文字符。

1. 加解密测试的不同点

过程方向：

·加密：从明文到密文，将可读的ASCII字符转为随机的二进制（密文）。

·解密：从密文到明文，恢复原始ASCII字符。

结果可读性：

·加密结果：通常是乱码，因为随机二进制无法映射为可读字符。

·解密结果：如果密钥正确，应还原为可读的明文。

密钥使用：

·加密使用的是密钥的生成过程，解密是加密过程的逆操作，依赖于密钥的正确性才能解密成功。

**第四关：暴力测试**

**1. 单线程 vs 多线程的基本原理**

S-DES的密钥长度为10位（2^10 = 1024），因此有1024种可能的密钥。暴力破解的核心原理是枚举每一种可能的密钥，尝试对明文加密，看加密结果是否与已知的密文相匹配。如果匹配，则该密钥就是正确的密钥。

单线程破解：使用一个线程顺序地遍历所有可能的密钥，尝试每个密钥对明文进行加密，直到找到匹配的密钥。所有可能的密钥是按顺序测试的，这种方法易于实现，但效率较低。

多线程破解：将所有的密钥空间分割为若干部分，每个线程负责其中一部分密钥的测试，以加快暴力破解的过程。多个线程并行处理。这样可以显著减少暴力破解的总时间，但线程数量受限于系统的核心数量，线程之间的调度开销也需要考虑。

**2. 实现步骤**

·获取明文和密文对：找到至少一个明文和密文对。

·编写S-DES加密函数：实现S-DES加密算法，用于在破解过程中对每个密钥进行验证。

·单线程暴力破解：编写单线程版本的破解算法，遍历所有可能的密钥。

·多线程暴力破解：编写多线程版本，将密钥空间划分给多个线程并行处理。

·记录时间戳：在开始破解和完成破解时记录时间，以计算破解时间。

**3、时间戳记录与展示**

使用time库记录开始和结束的时间戳，显示破解耗时。同时，可以通过屏幕录制软件（如OBS）记录单线程和多线程破解的时间对比，生成图片进行展示。通过并行处理提升效率，在多核CPU上多线程通常比单线程快数倍。

**4、效果展示**

·单线程破解（如图7所示）：

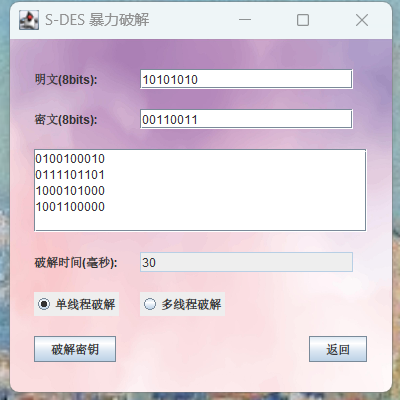
****

图7：单线程暴力破解

·多线程破解（如图8所示）：

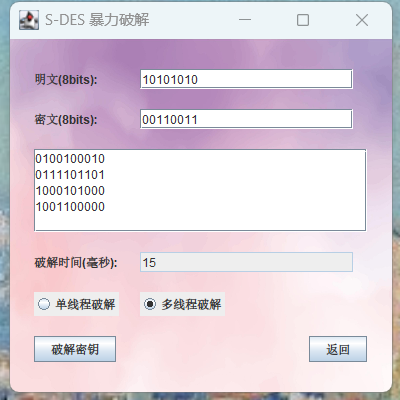
****

图8：多线程暴力破解

**第五关：封闭测试**

**1、问题一：**对于一个随机选择的一个明密文对，是不是有不止一个密钥Key？

对于一个随机选择的明密文对，可能存在不止一个密钥Key能够成功解密出正确的明文。这种现象被称为密钥冲突，即不同的密钥可能在特定的明密文对上产生相同的加密或解密结果。原因如下：

S-DES的简化设计：由于S-DES是一种教学算法，密钥空间较小（8位密钥），仅有256种可能的密钥。这种密钥空间较小的设计增加了发生密钥冲突的概率。

有限的加密轮数：S-DES的加密过程轮数较少（通常为两轮），导致加密过程的复杂性不如真正的DES或其他现代加密算法那么高。这可能导致不同的密钥在某些特定的明密文对下产生相同的加密结果。

1. **问题二：**对应明文空间的任意明文分组[P{n}](" \l ")，是否会出现选择不同的密钥加密得到相同密文[C{n](" \l ")}的情况？

会存在这种情况，这种现象被称为密钥碰撞，在一些加密算法中尤其常见，特别是在密钥空间较小或加密过程简化的情况下。原因如下：

（1）密钥空间的限制

在S-DES等简化算法中，密钥空间有限（如S-DES只有256个密钥），这大大增加了发生密钥碰撞的概率。随着不同密钥的数量接近密文的数量，碰撞变得更加可能。

（2）映射冲突

S-DES作为对称加密算法，将明文映射到密文是一种确定性函数，即每个明文和密钥组合将唯一生成一个密文。然而，密钥空间的有限性可能导致多个密钥 KiK\_iKi​ 和 KjK\_jKj​ 对相同的明文 PnP\_nPn​ 进行加密时，结果映射到相同的密文 CnC\_nCn​。这种现象在有限密钥空间中几乎不可避免。

（3）S-DES的简化结构

S-DES的结构比标准DES简化了许多，它的加密轮数较少，导致加密过程的混淆和扩散效果不如标准DES那样复杂。这种简化可能增加了不同密钥对产生相同密文的机会。

（4）实际加密算法中的碰撞现象

在现代的加密算法中，如DES或AES，密钥空间非常大，密钥碰撞的概率非常低。相比之下，S-DES由于其密钥空间仅有256种可能，更容易发生密钥碰撞。