代码说明

最初想要仿照 pollard-rho 算法设计一个生成比较碰撞序列的函数，但 pollard-rho 的那个函数很巧妙的一个点是如果x_i和x_j模 N 的一个因子 p 同余那么序列里所有相距 i-j 的两个数模 p 都同余，所以如果能证明序列在一个循环内是随机的就省去了朴素生日攻击要检测所有的数两两之间作差求其与 N 的最大公约数这个环节，极大的降低了时间复杂度。但是对于 SM3 算法想要构造一个有相似性质的优秀函数十分困难，于是我将优化方向转变为主要对空间复杂度进行优化。一开始我选择对两个数据分别迭代 hash 然后每次比较 hash 结果是否碰撞，但这样和固定一个 hash 结果不变，只迭代 hash 另一个数据并和那个固定的 hash 结果比较是否碰撞所需要的比较轮数期望相同。故最后我采用求第二原象的算法，对于原始数据 0 和原始数据 1，分别计算其 hash 值并比较是否碰撞，如果没碰撞则保存原始数据 0 不变，原始数据 1 赋值为其 hash 值，并再次计算原始数据 1 的 hash 值是否与原始数据 0 碰撞。

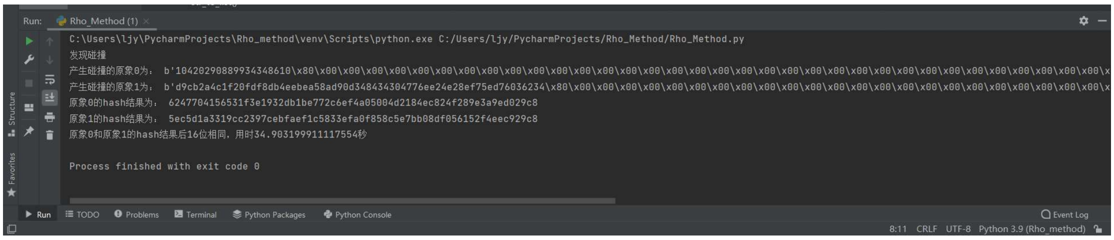
运行指导

下载项目解压后在 Pycharm 中打开该项目，安装国密库 gmssl 直接运行 Rho_Method.py 文件即可

运行过程截图

Hash 结果后 8-bit 碰撞

Hash 结果后 16-bit 碰撞



Hash 结果后 20-bit 碰撞

成员分工

该项目由廖健有独立完成