线性攻击小结：

线性密码分析是对称加密算法的一种有效攻击方式，它利用加密算法中的线性关系，通过构建线性逼近表达式来推测密钥。攻击者需收集大量明密文对，计算这些线性表达式的实际概率，并与随机猜测的概率进行比较，以评估其有效性。

然而，对DES算法实施线性攻击面临一些挑战：一是获取足够多的明密文对非常困难，二是DES算法的密钥空间巨大（2^56种可能），需要极大量的独立线性表达式和统计分析才能逐步缩小密钥搜索范围，这导致计算密集且耗时极长。尽管线性密码分析在理论上为破解DES提供了路径，但在实践中，其成功实施受限于数据获取难度、计算资源限制和时间成本。现代加密算法如AES在设计时已增强了抵抗此类攻击的能力。

在代码实现模拟攻击中，我通过随机生成明文和密钥来获取一些明密文对，并且缩小密钥空间来模拟攻击。但是没有大量的明密文对，且缩小了密钥空间，导致找到的最佳密钥候选得分较低，无法找到正确的密钥。虽然无法实现真正的对DES算法进行线性攻击，但是这个实验使我了解了对DES算法实施线性攻击的原理与实现的可能性。

（各代码作用）

（一）des\_create.py生成明密文对

·主要用于为DES的线性攻击提供明密文对以便进行分析

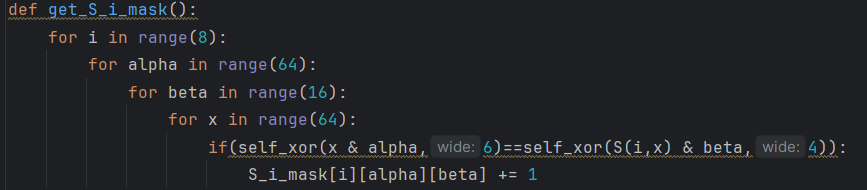
（二）DES\_s\_linear.py 针对数据加密标准（DES）算法中的S盒进行线性分析

·主要用于构建和分析 S 盒的线性逼近表，这是线性密码分析的核心步骤之一。

·核心代码分析：

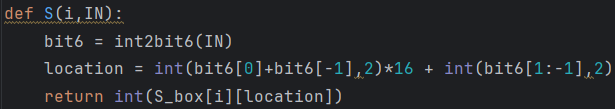
①get\_S\_i\_mask 函数

这是构建线性逼近表的核心函数。通过遍历所有可能的输入掩码（alpha）和输出掩码（beta），并统计满足给定线性关系的输入输出对的数量。对于每一个alpha-beta组合，如果某个输入输出对符合线性表达式，则计数器加1；否则减1。最终结果存储在 S\_i\_mask 中。通过该函数，我们可以确定哪些alpha-beta组合具有较高的相关性，从而找到最佳的线性逼近表达式。



②S 函数

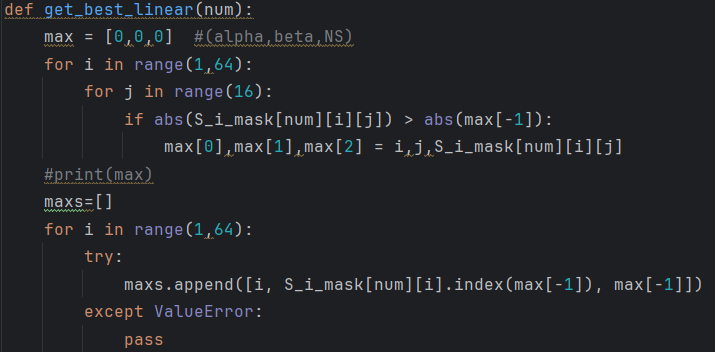
模拟单个S盒的行为，将6位输入转换为4位输出。作为基础组件，该函数确保了线性逼近表能够准确反映S盒的实际行为，进而保证了所计算的逼近方程的有效性。



③get\_best\_linear 函数

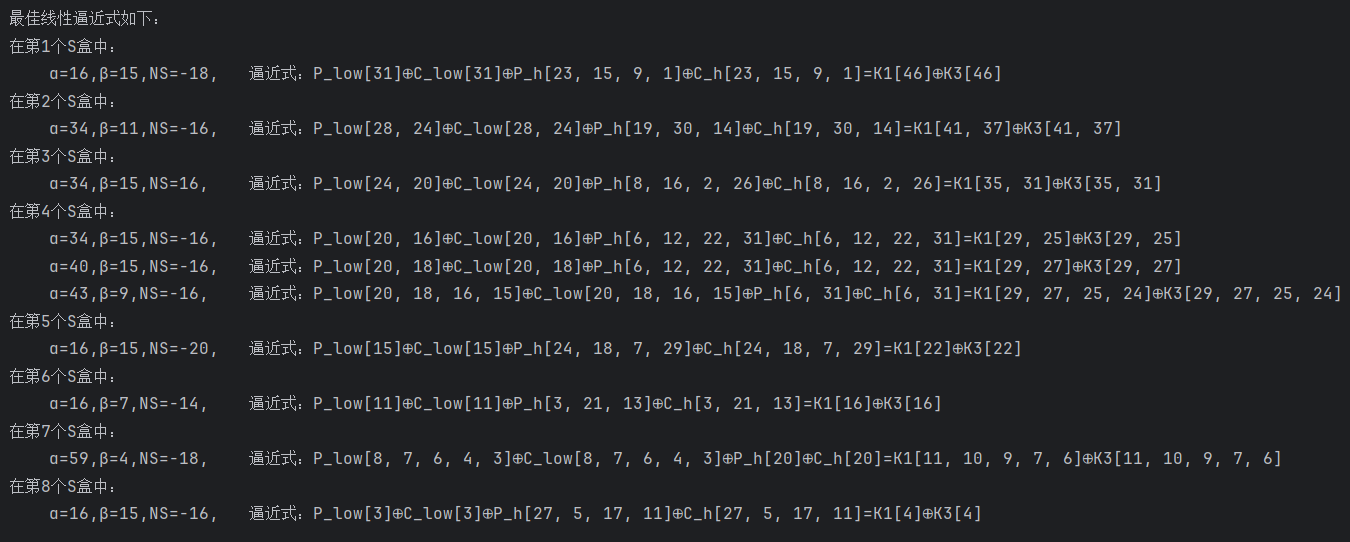
基于 S\_i\_mask 的结果，找出具有最大绝对NS（线性逼近偏差）值的alpha和beta组合，并根据这些组合生成线性逼近表达式。而且进一步解析这些表达式，以确定它们与密钥位的关系。

这个函数是确定最佳逼近方程的关键步骤。它不仅识别出最有可能的线性关系，还解释了这些关系如何映射到加密算法的具体操作中，如E扩展置换和P盒置换等。





·代码运行结果



（三）des\_L.py

·根据DES\_s\_linear.py所得到的结果构建线性近似表，是攻击的第一步，提供了关于 DES S 盒行为的详细信息。

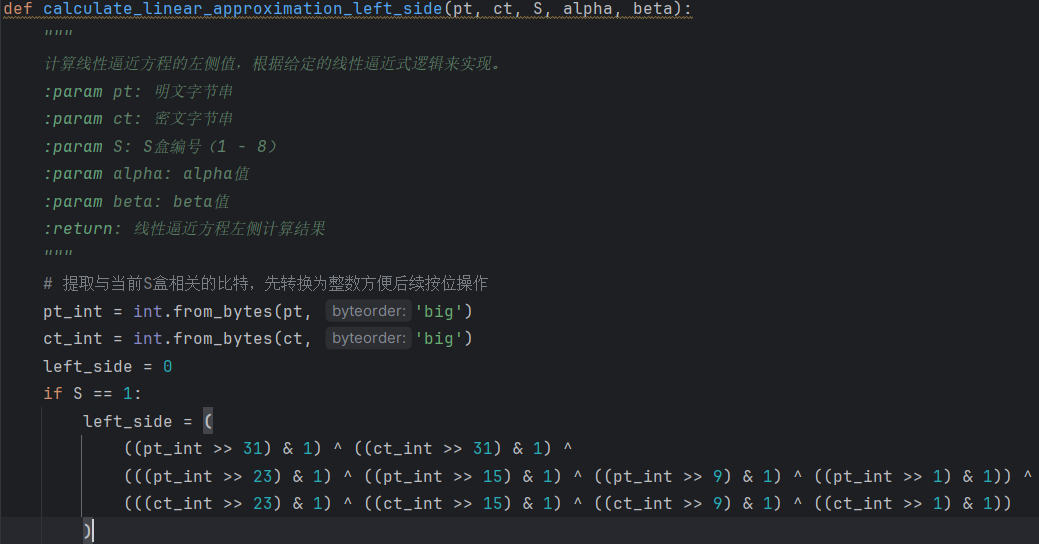
·遍历可能的密钥并对每个密钥进行评分，是攻击的关键，它量化了每个密钥候选与已知明文-密文对之间的匹配程度。

·寻找最佳密钥候选，是最终目标，即找到最有可能揭示真实密钥一部分的候选。

·核心代码分析：

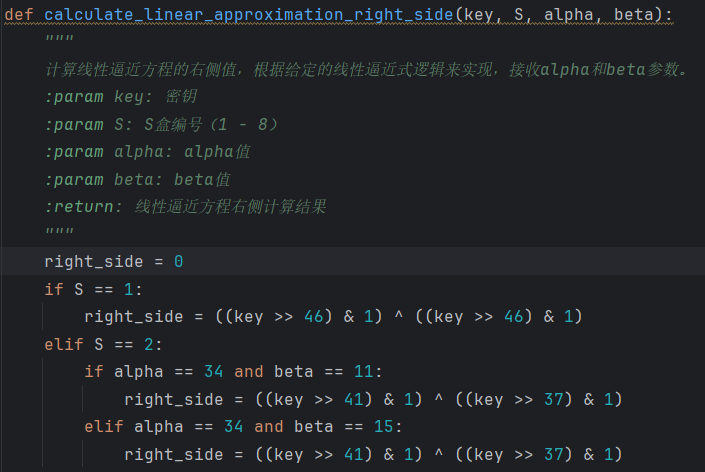
①calculate\_linear\_approximation\_left\_side 函数

根据DES\_s\_linear.py所得到的最佳线性逼近方程计算线性逼近方程的左侧值，模拟加密过程中S盒的行为，检查实际的明文-密文对是否符合预期的线性表达式。



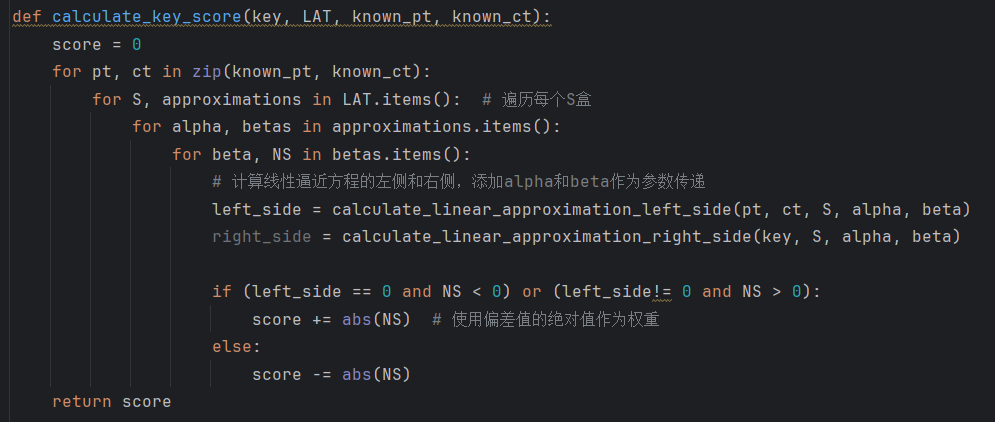
②calculate\_linear\_approximation\_right\_side函数

根据DES\_s\_linear.py所得到的最佳线性逼近方程计算线性逼近方程的左侧值，目的是生成一个假设的输出，以便与左侧值进行比较，验证密钥候选是否合理。



③calculate\_key\_score 函数

对每个明文-密文对，遍历所有S盒及其对应的alpha-beta组合。计算线性逼近方程的左侧和右侧，如果左侧和右侧的异或结果与NS值的符号相同，则增加得分；否则减少得分。使用偏差值的绝对值作为权重，确保高相关性的alpha-beta对具有更大的影响。



④find\_best\_key\_reduced 函数

遍历所有可能的前16位密钥候选（密钥空间过大，计算机运行速度无法短时间内完成全部密钥的遍历），调用 calculate\_key\_score 函数为每个候选评分，最终返回得分最高的密钥及其得分。

