|  |  |
| --- | --- |
|  | 图片包含 瓶子, 标志, 不同, 桌子  描述已自动生成 |



**合肥工业大学**

**计算机与信息学院**

**课 程： 应用密码学综合设计**

**专业班级： 信息安全20-1班**

**学 号： 2020210594**

**姓 名： 米永丰**

**必 做**

**一、实验题目**

网络入侵——口令破解HASH实验

**二、实验要求**

根据实验指导完成实验。

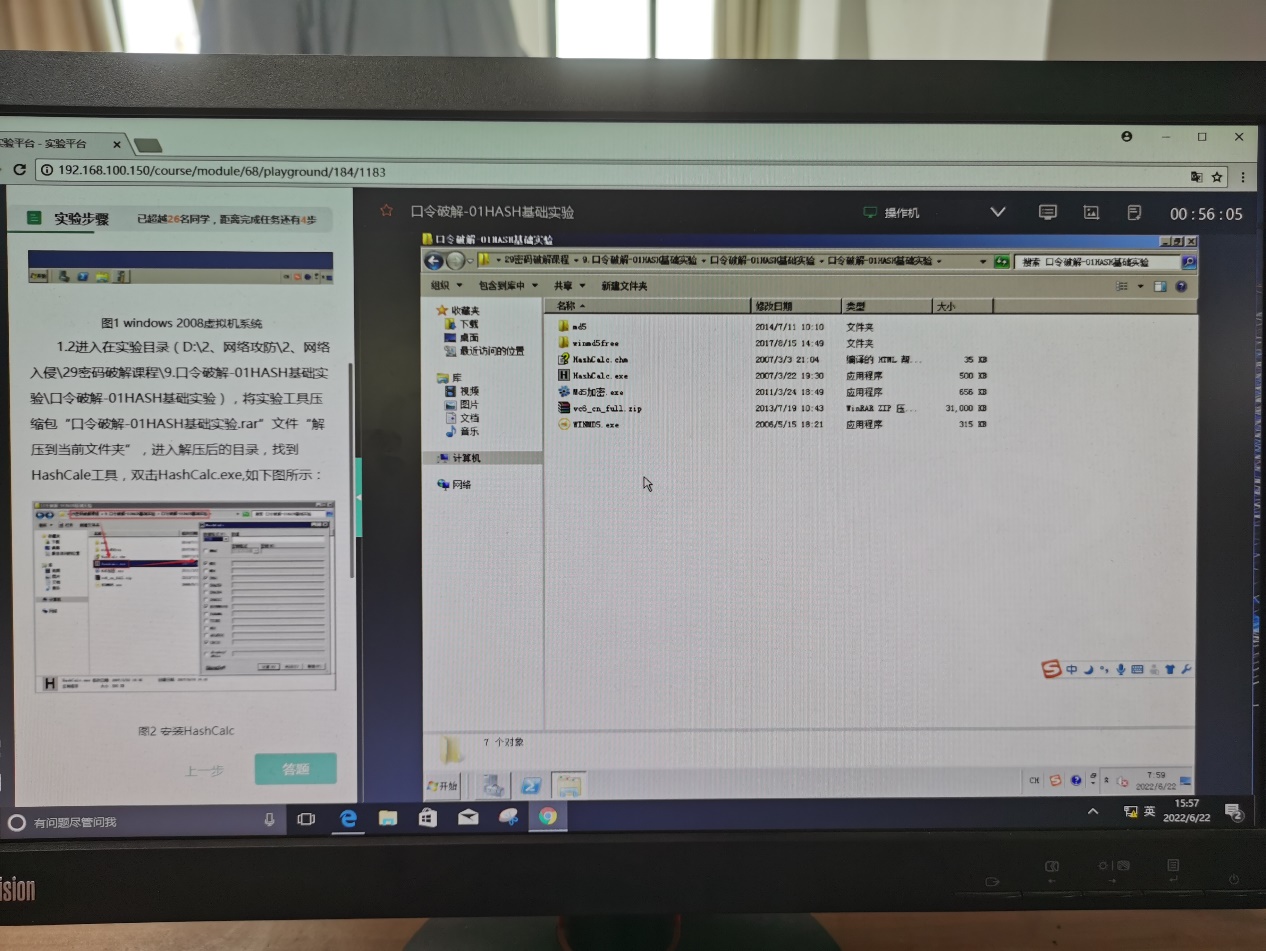
**三、实验步骤**

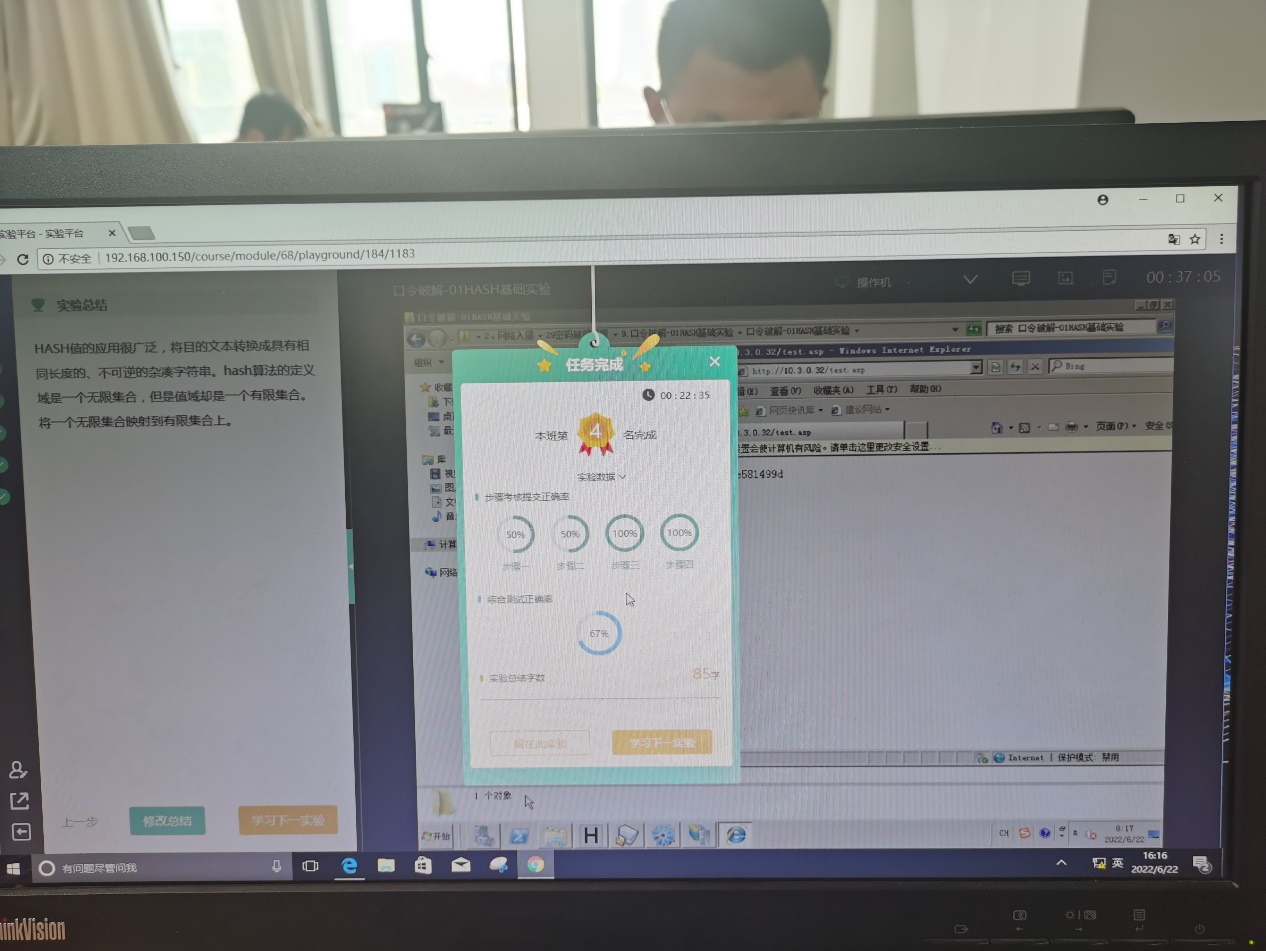
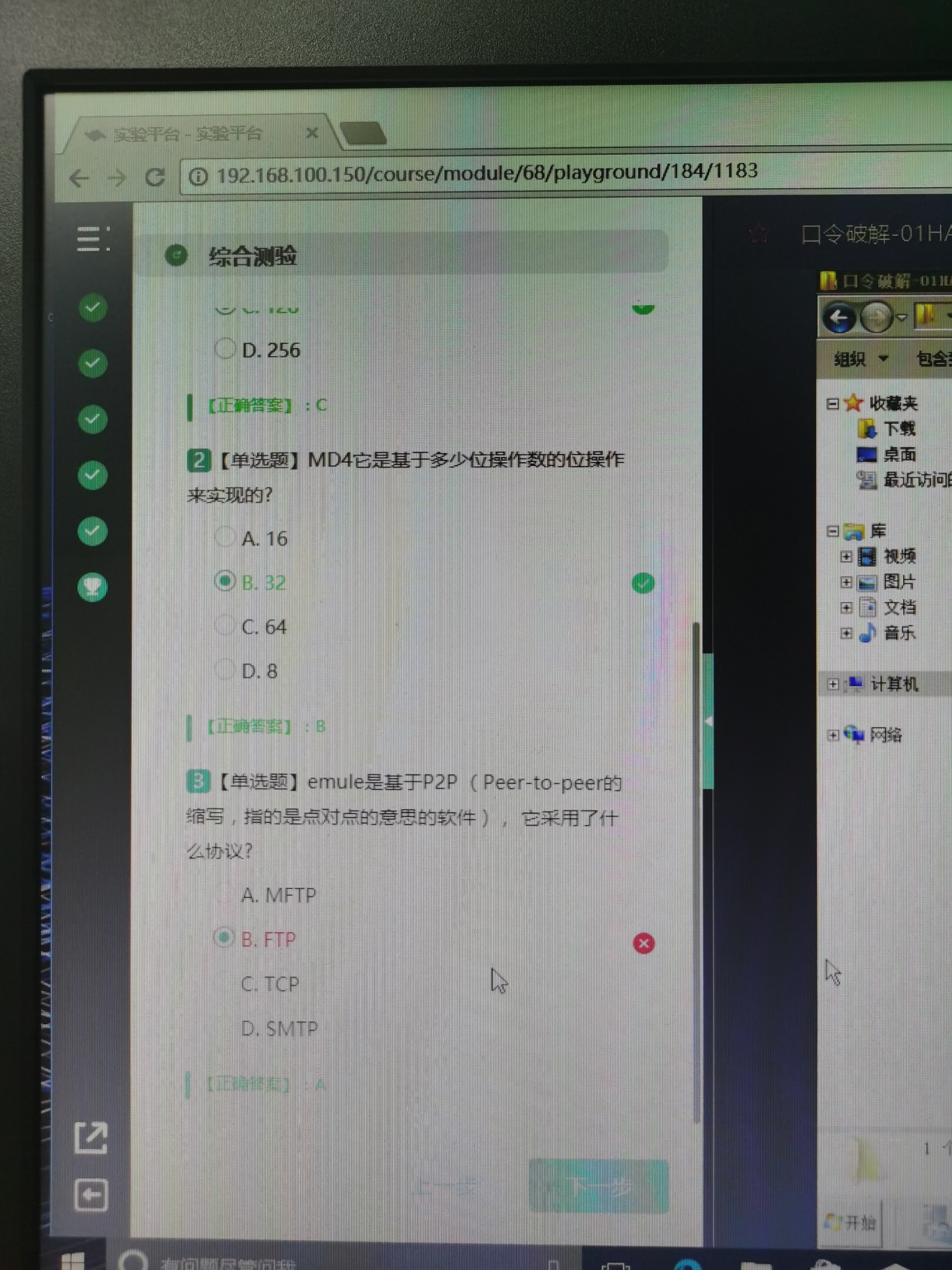
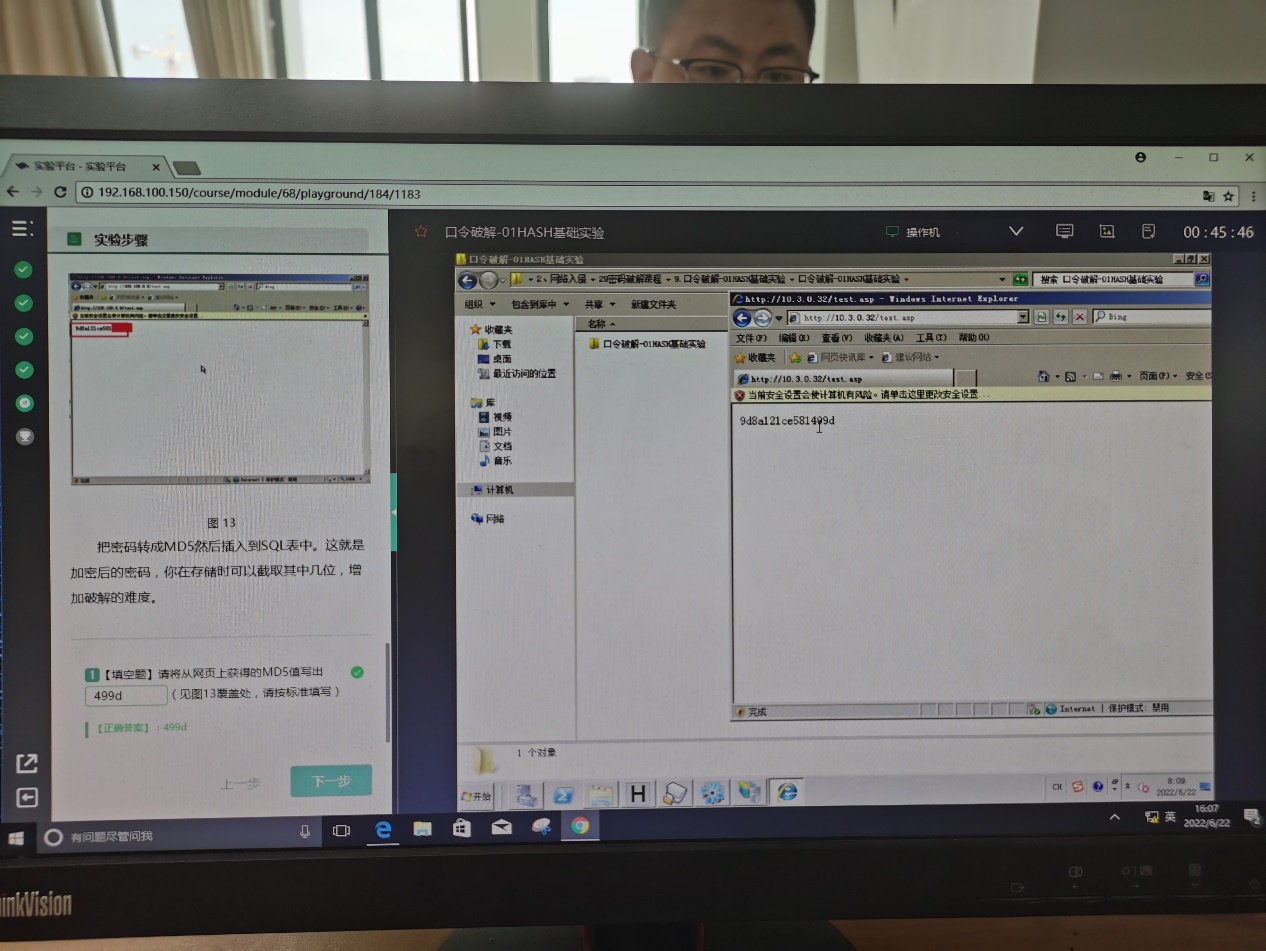
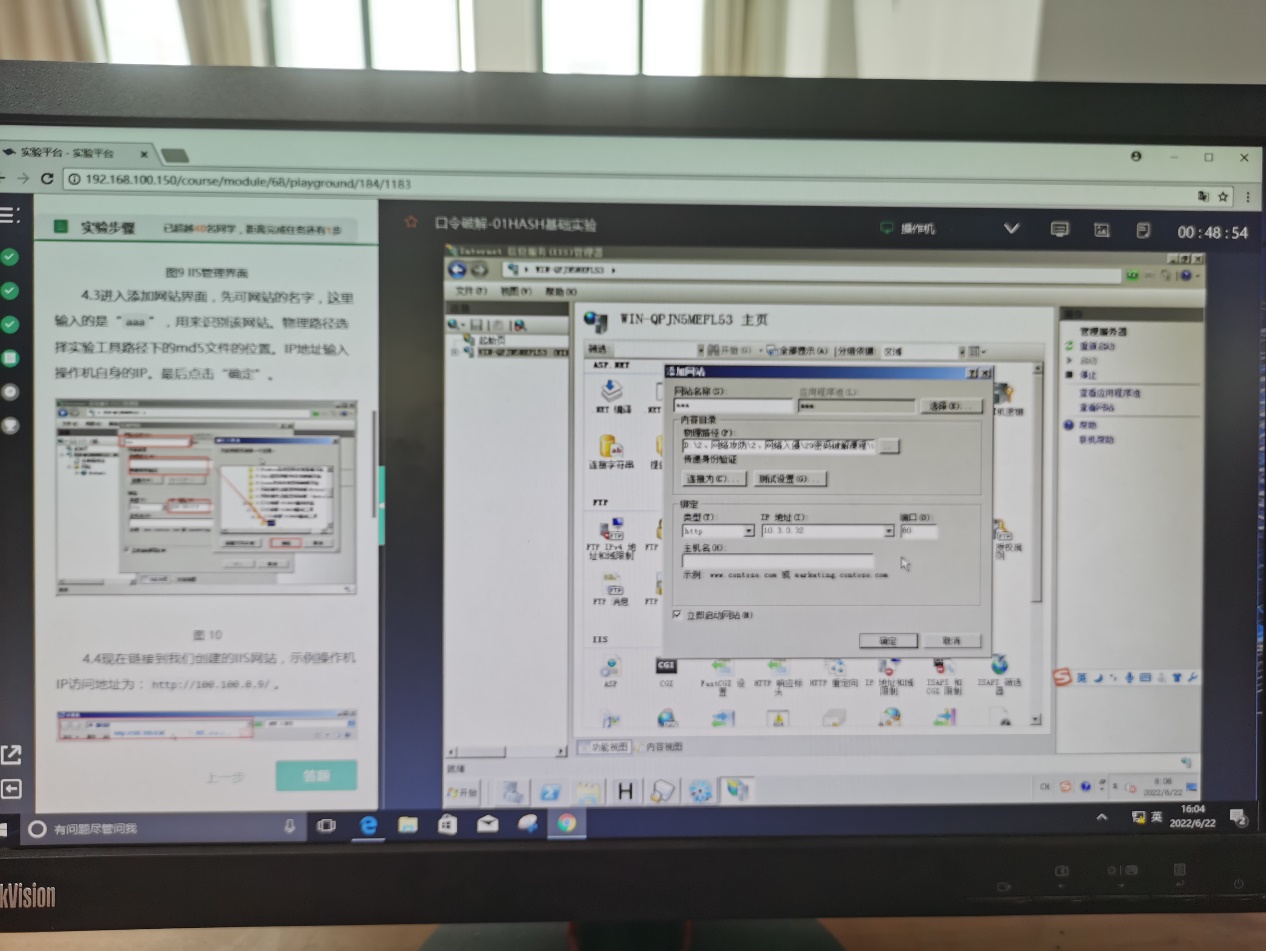
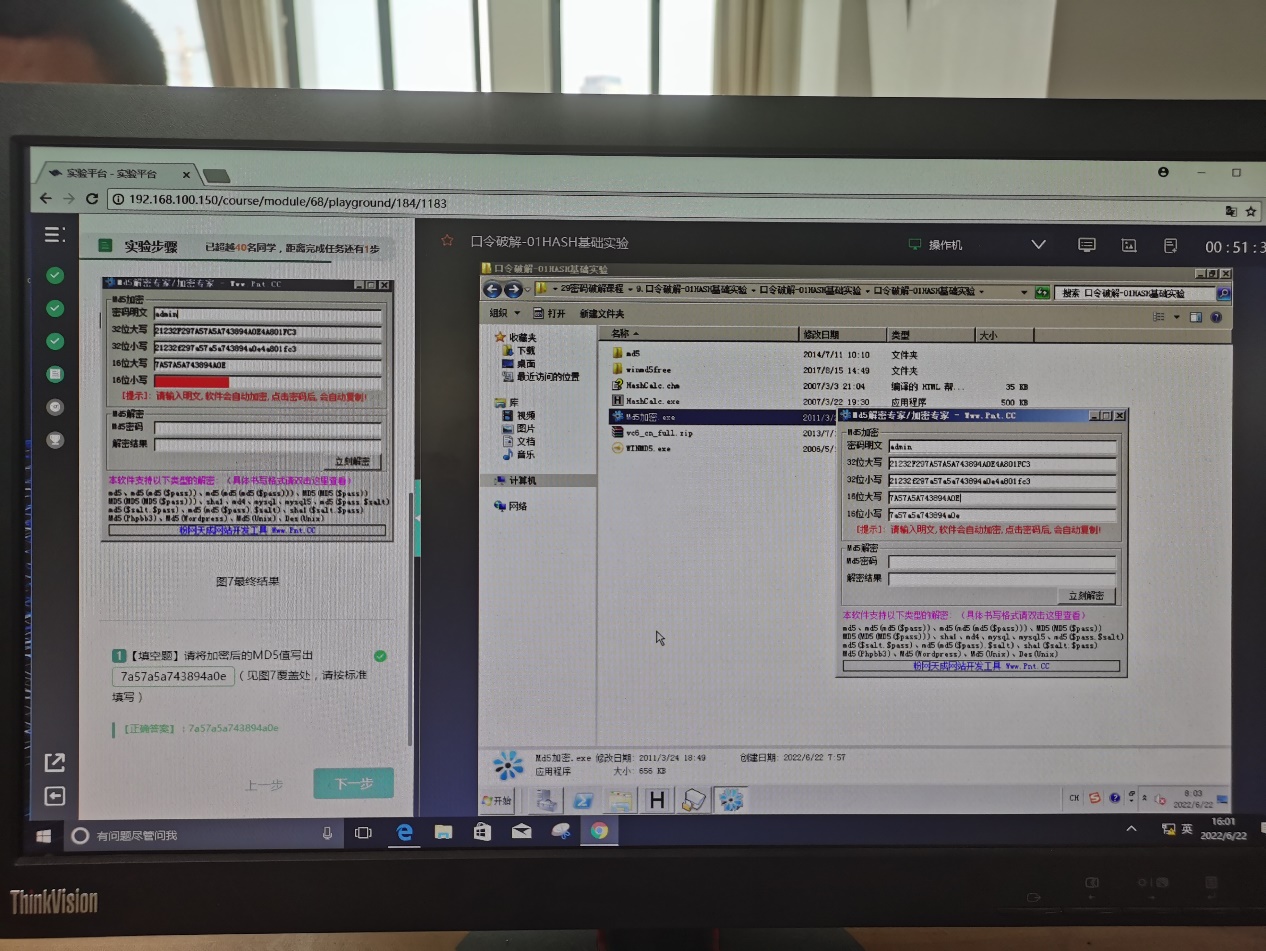
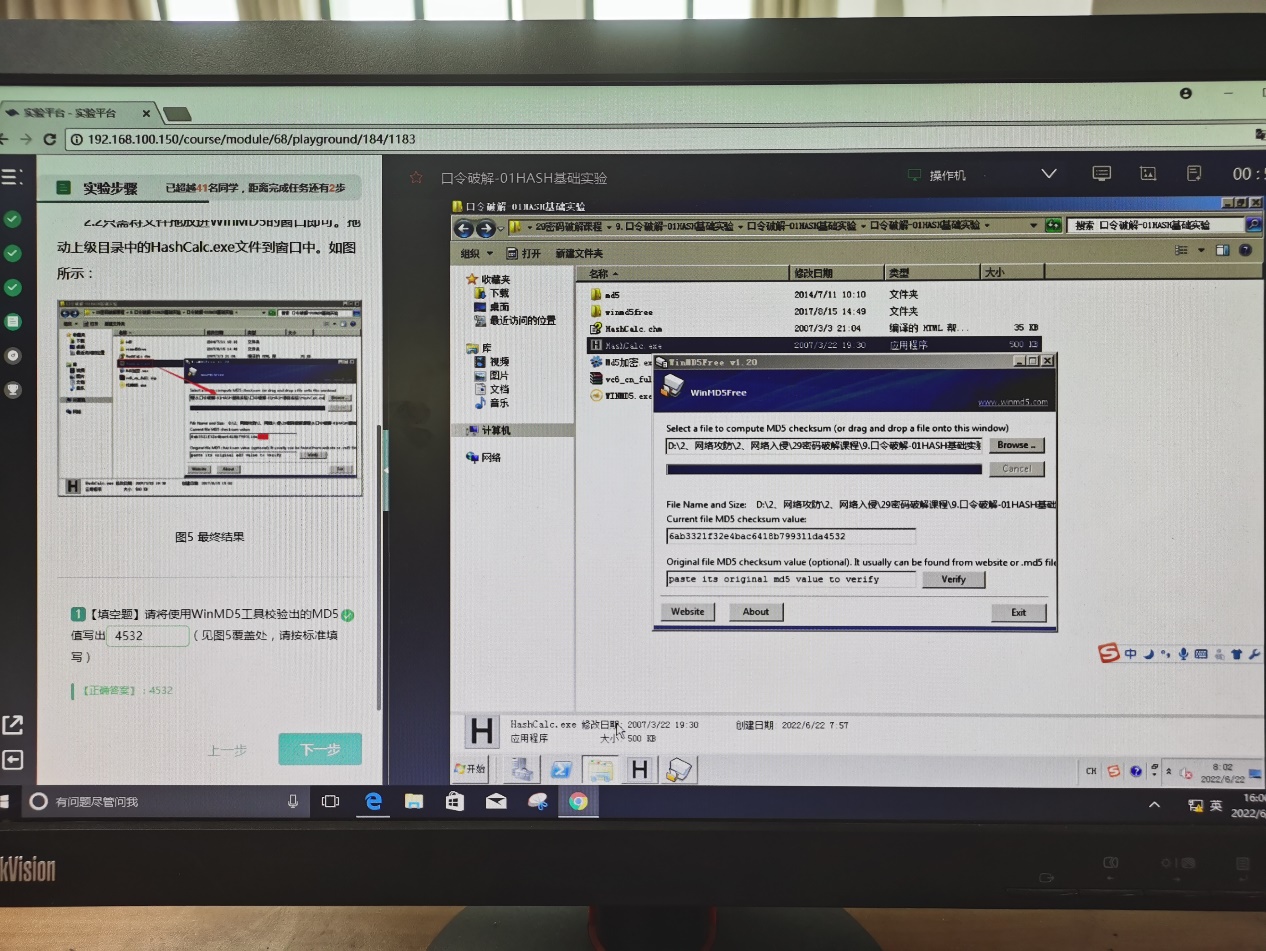
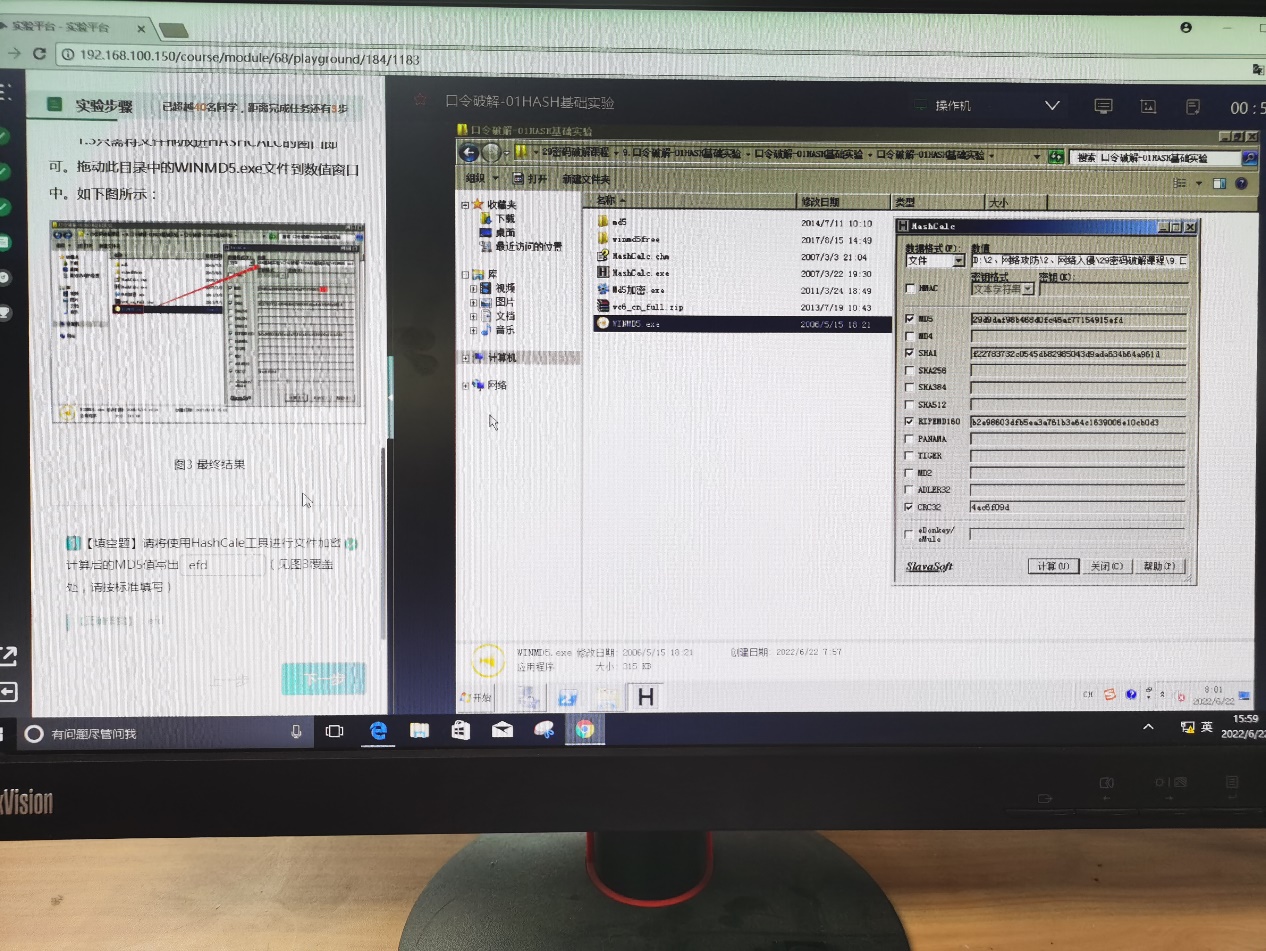
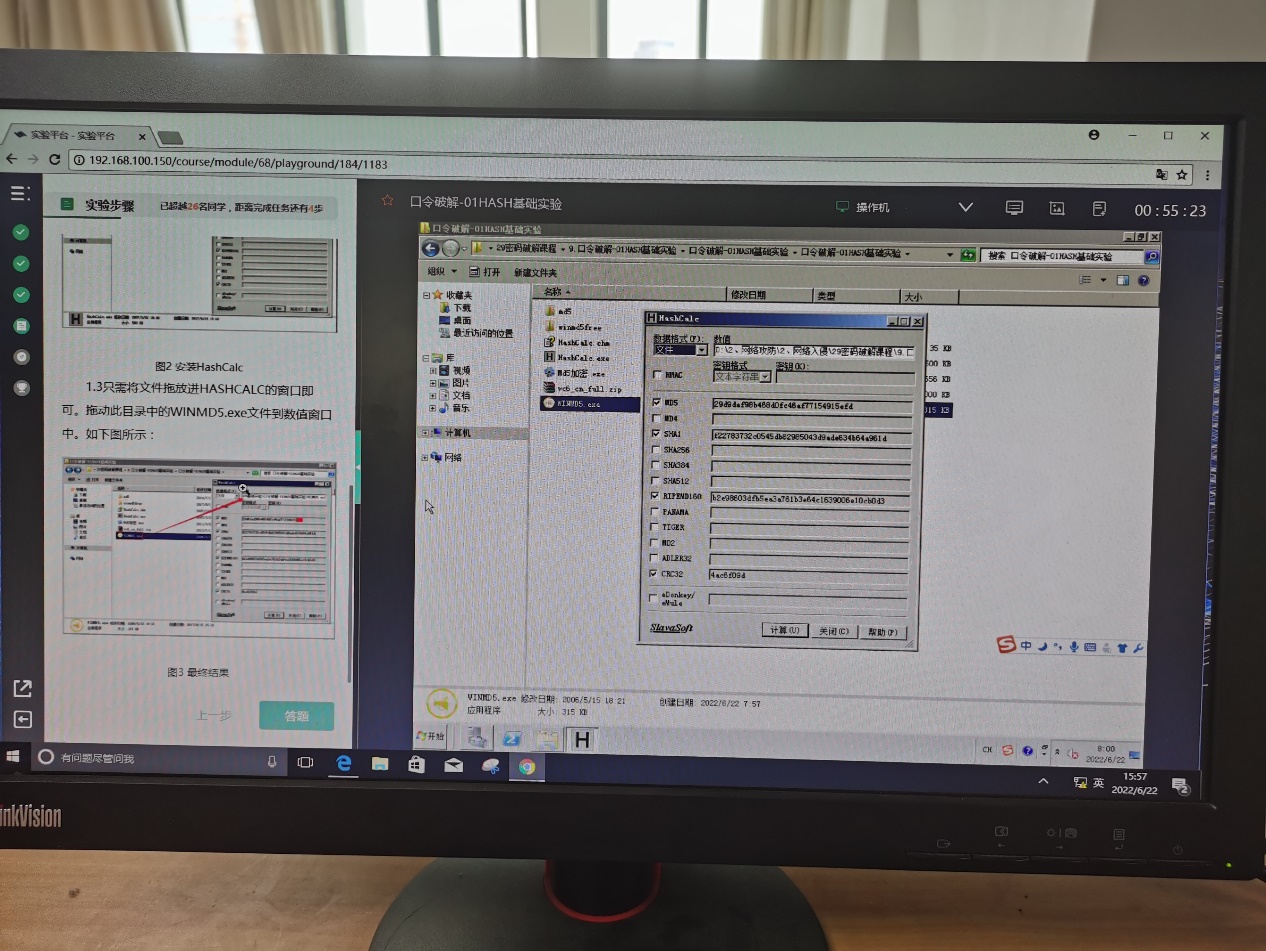
1.进入相应目录下，选择实验工具，将目标文件拖入，记录得到MD5值；

2.利用校验工具检测得到的值是否正确，并填写校验值；

3.打开网站，把输入的密码转成MD5值并插入到SQL表中，填写加密的密码， 可以截取后几位加大破解难度；

4．作答测试题。

**四、运行结果**

****

**选 做**

1. **设计题目**

密码管理器

1. **设计要求**

1）用户用口令登陆系统

2) 用户可创建多个用户名和相应密码对，即(用户名，密码)，密码在指定字符集中随机产生

3）系统存储登陆口令的HASH值，用户名及其HMAC值，密码的AES密文

4）由登陆口令生成HMAC和AES的密钥

5）需要时可恢复用户名和相应密码

1. **系统设计**

**运行环境：**

Linux version 5.13.0-51-generic (buildd@lcy02-amd64-046) x86\_64

Ubuntu 20.04.4 LTS；

gcc 9.4.0

需要安装依赖库以及配置相关环境

**设计思路：**

1. 用户使用口令登录，这里首先需要后台注册信息，口令由指定字符集随机生成（ASCII字符集，使用当前时间作为随机种子）并展示给用户，可以在登录过后自行修改口令。
2. 口令使用SHA-1算法生成HASH值单独存放；用户登录后可以自行创建用户名和密码，密码创建后只显示个数，由系统存储用户名的HMAC值和密码的AES值，用户可以选择是否展示密码。
3. 以上数据均存放在数据库里。

**设计反思：**

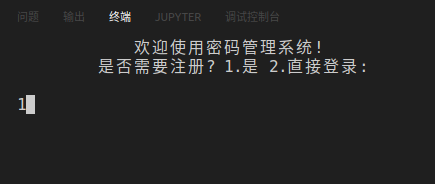
1. 首先声明，现在做出来的这个东西所有简化的地方全都是为了最初实现的时候方便管理，毕竟我也是边学边做的，不敢太过复杂，见谅。
2. 初步设计的想法是，每次用户注册成功时，系统提供一个由自定义字符集所产生的随机字符串作为口令，并存储其HASH值。但是在实现系统的过程中发现，MySQL的query语句并不支持函数作为参数，只能接受常量。为了简化系统雏形方便后续设计，现在实现的系统只能由使用者自己提供初始的登录口令（虽然这么做非常合理，但是不符合题目要求）。在后续的完善中可以使用外部函数接口提供口令输入到query语句中。
3. 系统未实现HMAC对于输入用户名的加密存储，因为我使用的SHA-1算法并不能与HMAC兼容……因为我使用的SHA-1算法是之前密码学大作业的，输入和输出都和我这个系统不相兼容，我缝缝补补修改半天才让输入和输出能够正常匹配；除此之外，每一次FT函数和循环左移，模32加这些函数都会存储当前的信息，就导致不能重复调用，除非清空初始化。但由于我当时没有提供这些接口，所以我只能封装在类里面，然后确定作用域调好生存时间，确保每次调用之前我创建的类对象都已经被析构掉了（这里我不知道是否能够显式地调用析构函数，听说有危害就没调用）。这样一来，我就更没法实现HMAC了，因为类的关系太乱了，所以直接使用SHA-1加密了其实也是因为我对于加盐的过程还不太熟悉。再说了，这是第一版所谓的beta版，后续可以加功能嘛。
4. 由于需要输入query语句，所以输入输出大多是string和char[]，互相转化的关系有点乱，需要再捋一捋。
5. 这一条不算反思——AES加密的输入和输出我定的都是string.size() = 32，所以用户管理密码的时候不能超过32个字符，否则就插入失败；如果没有32位，就在后面一直加‘0’，并且表的列属性里面有一条记录的就是原始的密码长度，所以加密后再解密就可以通过这一个属性从解密的字符串里截出来原始的密码信息。
6. 最后总结几个遇到的问题，每一个都耗费了我半天的时间。一、注意不要用关键字插入值，否则报错，我就是用了”INSERT INTO mi\_Contains VALUE （key char(32) NOT NULL)”这种语句报的错……还有，注意每次通过query语句获取结果之后都要清空结果集MYSQL\_RES\*，否则报错。
7. **运行结果**

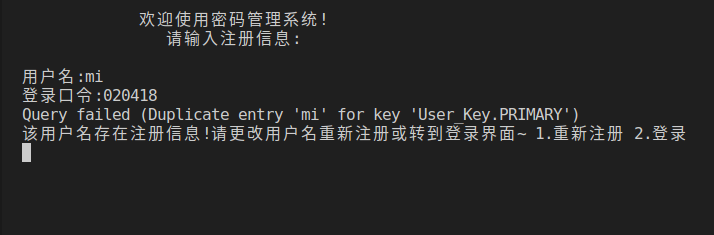
**运行流程：**

1. 选择是否注册，如果注册，进入到注册界面，输入用户名以及自定义登录口令（设计反思里对于口令的设计有说明）。如果系统发现该用户名已存在则提示，用户选择是否重新注册或直接登录。
2. 注册成功或选择直接登录会转到登录界面，登录界面输入用户名和登录口令；用户名不存在或者登录口令和用户名不匹配会有相应的提示信息。如果登录成功，会转到管理界面。
3. 管理界面就是使用界面，一共有不同的选项对密码进行管理。可以查看、增添、修改、删除密码信息，还有登出和退出两个选项。
4. 选择相应功能，查看和验证系统完整性、鲁棒性。

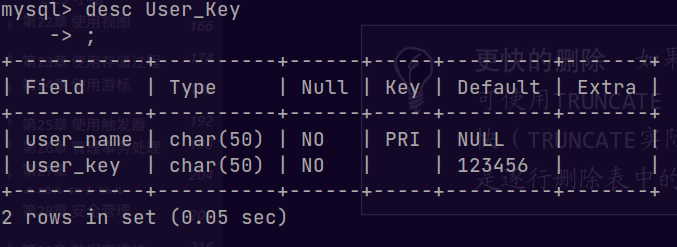
**整体流程运行结果图：**

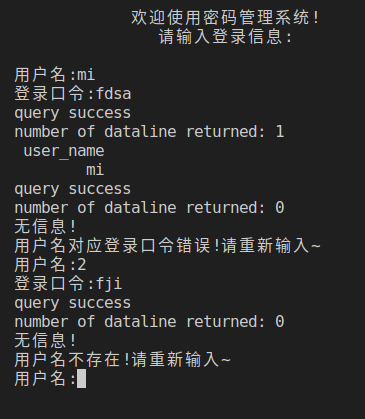
1. **注册**

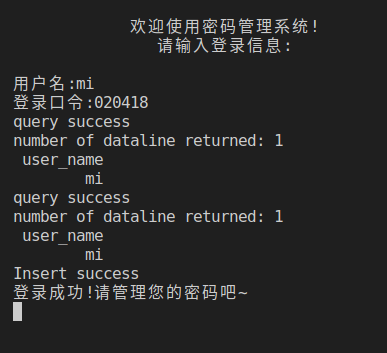
****

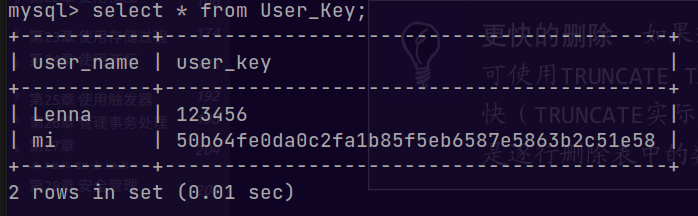
****

1. **登录**

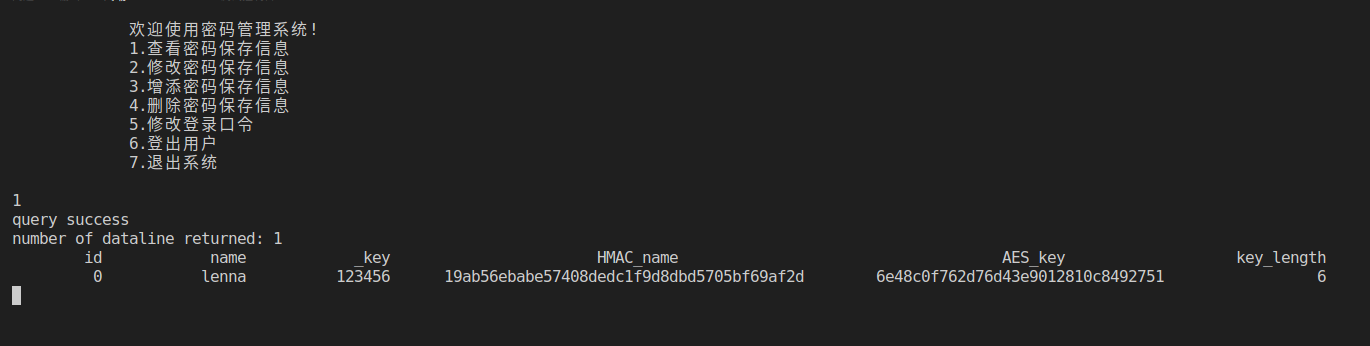
****

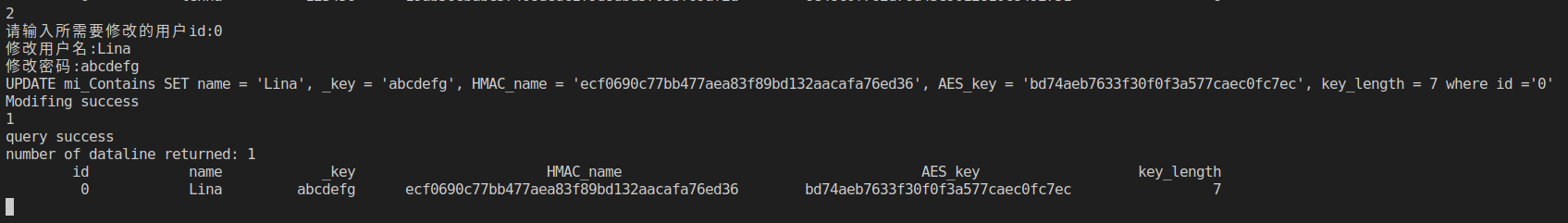
****

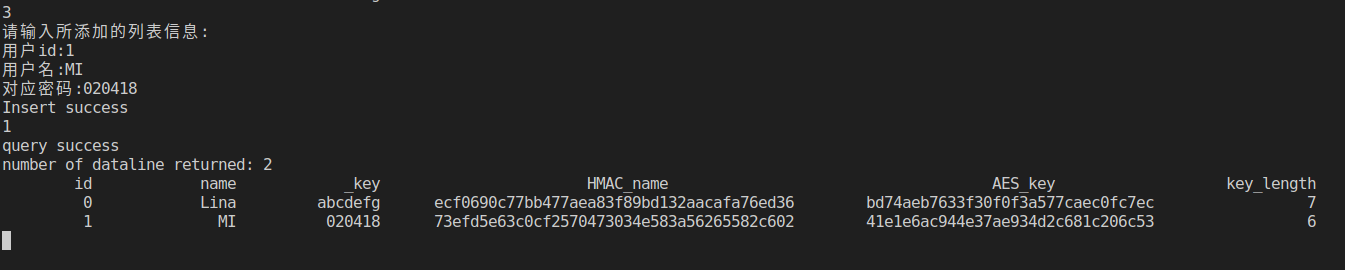
****

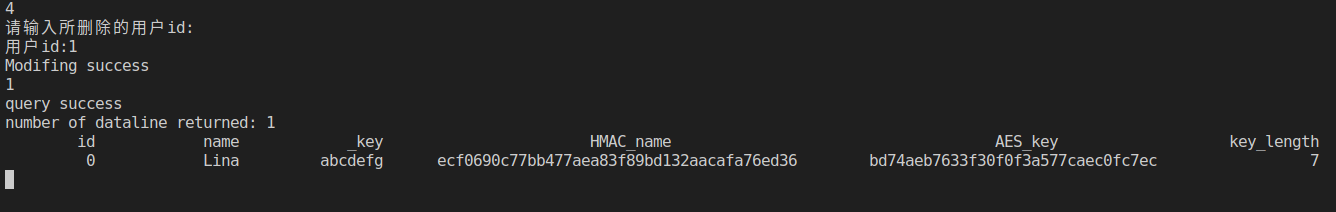
****

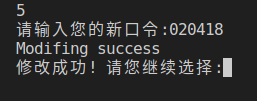
1. **管理**

****

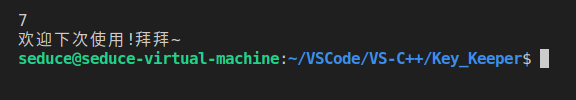
****

****

****

****

**6就直接返回登录界面了**

****

编译：g++ main.cpp -I/usr/include/mysql -L/usr/bin/mysql -lmysqlclient -o main

**五、安全性分析**

还是那句话，时间不够而且是初学者，所以功能并不完善。这里的用户信息都是可见的，其实可以学KeyPass的实现，就是都是加密的信息，只有用户在申请需要密码信息的时候才经过验证发送给客户。

密码的安全性来说，其实用SHA-256或者SHA-512更好一点？我之前用了SHA-1实在是不想改了。其他的没啥，本来实现的就是一个本地的密码管理器，不存在什么网络渗透一说。如果我把所有信息都加密，那么安全隐患就出在验证信息和加密算法本身了。但是没有网络的加入我又很难得到用户的验证信息比如说指纹、面容识别或者邮箱验证码之类的。所以，以后等技术精进了可以试着把这个系统部署到远程，然后利用服务器的数据库来实现。

**六、总结展望**

呃，安全性分析已经说的差不多了，其实整体参考Keypass的实现就行。如果有机会可以看看源码，但是我觉得现在也来不及。等社会发展科技进步也许能有更高效的密码存储机制，那时候就又是一种新型的技术了。

**七、参考文献**