《数据安全》实验报告

姓名：管昀玫 学号：2013750 班级：计科一班

**实验名称：**

频率隐藏OPE方案实现

**实验要求：**

参照教材6.3.3FH-OPE实现完成频率隐藏OPE方案的复现，并尝试在client.py中修改，完成不断插入相同数值多次的测试，观察编码树分裂和编码更新等情况。

**实验过程：**

1. 理论知识

频率隐藏保序加密：密文和如果满足，则。

**有交互的FH-OPE方案**：该方案隐藏相同明文出现的频率，在一定程度上提升了方案的安全性，并抵御了一部分利用明文频率发起的攻击。该方案在客户端维护一个二叉排序树，将明文插入到二叉排序树中。通过参数的设定来减少排序树的调整。但是排序树一旦发生调整，将带来巨大的性能消耗。不仅如此，客户端的大存储，也使得该方案不实用。

当对明文进行加密时，从根节点开始遍历，依次与当前节点中存储的明文进行比较，如果当前节点的明文等于插入的明文，则生成随机数，否则继续向下层遍历。在解密时，需要从根节点开始对索引树进行遍历。如果索引树需要调整，它必须是一个二叉平衡数。

**无交互的FH-OPE方案**：为了在无额外交互下运行算法，该方案设置了一个本地表作为客户端存储，记录明文以及出现次数。每当新的明文被加密时，在本地表的帮助下，找出有多少现有的明文值小于和等于的。因此，很容易确定相应该明文在B+树中的随机顺序。该方案改进了B+树，每个中间节点不再存储关键词，而是存储节点后代中包含的密文的数量。从而完成在1次交互下将密文插入B+树中。

当树需要重新平衡时，密文的路径编码即保序编码将被更新。如果更新涉及的密文很多，会严重降低OPE方案的性能。对于服务端树，为了减少密文重新编码的频率，需要在调整树时不更新保序编码。该方案采用区域编码策略：每个叶节点值区间为，默认新节点编码策略：，为左邻居的编码，为右邻居的编码。节点内编码更新策略：区间、密文个数为，更新后。在这种情况下，树的平衡调整不会引起编码的更新，插入可能引发节点内数据密文编码更新，但是其他节点不发生更新。

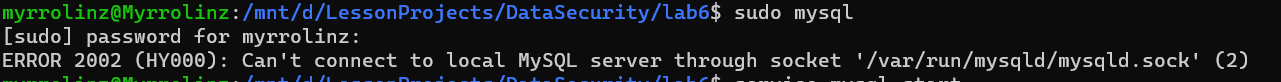


2. 实验环境安装

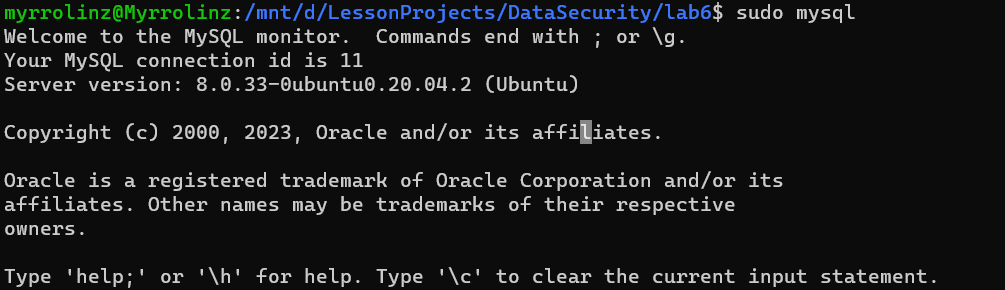
首先执行以下命令，以在我们的环境中安装sql。

[sudo] apt install mysql-server libmysqlclient-dev

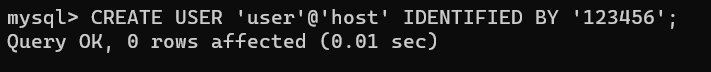
然后用sudo mysql指令登录数据库。这里遇到一个问题（我使用的是wsl2环境）如下所示：



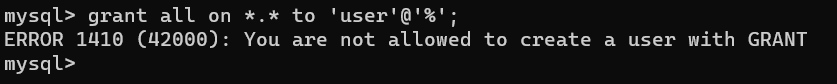
上网搜索解决方法，感觉都不像是我本地出问题的原因。心灰意冷之下再重新输入了sudo mysql指令一次，居然通了，这个问题我到现在也不知道是怎么导致的，只能说有玄学，先往下做实验好了：



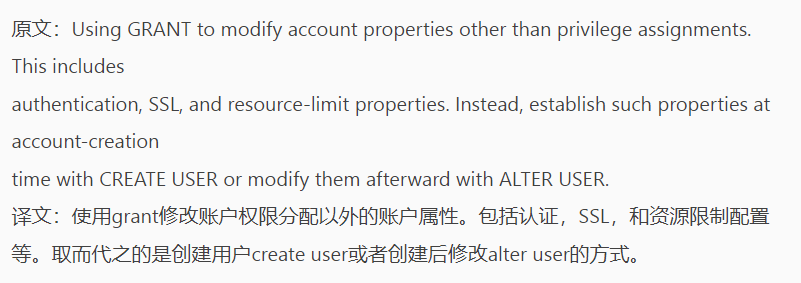
然后使用create user指令创建用户，如下：



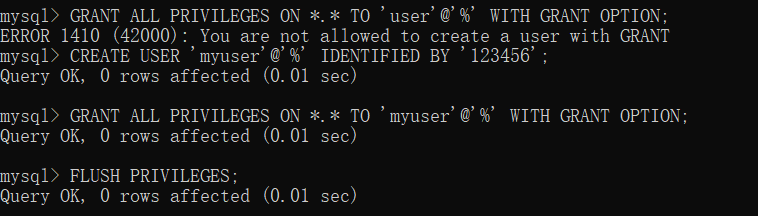
创建完成后，使用grant命令授予权限。这里又遇到了一个问题：



这是因为在8.0后这个特性已经被移除了：



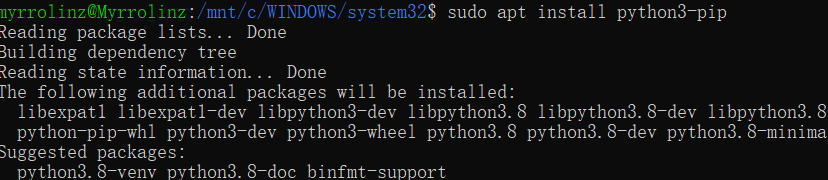
修改语句重新尝试，使用新的语句之后就ok了：

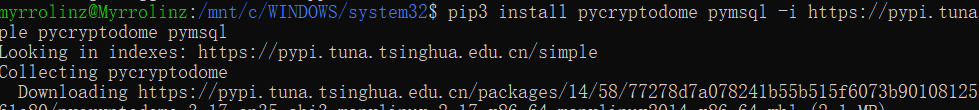


之后创建数据库test\_db:

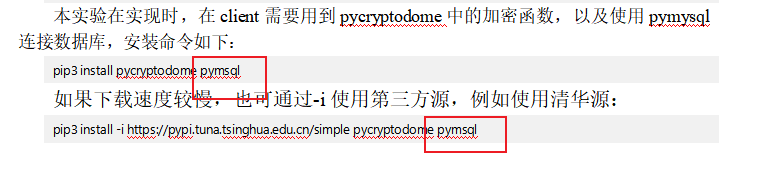


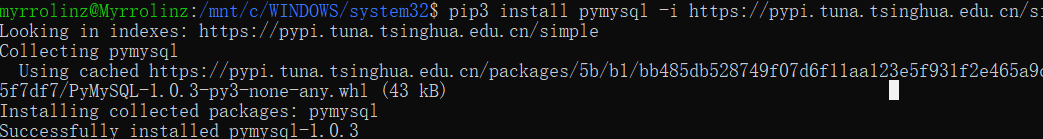
Python相关：安装pip3和相关库，实验环境配置即可大功告成：





注意一下，这里教材打错了，是pymysql，而非pymsql，会报错

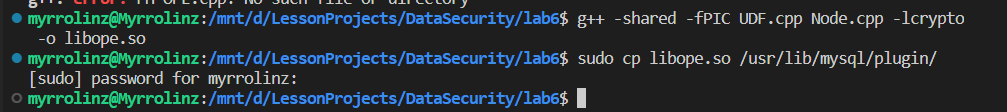




实验环境配置到此结束。

3. 编译生成动态链接库

提前准备好Node.h, Node.cpp, UDF.cpp等文件，然后执行以下指令即可：

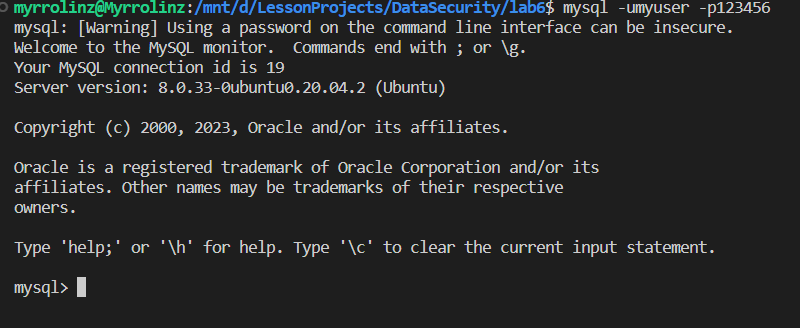


需要指正的是，教材这里也出现了纰漏，此处应为Node.cpp：

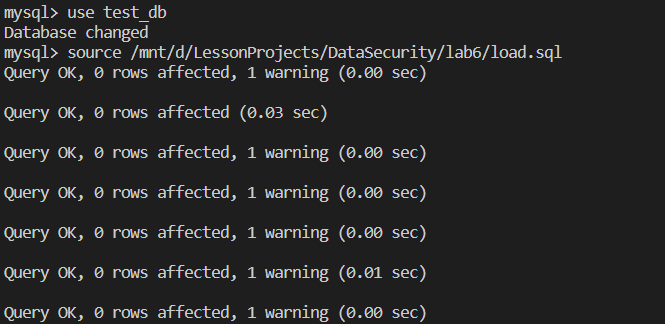


1. 导入MySQL

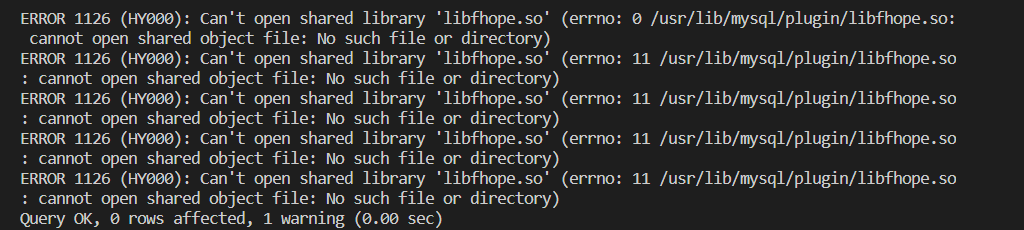
首先登录数据库：



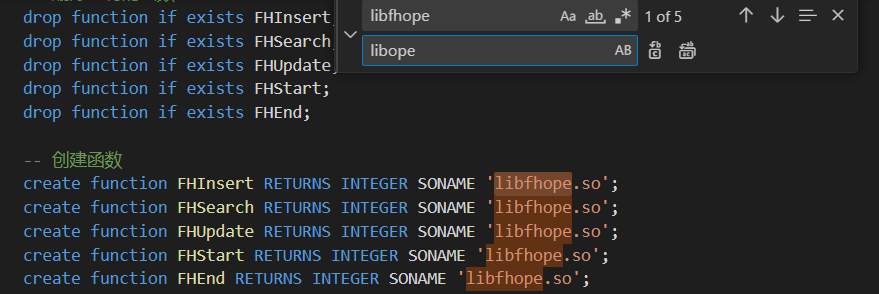
然后使用test\_db数据库，并导入sql文件。注意，此处应使用绝对地址：



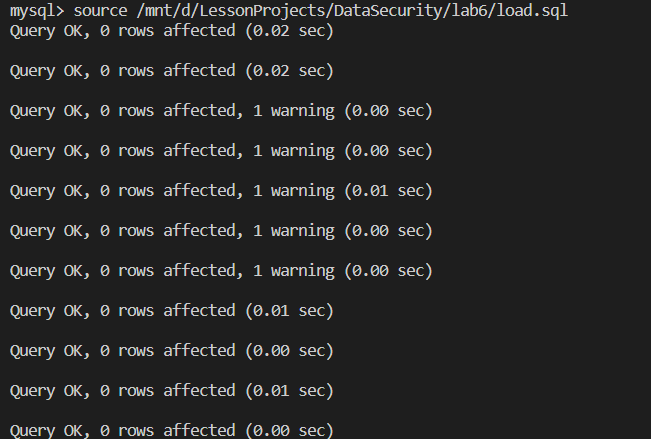
但是随后出现了一个问题：



这是文件名不一致造成的。教材这里前后名称不一致，应注意修改。这里我选择一键替换sql文件中的文件名：

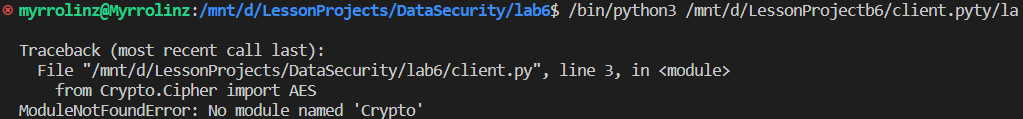


之后重新执行一遍该指令即可：

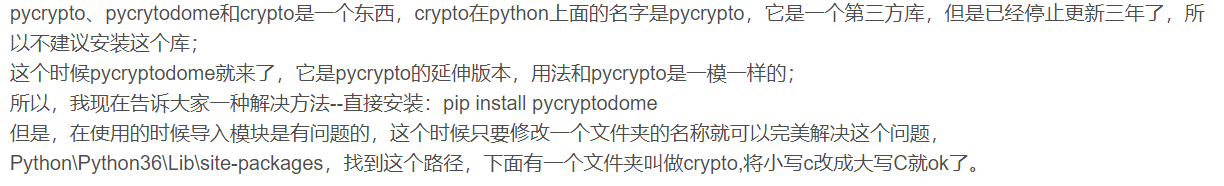


1. 执行client.py

一执行就发现有报错提示：

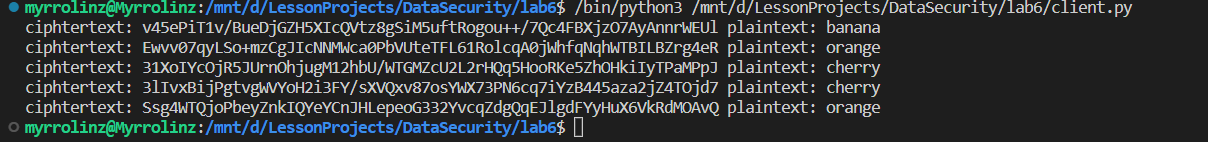


查资料后才发现：



（但是之前我已经执行了pip install pycryptodome啊，不应该啊，~~也有可能我装在Windows里了~~）

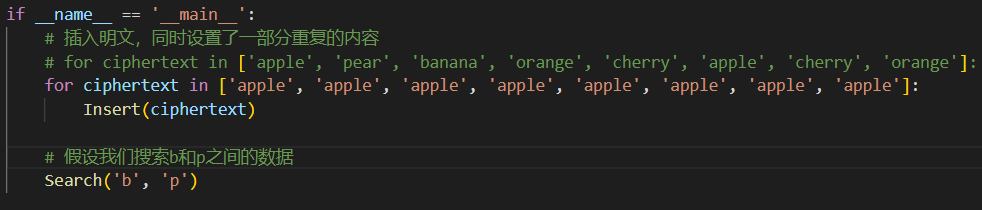
装完库执行后就可以成功：



可以看到实验搜索了b和p之间的数据，结果打印了密文，以及解密后的明文。从中可以看到，其中orange和cherry编码有更新。

1. 修改client.py，多次插入相同数值

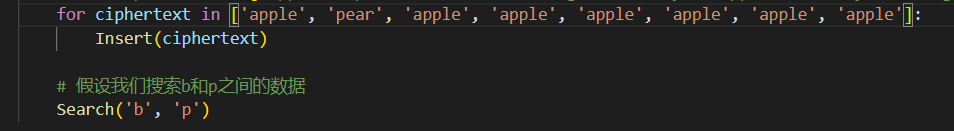
这里我尝试将所有ciphertext都改成apple：



执行，搜索，发现没有没有任何数据。这是因为所有的值都是一样的，b与p之间自然没有数据。



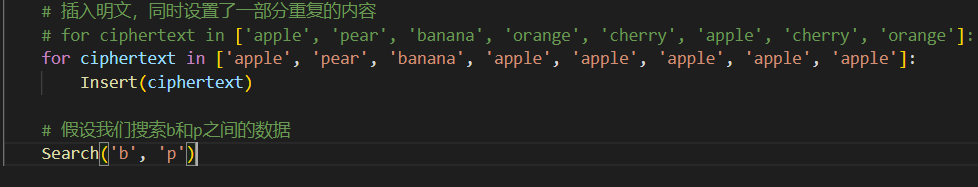
再次尝试，只有apple和pear：



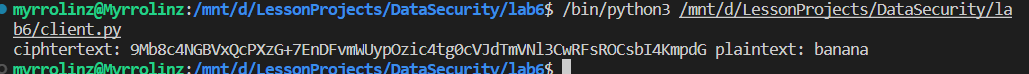
执行，同样还是没有数据：



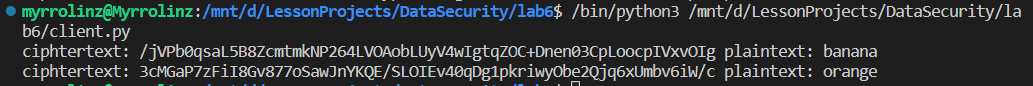
再次尝试，现在拥有apple, pear, banana：



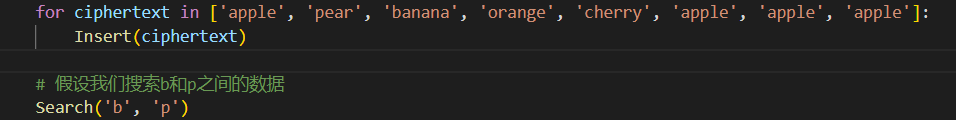
执行，我们发现b与p之间有一个值了，这个值是banana：



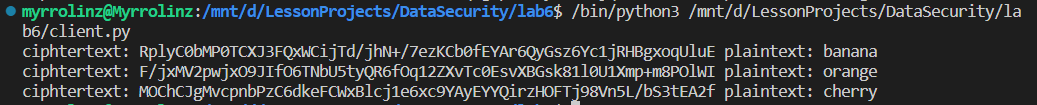
再次尝试，现在有连续4个不同值，搜索后发现b与p之间的值增加了，有banana和orange



再增加cherry，现在有连续5个不同值，但最后三个值还是以三个Apple结尾：



从结果中我们可以看见该二叉树有更新，现在b与p之间有三个值了：



通过这些实验结果，我们可以发现该程序维护的二叉树是一个不断更新的过程，在分裂和更新中保持平衡，而编码有时会随着二叉树的调整而进行更新。

**心得体会：**

通过本次实验，我更深入地了解到了HF-OPE方案。在实验过程中，我也遇到了一些问题，特别是MySQL的知识我已经忘得差不多了，这次实验帮我回想了知识，而且也提升了我独立解决问题的能力。

总之，动手的实践作业让我收益良多。