**测试运行结果**

**第1关：基本测试**

1、启动 GUI：在项目目录运行：

python3 T1\_Main.py

2.打开「基本加/解密」选项卡：

在“明文/密文 (8-bit)”输入框输入一个 8-bit 位串（示例：10110101）。

在“密钥 (10-bit)”输入框输入一个 10-bit 位串（示例：1010000010）。

选择模式“加密”或“解密”，点击“执行”。

观察输出：

主输出框显示 8-bit 结果（密文或明文）。

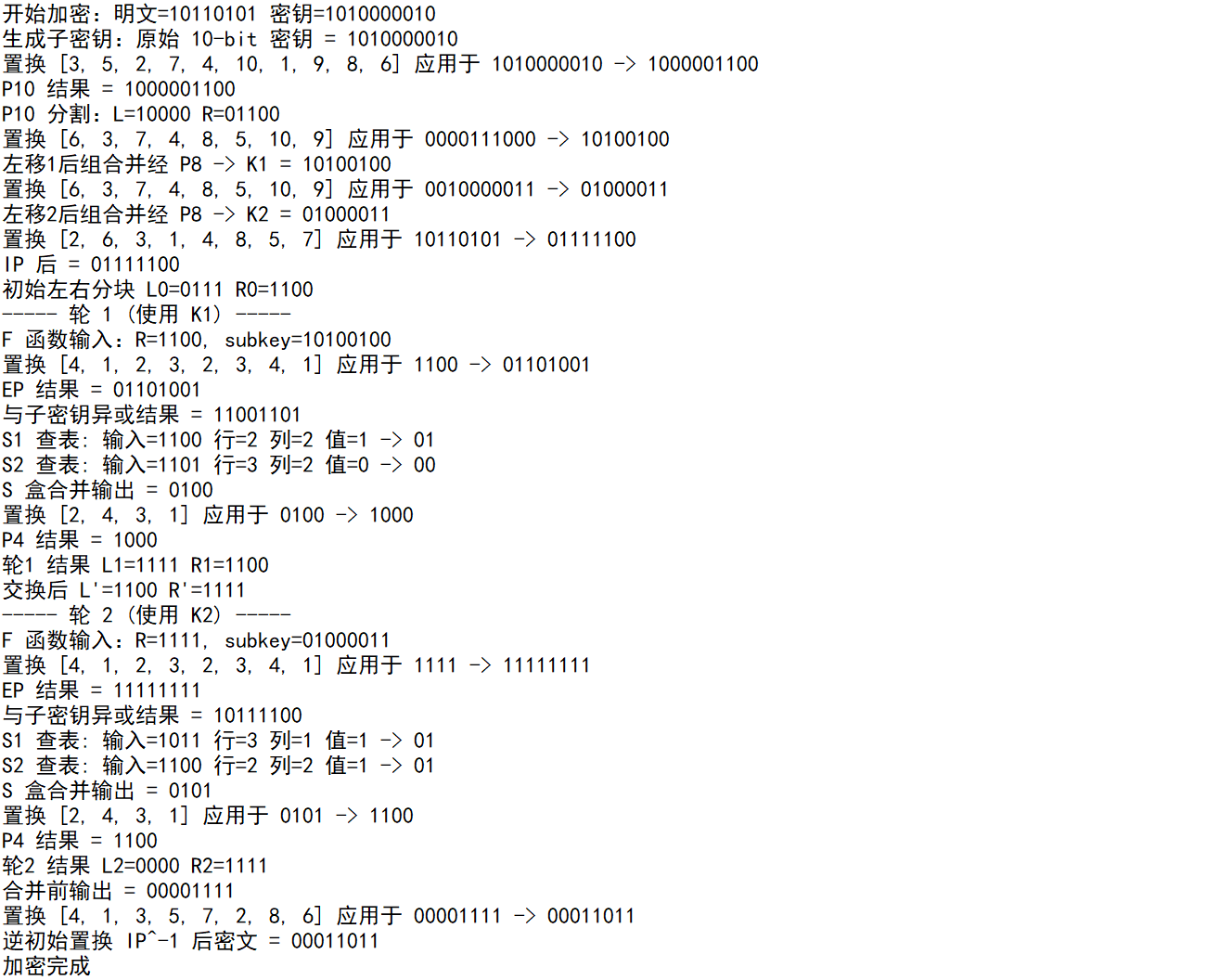
下方“详细过程日志”展示每一步置换、S-Box 查表、左右分割/交换等中间信息。

**示例一：**

加密：



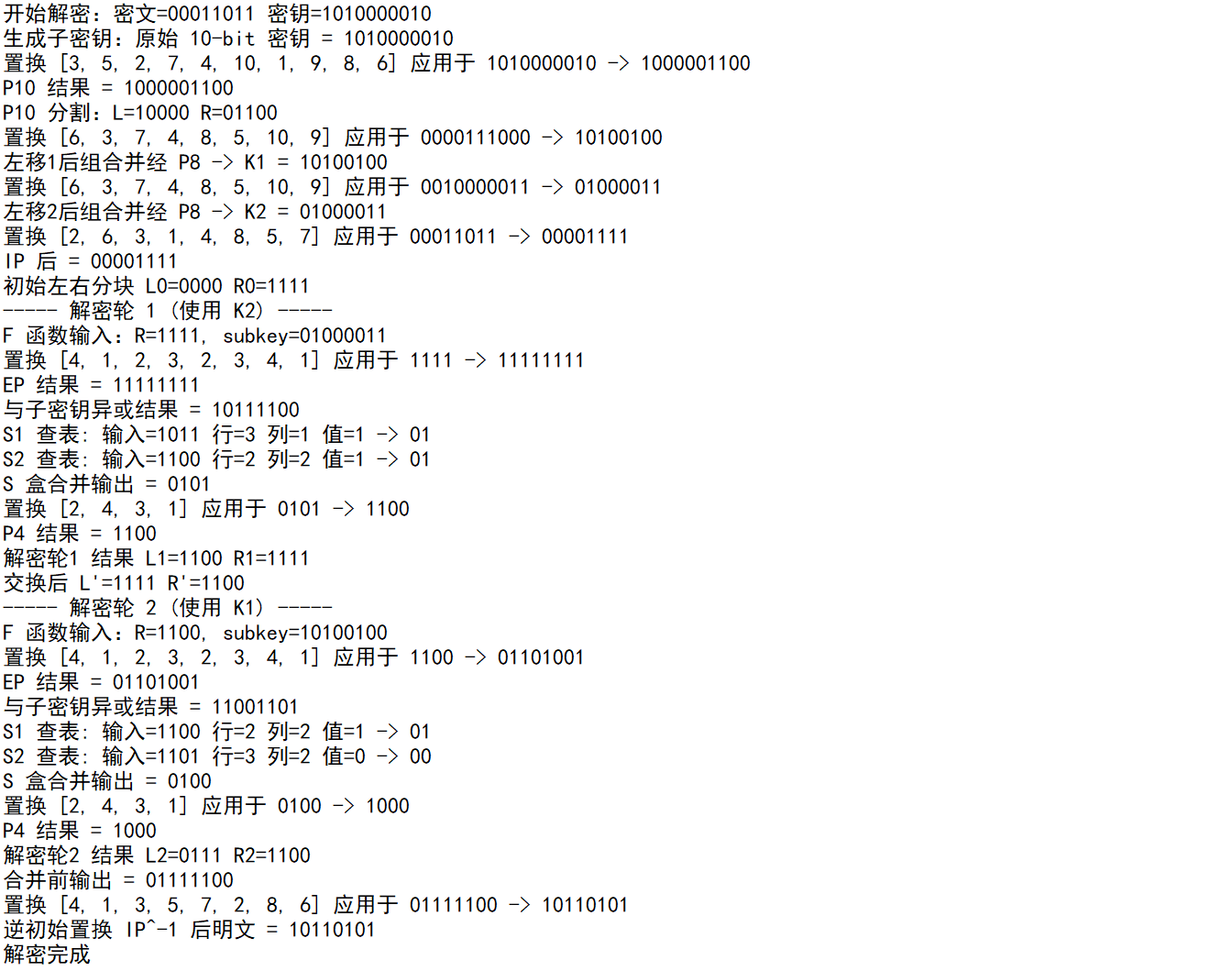
完整详细过程日志：



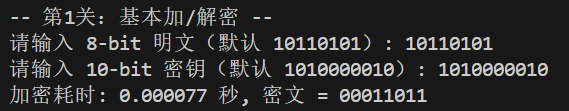
解密：



完整详细过程日志：



其他检验信息：







**示例二：**

加密：



解密：

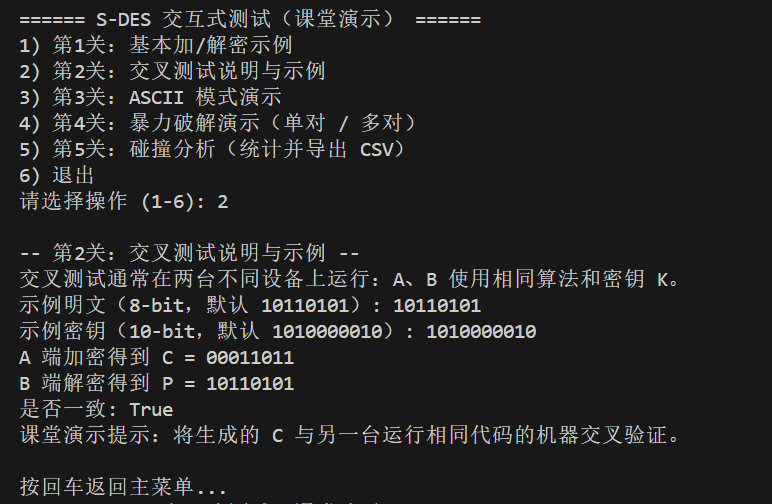


**第2关：交叉测试**

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(P-Box、S-Box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。

理想情况下：



组B使用相同明文和密钥的运行结果：



出现这种交叉结果不一致的现象时，最关键的结论是：两边实现某处基础定义不一致，导致生成的中间量（尤其是子密钥 K1/K2 或 IP 后的左右半部分）不同，从而输出不同密文。

**常见原因分析：**

原因 1：置换表（P-Box / S-Box / IP / EP 等）定义不同

若两组中有任意一个置换表与标准定义的索引顺序不同（哪怕只有一位次序错位），整个算法输出都会完全不同。

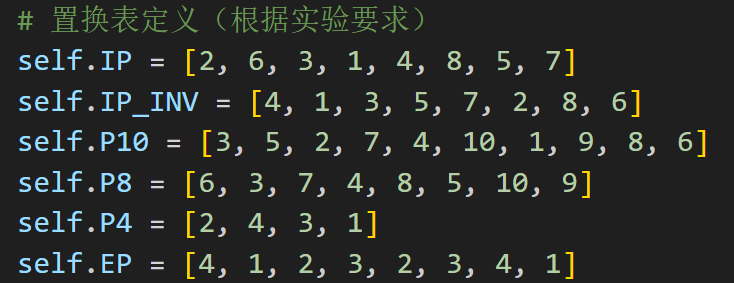
原因 2：S-Box（S0 / S1）定义或索引方式不同

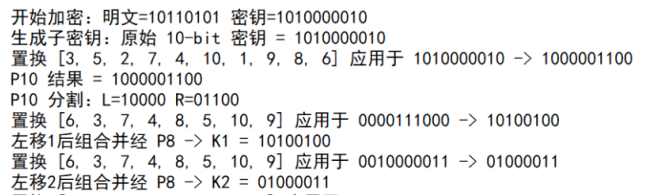
原因 3：密钥生成过程或左移逻辑不同

**验证方法：**

1. **比对子密钥 K1、K2（首要）**

子密钥生成依赖 P10、左移和 P8，任何问题最先体现在 K1/K2 上。让 B 组打印或提供他们的 K1 和 K2（10-bit 主密钥 1010000010 应生成固定的 K1/K2）。已知我们的K1、K2：





若 K1/K2 不同，则问题在 P10/P8 或左移实现上，优先核对 P10、P8 表与左移方向与步数。

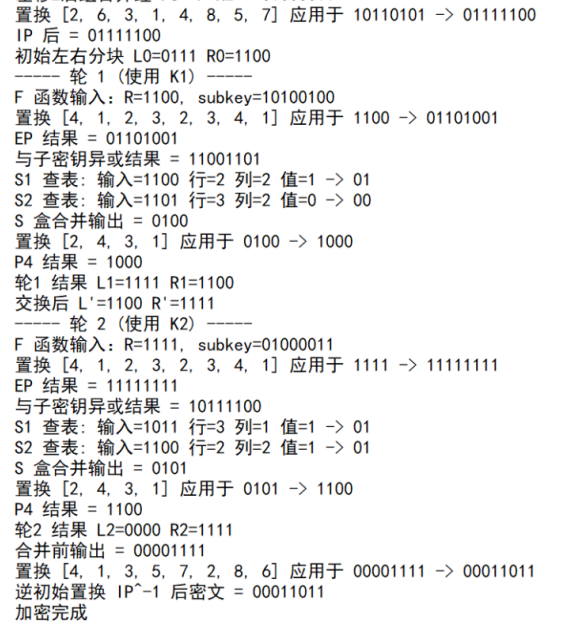
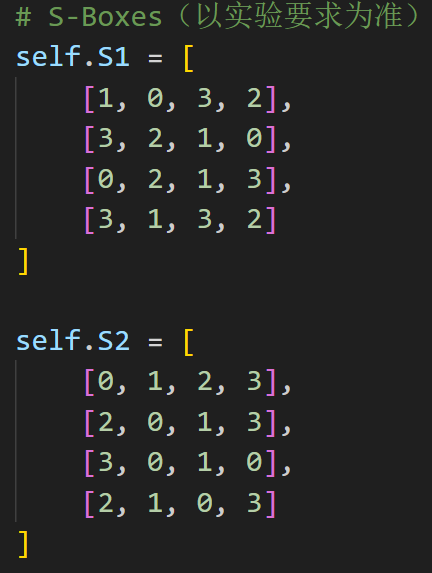
1. **对比 IP（初始置换）与 IP^-1 的实现**

若子密钥一致而输出仍不同，检查两边 IP 后的 L0、R0 是否一致（日志里有 IP 结果）。如果 IP 实现不同，后续所有轮函数都会不同。

1. **核对 S-Box 与索引规则**

若前两项均一致，再检查 S0/S1 表内容和行列索引方式（行 = 第1位+第4位，列 = 第2位+第3位）。S-Box 的一位错误会在轮函数输出处立刻显现，改变最终密文。

已知我方：



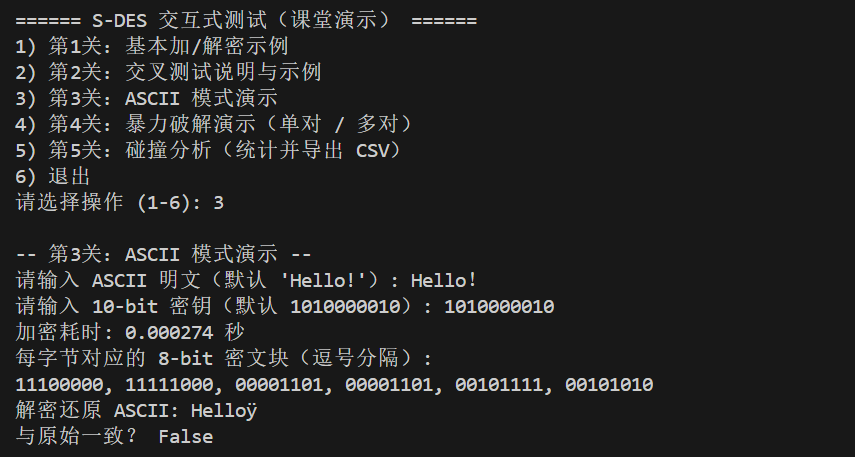
**第3关：扩展功能**

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1 Byte)，对应地输出也可以是ACII字符串(很可能是乱码)。

窗口正确输出：



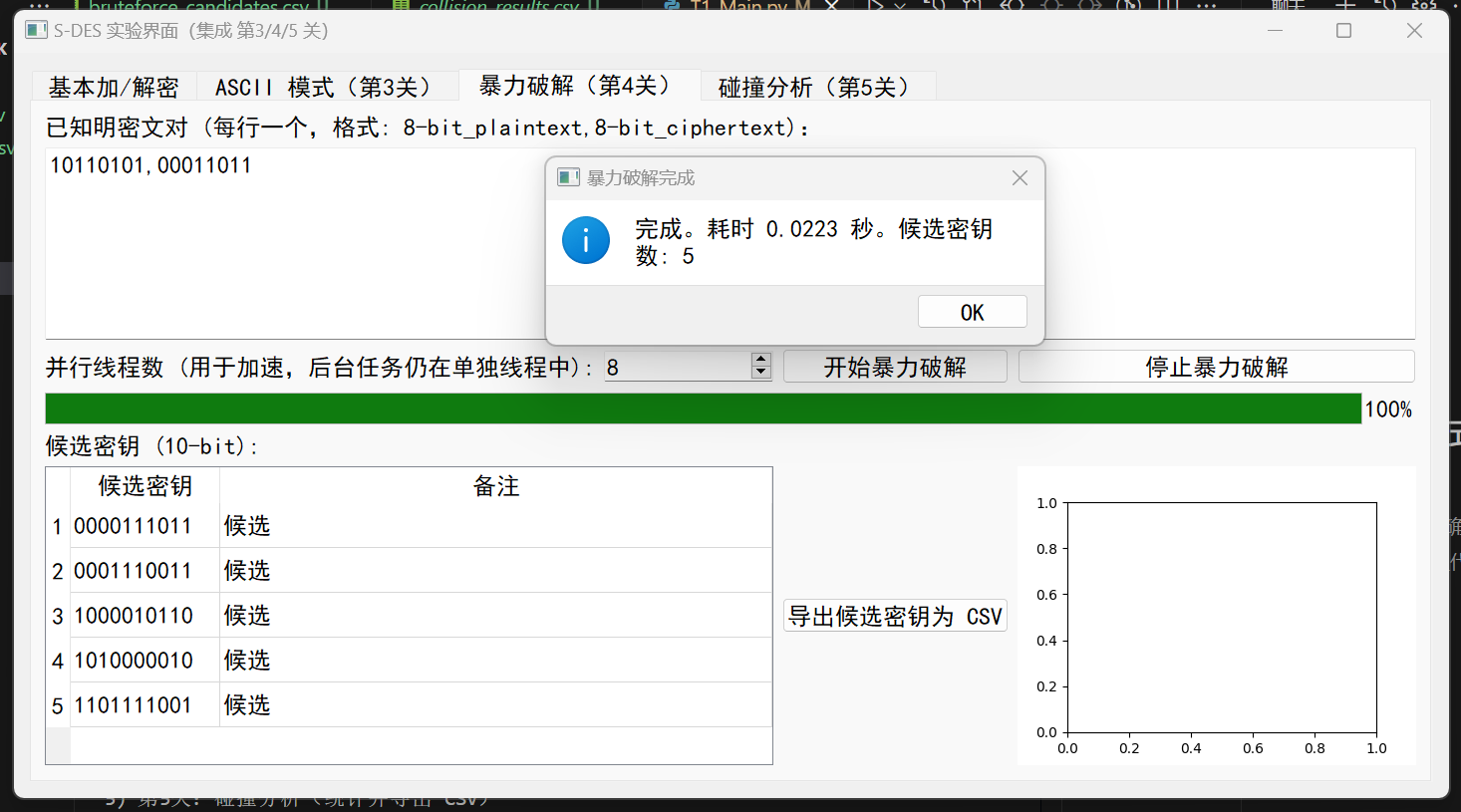
终端运行乱码：

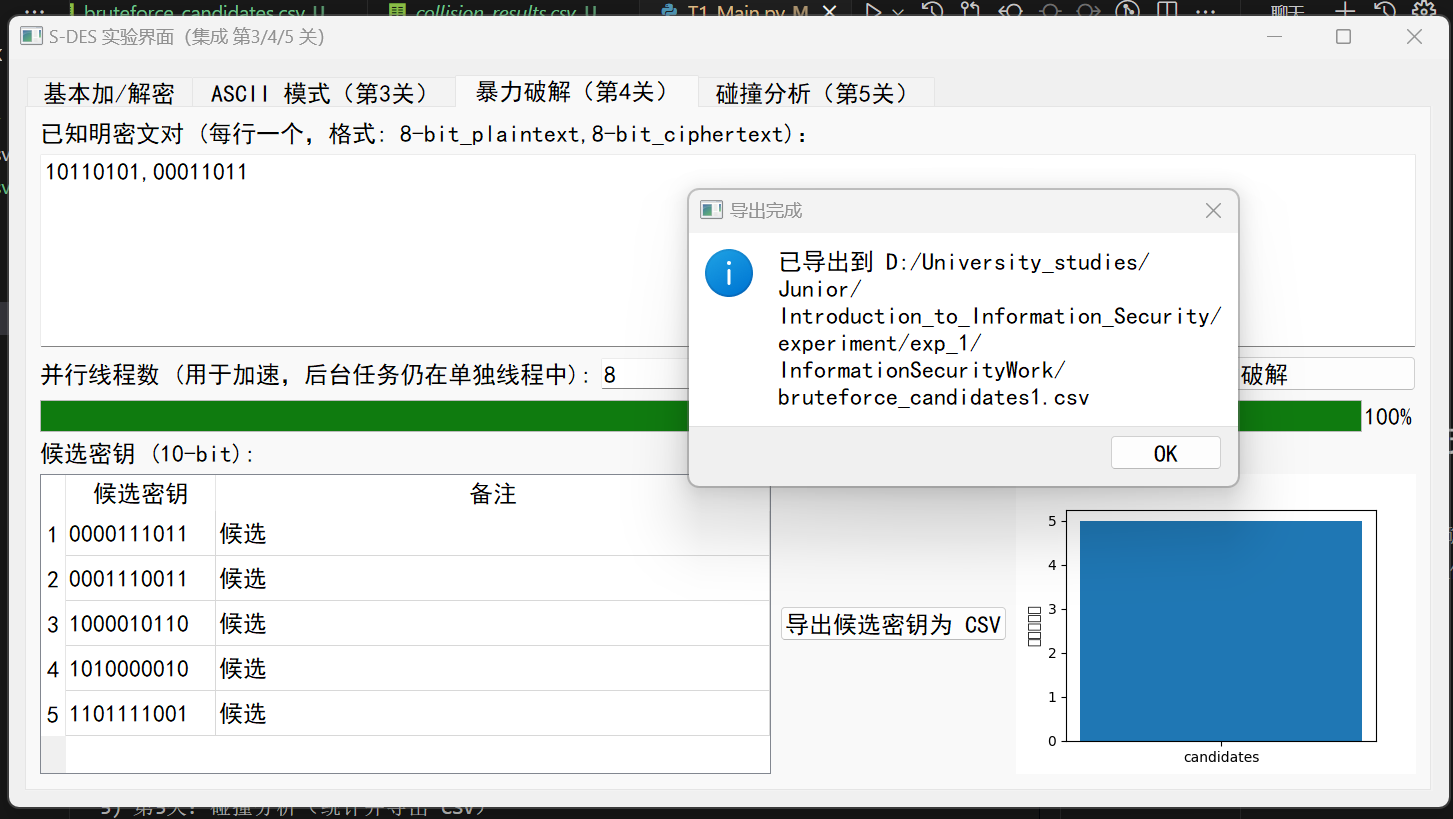


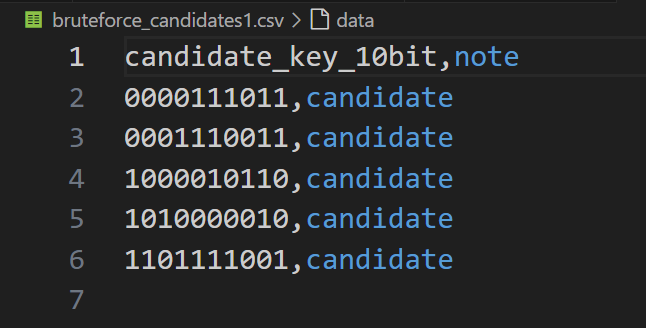
**第4关：暴力破解**

假设你找到了使用相同密钥的明、密文对(一个或多个)，请尝试使用暴力破解的方法找到正确的密钥Key。在编写程序时，你也可以考虑使用多线程的方式提升破解的效率。请设定时间戳，用视频或动图展示你在多长时间内完成了暴力破解。

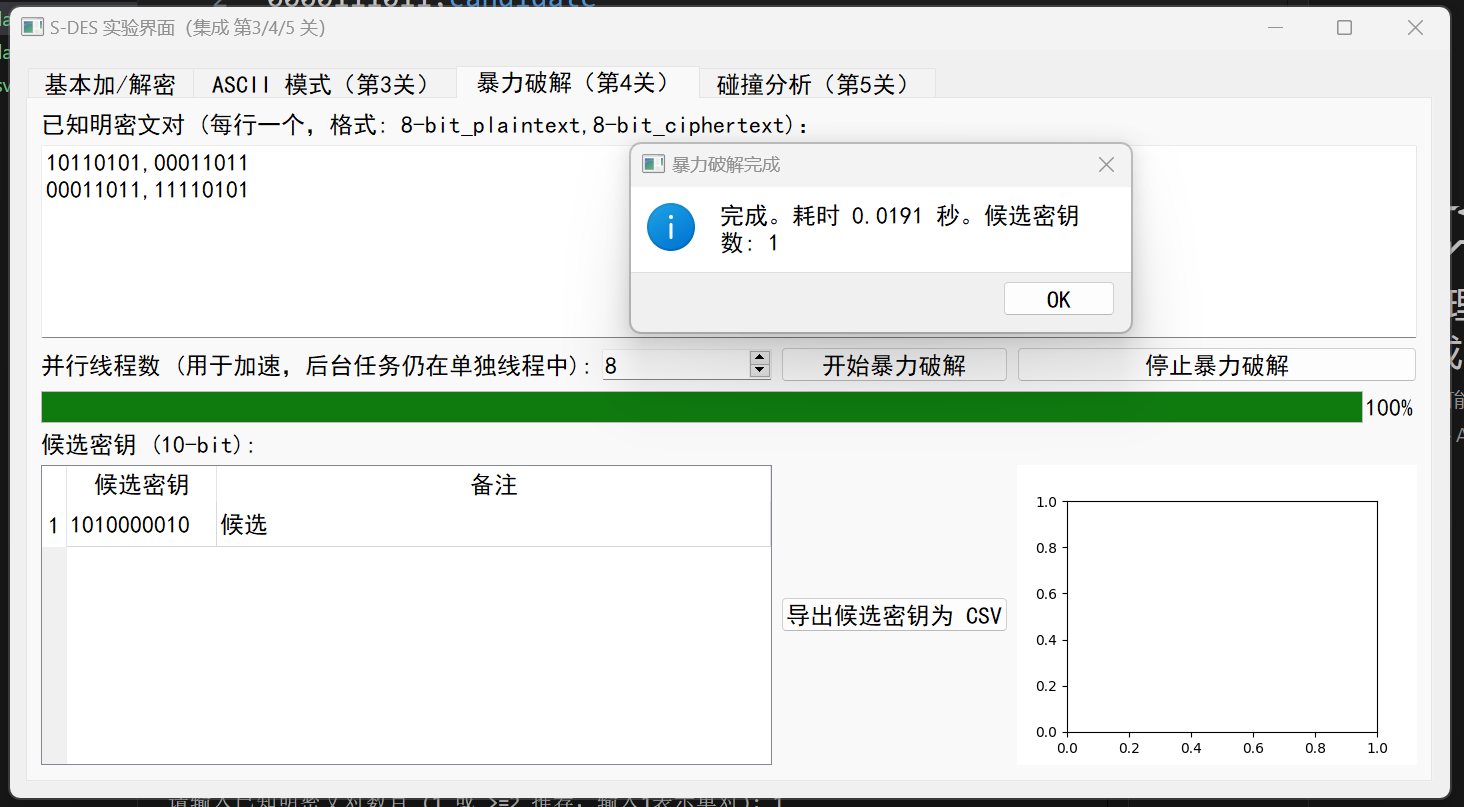
**1、一对明、密文对（以关卡一示例一为例）**

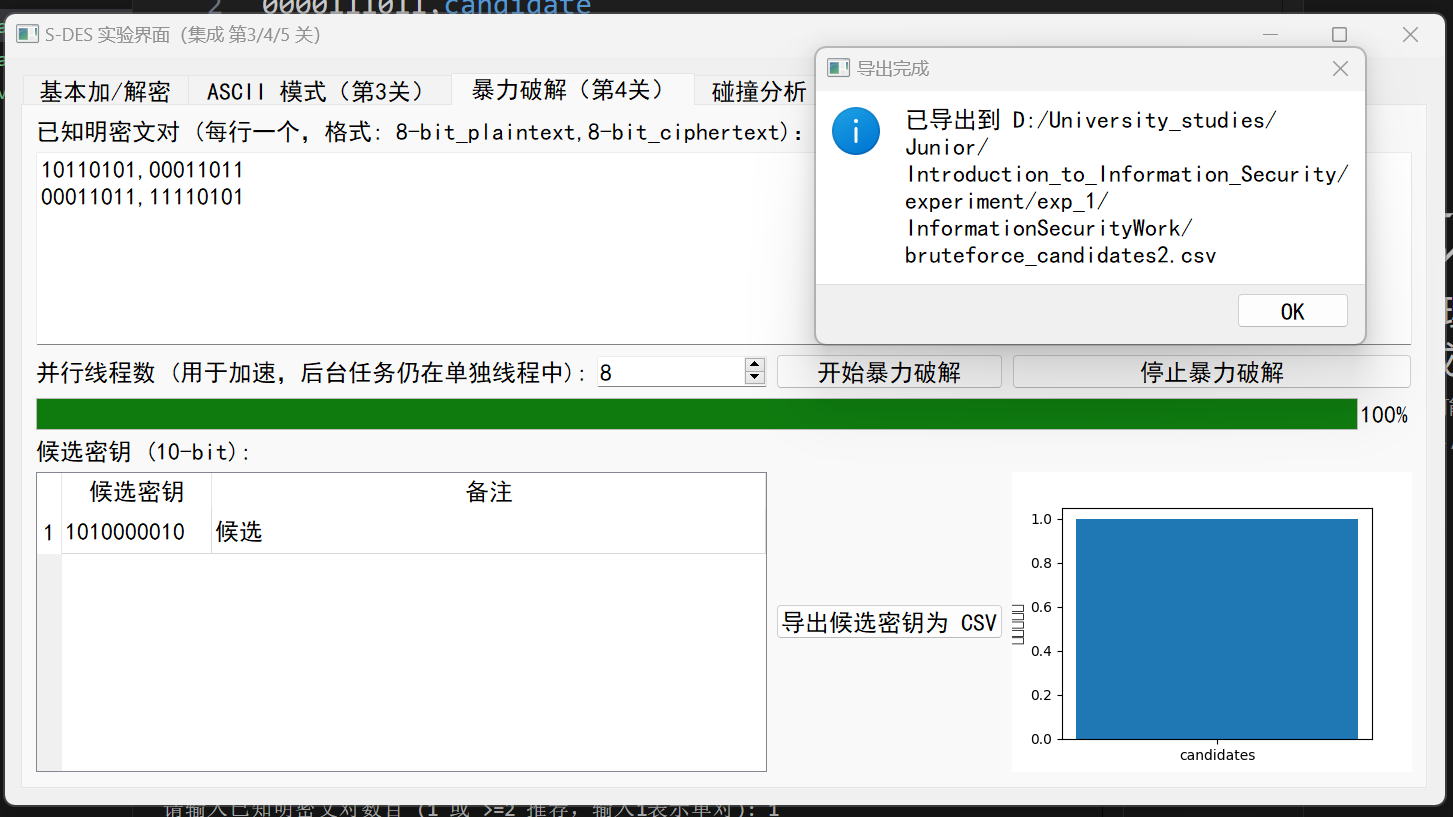


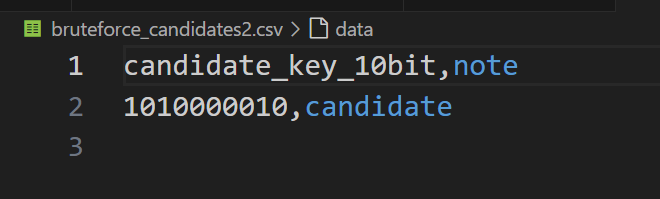




**2、两对明、密文对（以关卡一中两对示例为例）**







**第5关：封闭测试**

根据第4关的结果，进一步分析，对于你随机选择的一个明密文对，是不是有不止一个密钥Key？进一步扩展，对应明文空间任意给定的明文分组Pn，是否会出现选择不同的密钥Ki≠Kj加密得到相同密文Cn的情况？

答：随机选择一个明密文对，是有不止一个密钥Key。会出现选择不同的密钥Ki≠Kj加密得到相同密文Cn的情况。

以关卡一示例一为例：

