**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）。

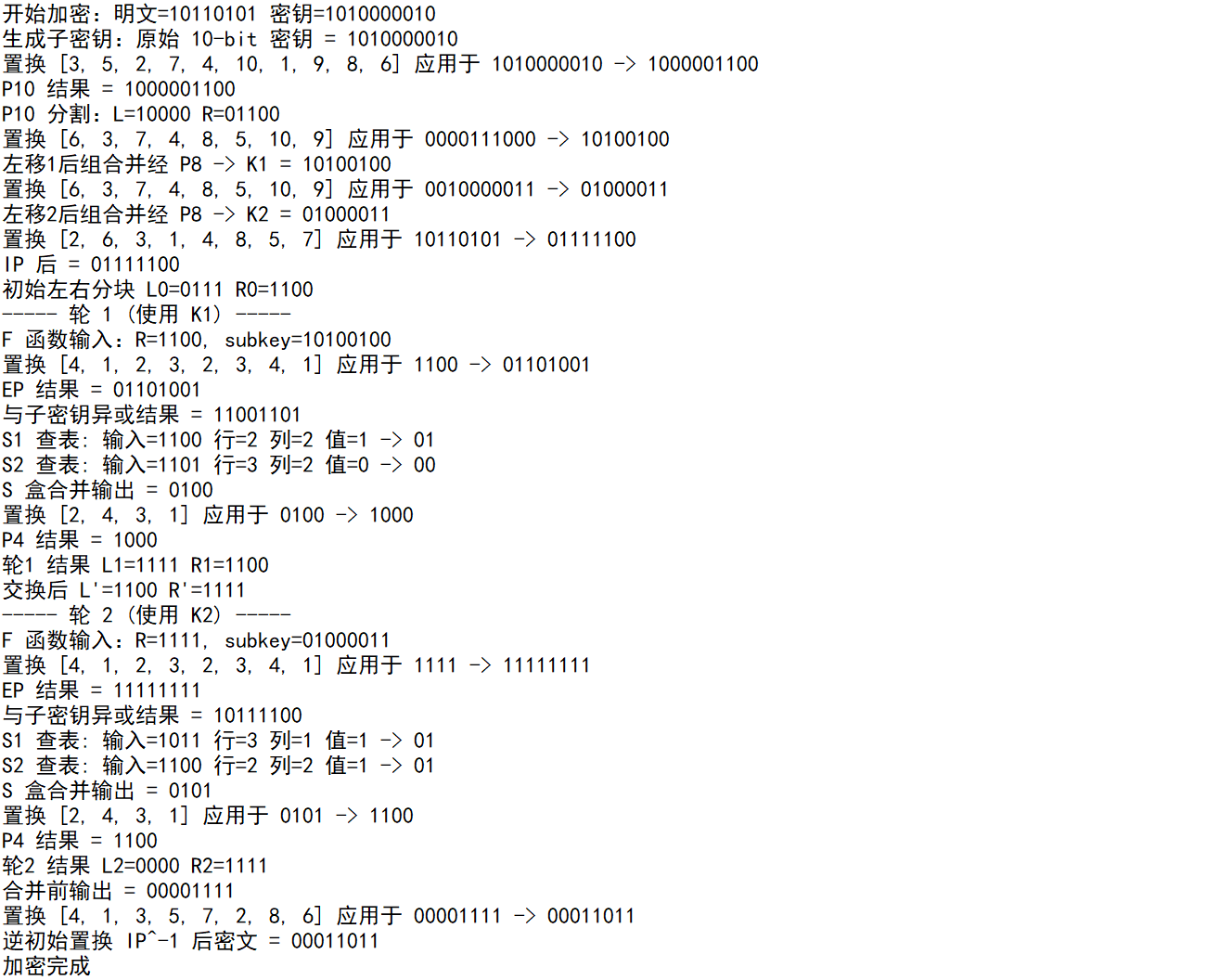
下方“详细过程日志”展示每一步置换、S-Box 查表、左右分割/交换等中间信息。

**示例一：**

加密：



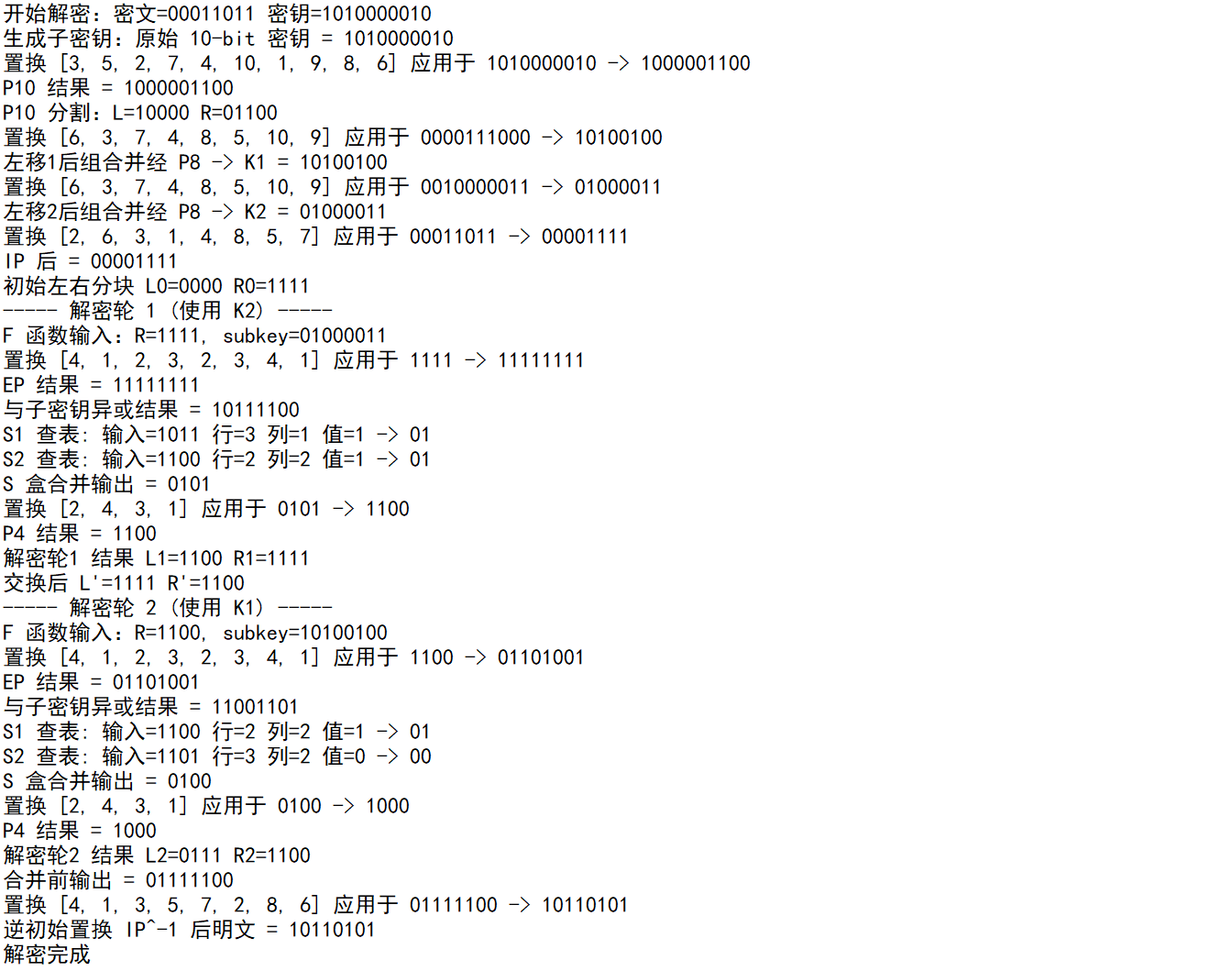
完整详细过程日志：



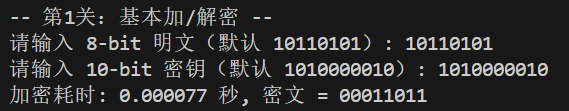
解密：



完整详细过程日志：



其他检验信息：







**示例二：**

加密：



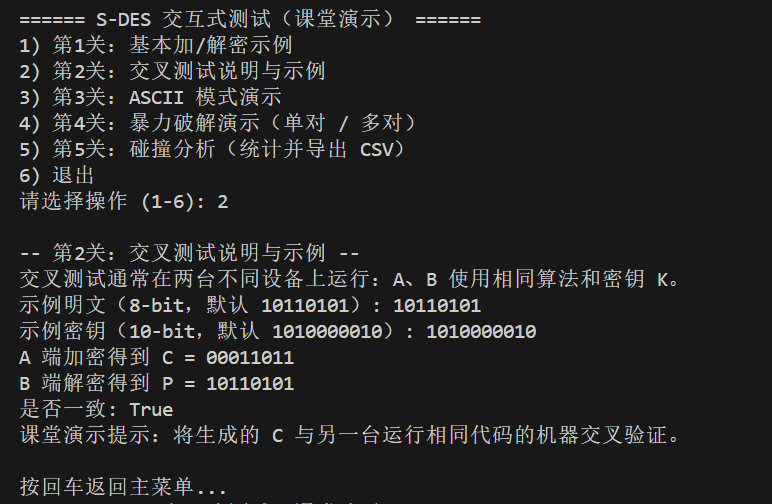
解密：



**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。



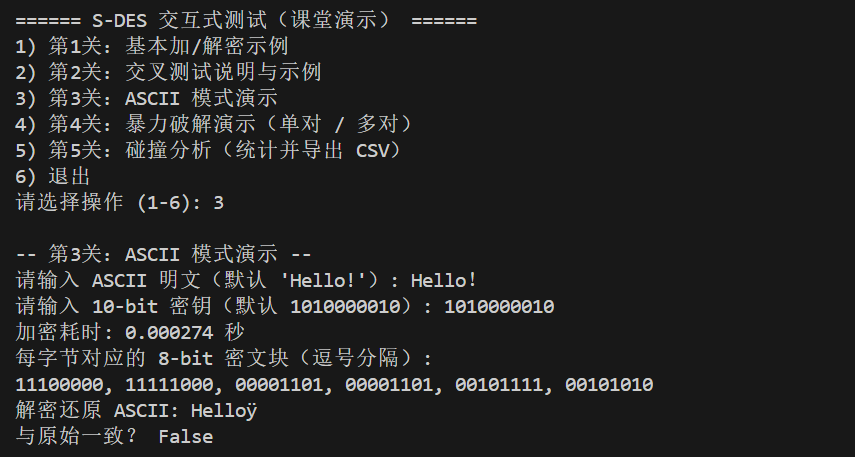
**第3关：扩展功能**

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：



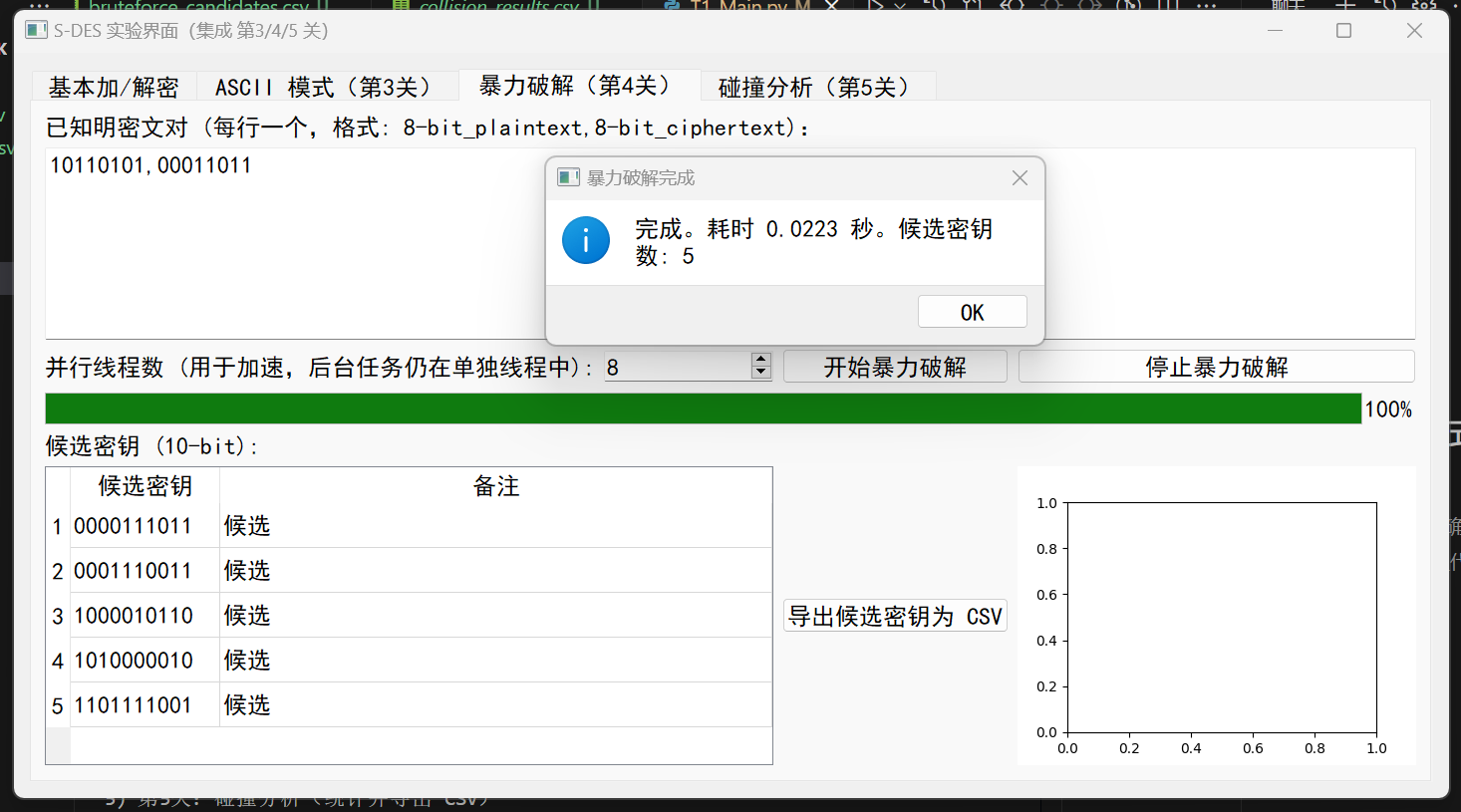
终端运行乱码：

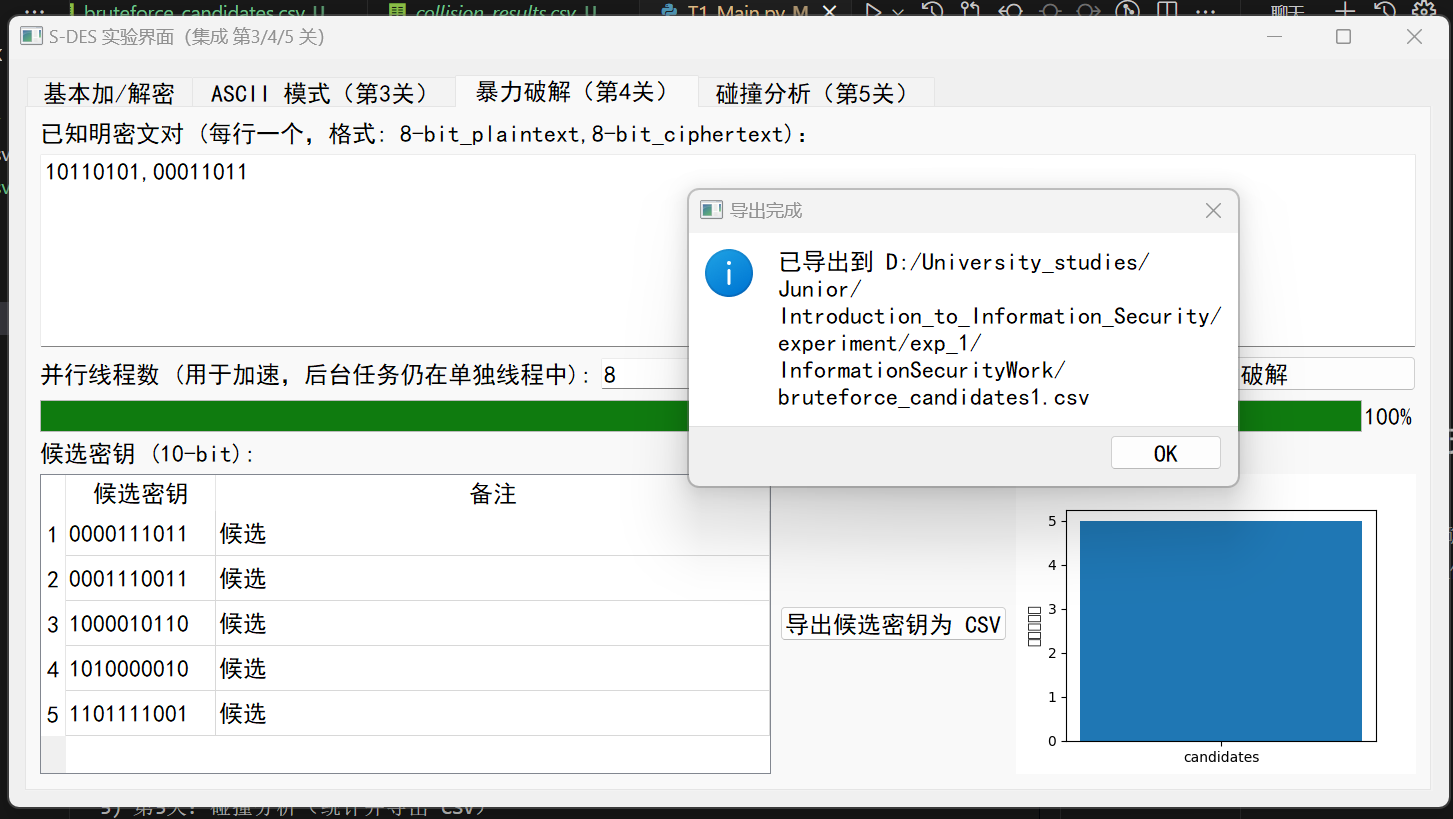


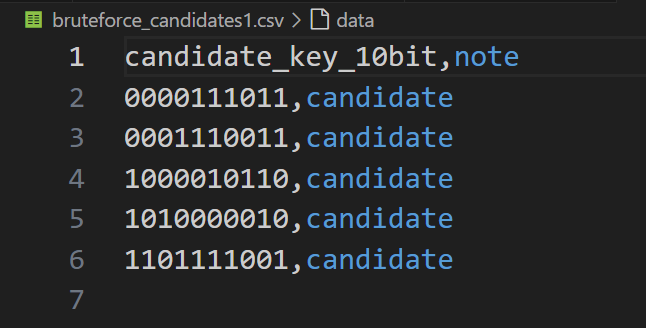
**第4关：暴力破解**

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。

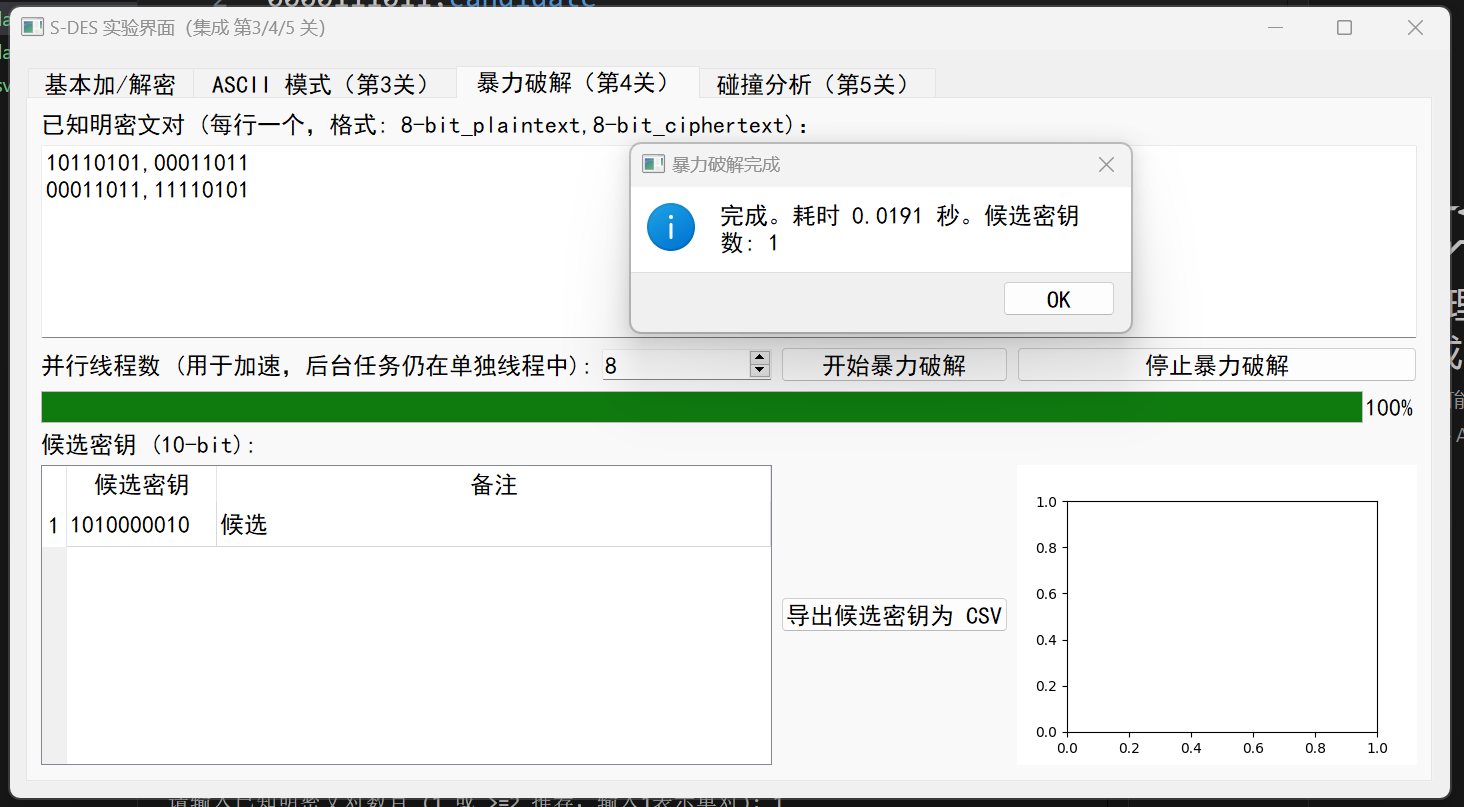
1. **一对明、密文对**（以关卡一示例一为例）

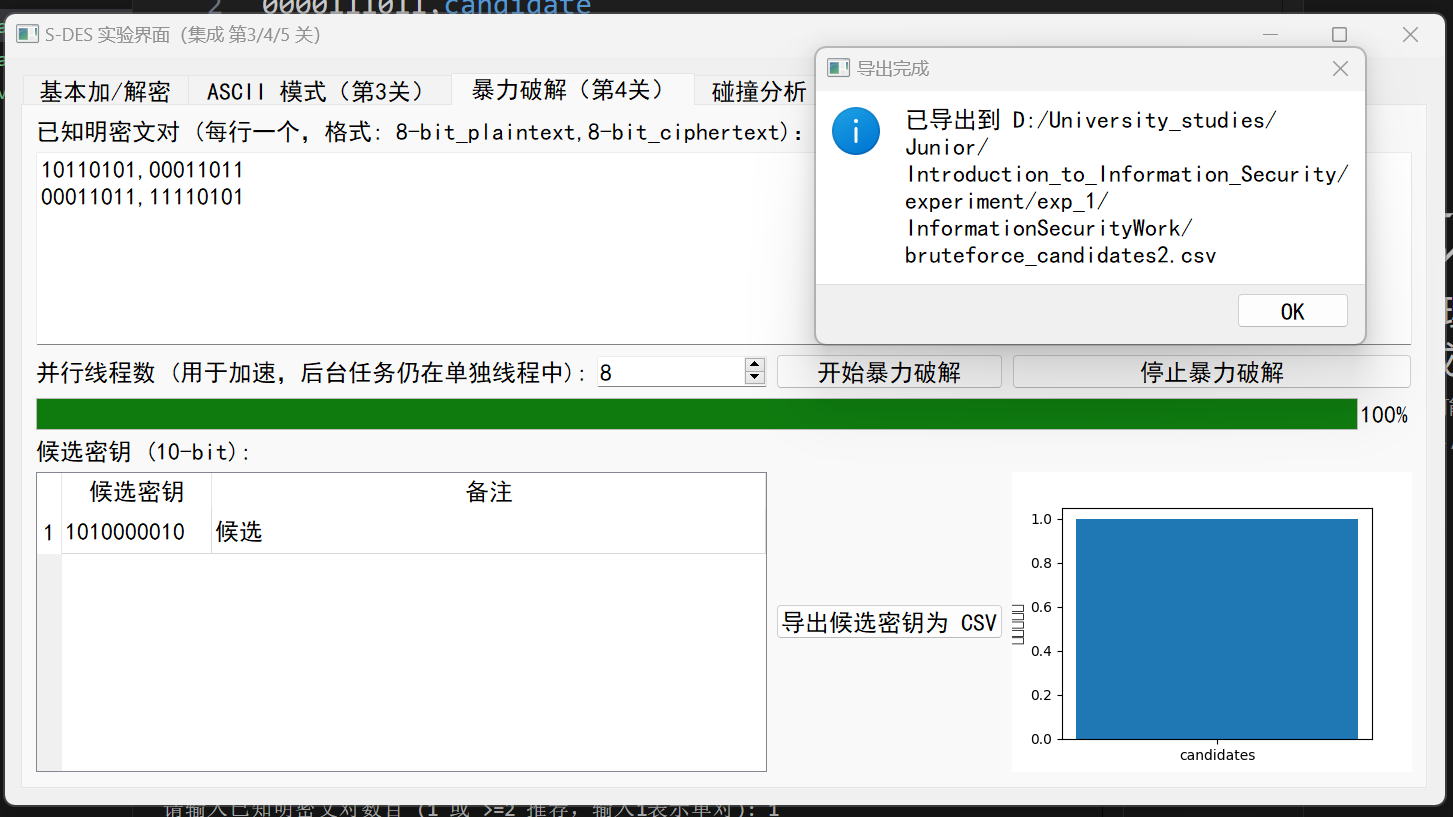


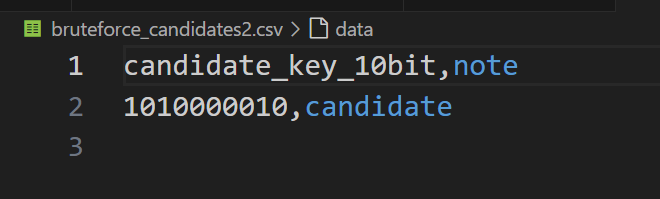




1. **两对明、密文对**（以关卡一中两对示例为例）







**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：

