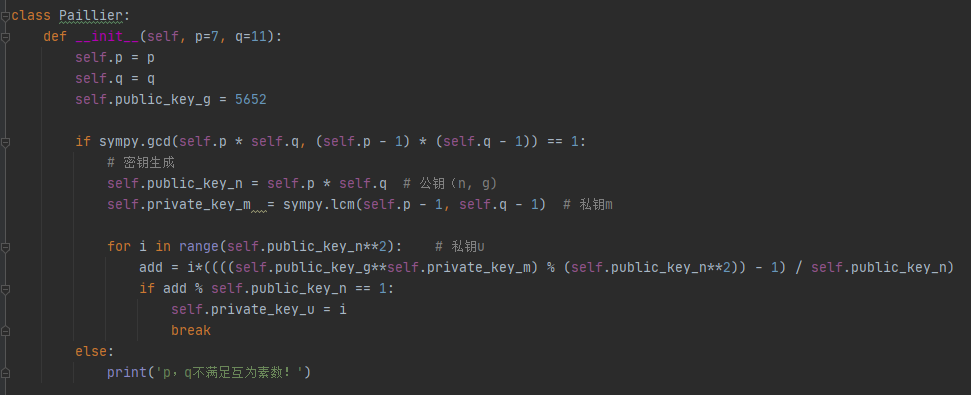
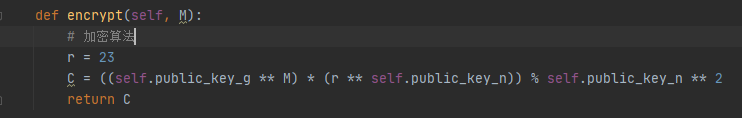
1. 实现一个乘法同态加解密算法并验证其满足的同态性
2. 实现一个加法同态加解密算法并验证其满足的同态性

两题算法有一定相似性，故在同一个程序中实现，便于比较理解。

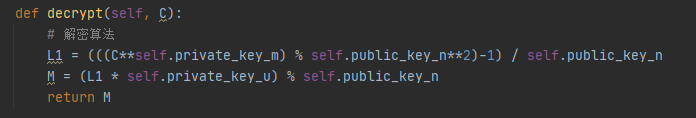
加法同态的类定义，主要包含有公钥密钥的生成以及加密解密部分



加密算法：



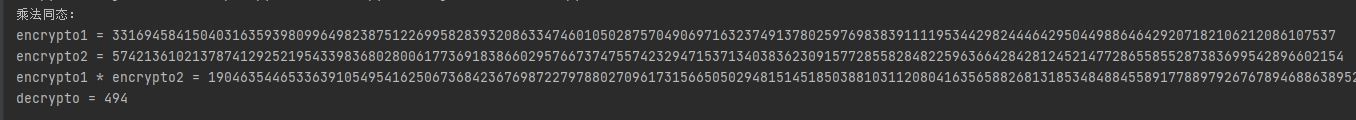
解密算法：



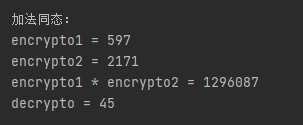
乘法同态和加法同态的函数调用部分，其中乘法同态使用的是库函数调用来实现，可以看到二者算法的过程步骤有一定相似性。



乘法同态的结果



加法同态的结果

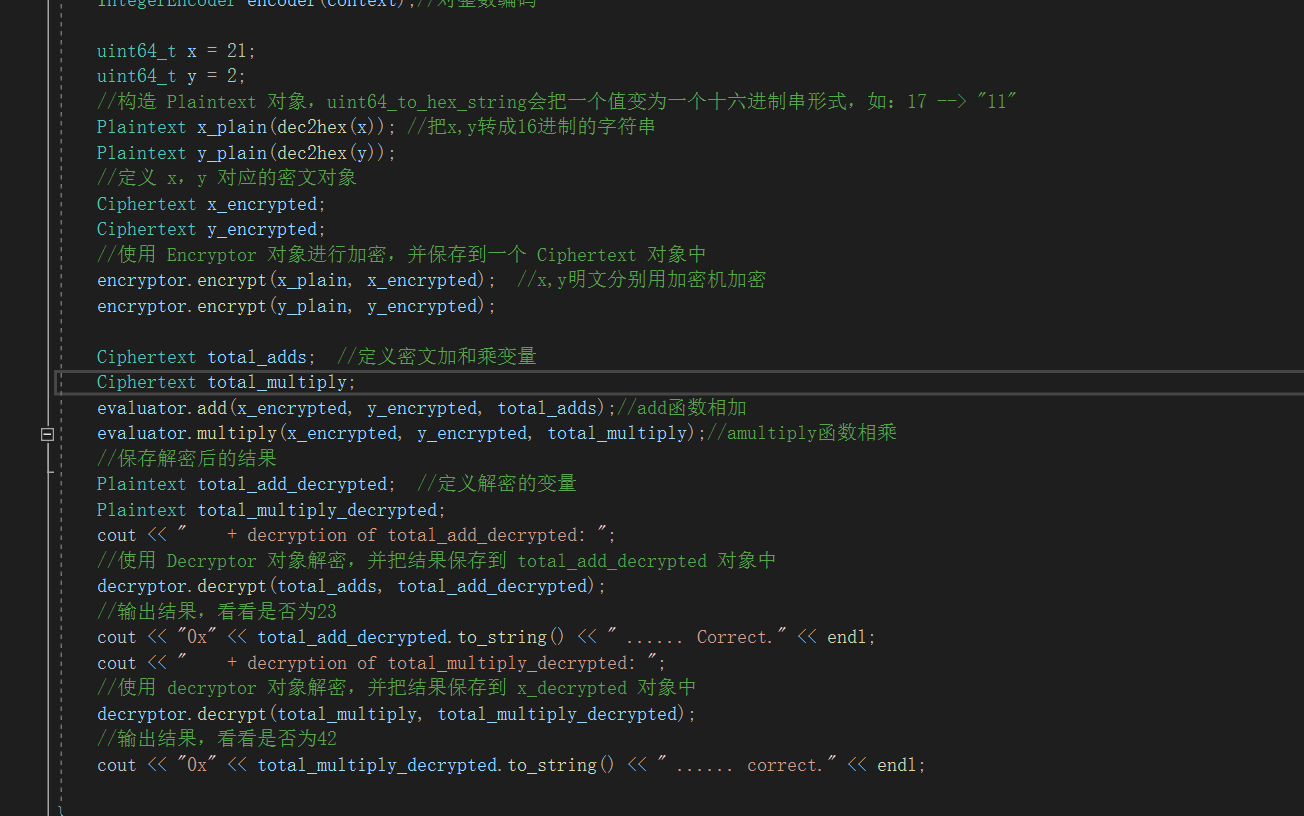


其中有一个比较重要的性质，在解密中，密文乘等于明文加。

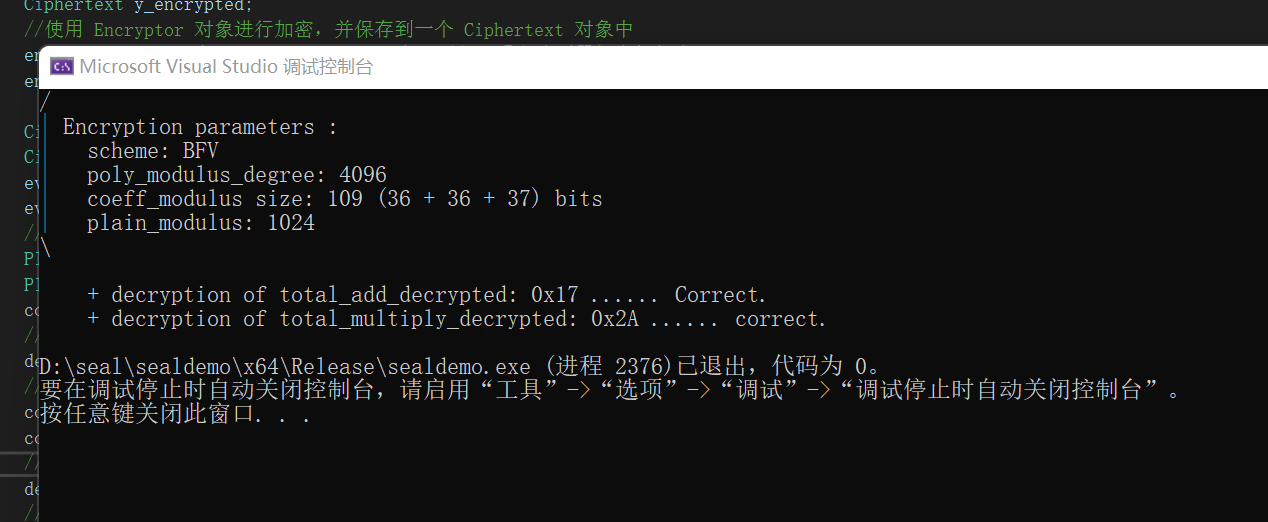
1. 基于全同态加密库SEAL实现全同态算法BFV并验证其满足的同态性

SEAL库是基于C++的库，很久没有使用C语言了，实现起来还是经历了不少麻烦。



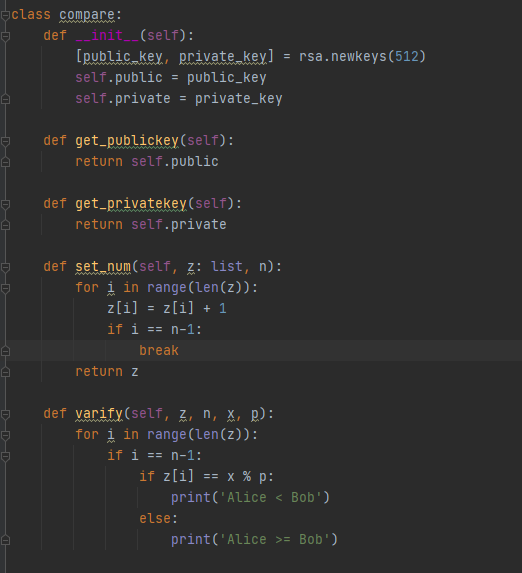


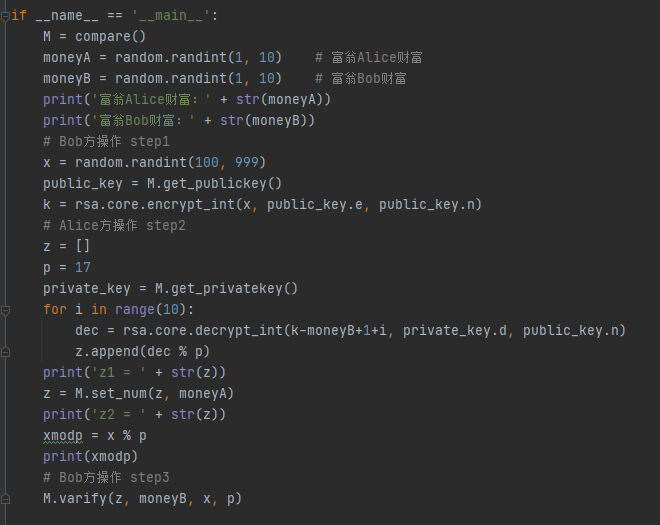
输出结果：



1. 百万富翁问题算法实现（注：自己设定进行比较的输入值，通过算法得到比较结果输出，保留必要的中间结果并在报告中进行安全性解释说明）

比较函数部分，定义了一个类，存储公钥私钥和对比。





算法核心：

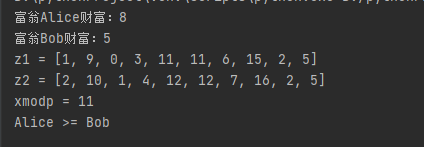
使用库函数，随机选取1-10的整数作为两个富豪的财富开始比较。

Bob挑选一个整数x，然后用Alice的公钥a加密。得到 k=Enc(x)。然后经过简单运算把k−j+1发给Alice。

Alice根据k−j+1，使用私钥解密得到Dec{k−j+1, k−j+2, k−j+3 ...., k−j+10}然后除以一个素数p取余得到：z1 = Dec{k−j+1}(mod p), z2, z3, .... z10，Alice根据自己的钱i， 在z1到z10的前i-1个数字加一。

Bob根据自己的财富j，只看z中的第j个数字，并与x mod p相比较，如果等于x(mod p)就说明Bob的财富与Alice相同或者比Alice多，否则说明Alice财富更多，然后bob把结果返给Alice。

**输出结果：**



Z1数据集为Alice解密出来没处理过的数据集，Z2数据集是Alice根据自己的财富值，在相应位置修改后的数据集。

Bob的财富为5，则会看到Z2的第5位为12，而x mod p的值为11，不相等则Alice的财富更多。