

网络安全实验报告

题 目 基于口令的认证过程实现

专 业 视听觉信息处理 .

学 号 7203610121 .

学 生 刘天瑞 .

指 导 教 师 王彦 .

# 一、实验目的

掌握随机函数的使用，掌握散列函数，加解密函数的使用。包的格式与发包的顺序，可以等同于协议的三要素。掌握程序与数据库的链接。

**二、实验内容**

1. 客户端输入用户名，口令，随机产生认证码，使用散列函数计算用户名与口令的散列值1，使用散列值1与认证码计算散列值2，将用户名，散列值2，认证码明文传送到服务器端。
2. 服务器端以数据库（如access）保存用户名和散列值1的对应关系。收到客户端信息后，以同样的方法计算散列值2’。 如散列值2’=散列值2，则认证成功，成功后用散列值1加密认证码发送给客户端。客户端解密后写到指定文件。用户可以修改自己的密码。

# 三、实验过程

**基于口令的认证过程实现实验基本信息：**

实验环境：Ubuntu 20.04 x64

编程语言：C 语言

**1. 需求分析**

该实验需要分别编写服务器端和客户端的程序代码，模拟完成基于口令的认证服务过程实现。实现过程中的需要注意的重点有如下列所示：

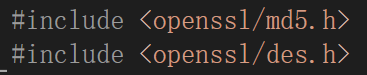
1. 散列函数与对称加密函数：客户端需要散列计算用户名、口令、散列值1与认证码，服务器端需要散列计算散列值1与认证码；二者都需要以散列值1为对称密钥加解密认证码；
2. 高级语言与数据库的链接以及对数据的存储和获取：该实验需要我通过编程来使用高级语言操纵mysql数据库，从而进行用户数据的查询等操作；
3. 模拟协议的三要素进行数据交互：模拟协议，设计数据的语法、语义和时序，保证通讯的双方能够得到期望得到的数据，而不会因为格式、语序等传输过程导致某一方得到失序或者错误的数据；
4. 随机函数。

**2. 程序结构**

**2.1 解决重要问题**

1.散列函数与对称加密函数：

使用openssl的MD5库作为散列函数；使用DES库作为对称加密函数，两头文件如下图所示：

****

2.高级语言与数据库的链接以及对数据的存储和获取：

在Ubuntu虚拟机上使用MySQL数据库对用户数据进行存储。链接数据库的头文件”mysql.h”来连接本地MySQL数据库如下图所示：



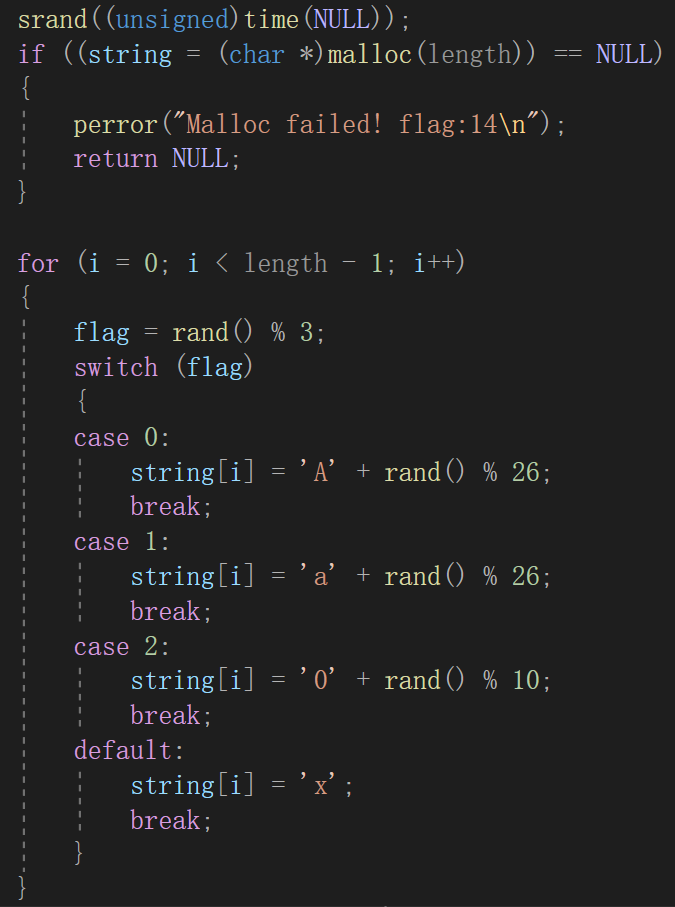
3.模拟协议的三要素进行数据交互：

在服务器端和客户端之间总共进行了三次数据交互如下表所示：

|  |  |  |
| --- | --- | --- |
| 发送方 | 接受方 | 数据内容 |
| 客户端 | 服务器端 | 用户名 |
| 服务器端 | 客户端 | “GET SEQ1” |
| 客户端 | 服务器端 | 认证码 |
| 服务器端 | 客户端 | “GET SEQ2” |
| 客户端 | 服务器端 | 散列值2 |
| 服务器端 | 客户端 | DEC散列值1（认证码） |

4.随机函数：

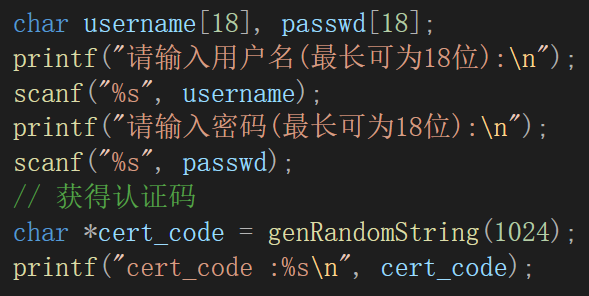
输入用户所期望得到的随机串string的长度length，其中调用malloc()函数来创造出足够的空间；以时间time作为随机数种子，将随机数伪平均地分配到所有字符。具体运行的部分关键代码如下截图所示：



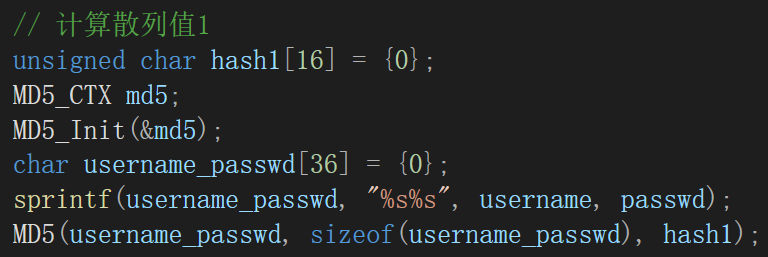
**2.2服务器端与客户端的传输代码实现**

**2.2.1.客户端**

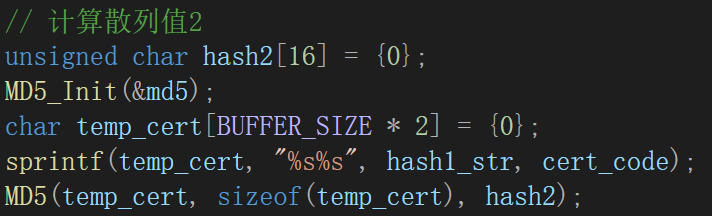
1）在客户端用户分别需要输入用户名和密码（最长到18位），使用随机函数获得1024 位长度的随机串作为认证码，如下图所示：



连接用户名和密码的字符串，使用MD5()函数进行散列操作得到散列值1如下图所示：



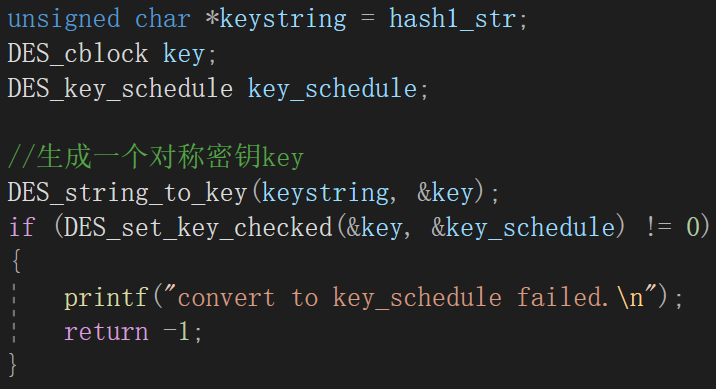
3）接着连接获得的散列值1与认证码，继而得到了散列值2如下图所示：

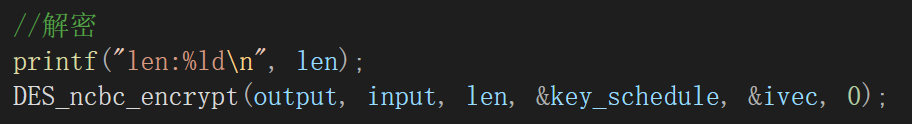


4）先创建套接字，再连接服务器端，按照模拟协议传输数据如下图所示：

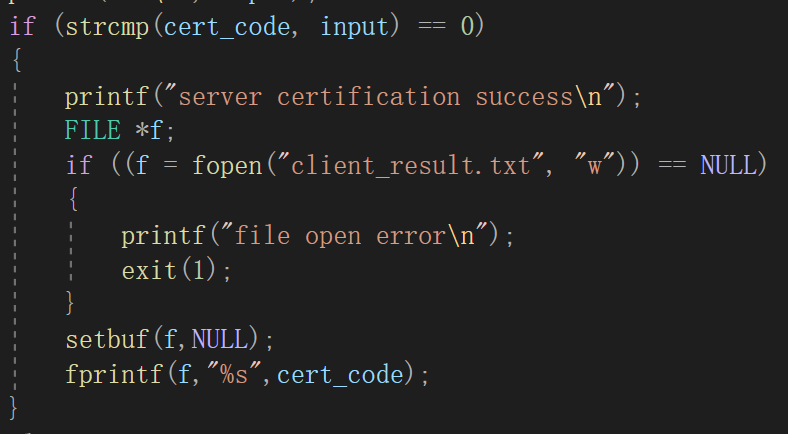


5）总共三次数据传输完毕后，再解密服务器端传输来的加密后的数据：使用散列值1来生成对称密钥key，用对称密钥key来解密传输过来的数据如下图所示：



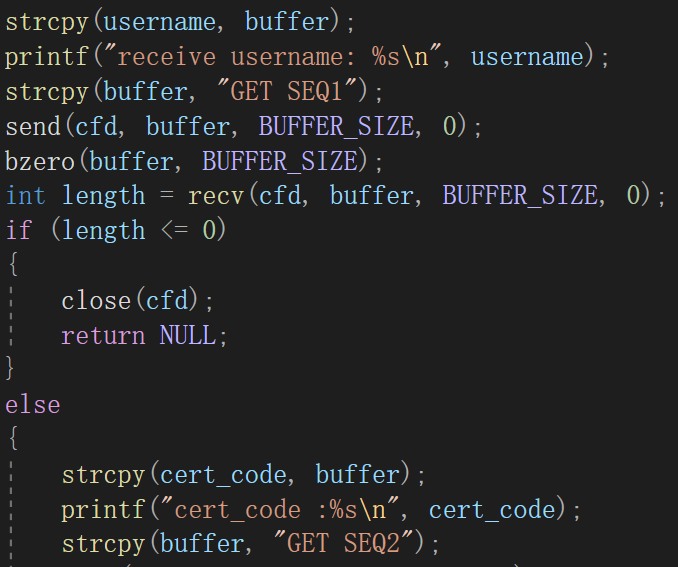


6）最后将解密后生成的数据与之前获得的认证码进行比较，如果相同则说明服务器端已认证成功，继而将认证码写入被指定的文本文件client\_result.txt，如下图所示：

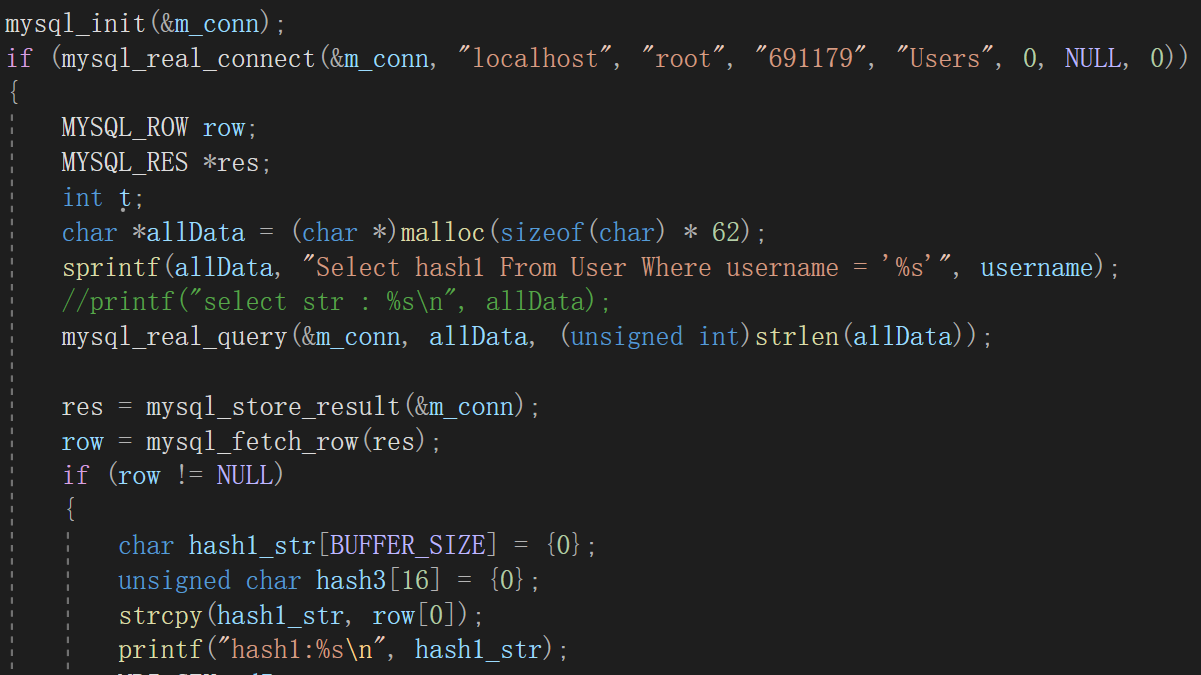


**2.2.2服务器端**

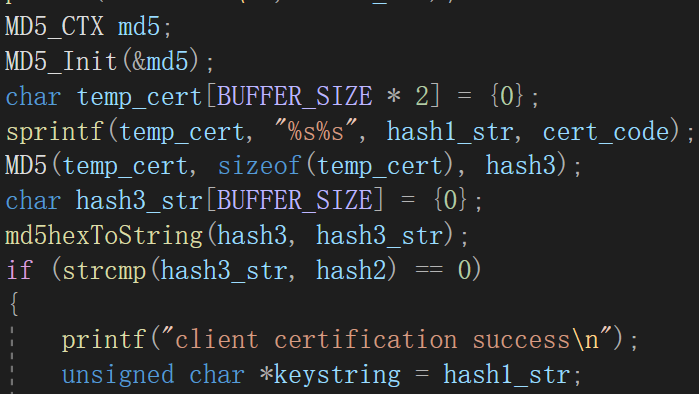
1）监听9999号端口，等待客户端的连接。如果客户端有请求连接，则创建子线程，转到子函数多线程来处理。在处理函数中，等待客户端传来的数据，并分别回应GET SEQn（其中n=1,2,3），如下图所示：



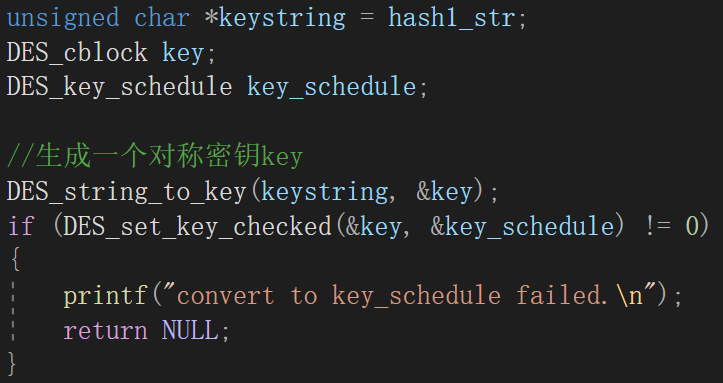
2）连接上本地的MySQL数据库，注意在mysql\_real\_connect()函数里可以随时对本地存储用户的密码进行修改，同时使用客户端传输来的用户名称来创建查询操作语句，用来找到服务器端的数据库中所存储的与该用户名称所对应的散列值1，如下图所示：

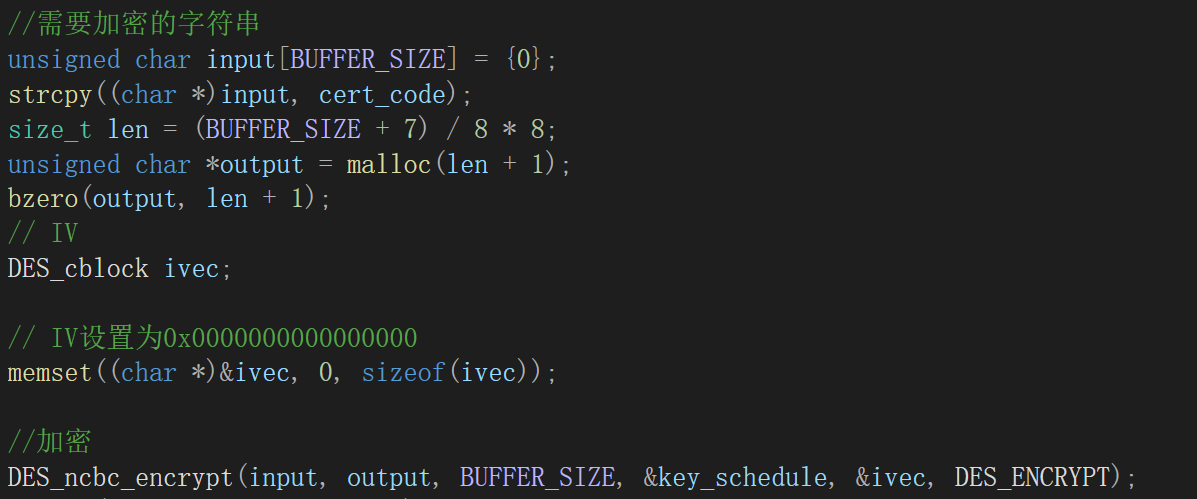


3）接着使用与客户端相同的算法来计算散列值2，并且与客户端传输步骤的第三步所传输来的散列值2进行比较，如果相同则用户名称与密码对应，用户认证成功，如下图所示：



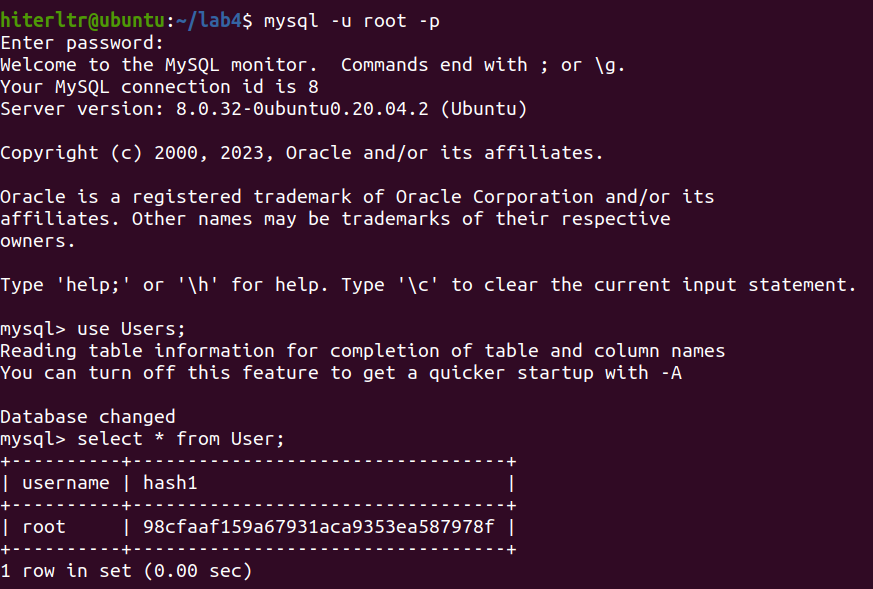
4）最后使用获得的散列值1来生成对称密钥key，加密认证码，加密完成后再发送回给客户端，如下图所示：





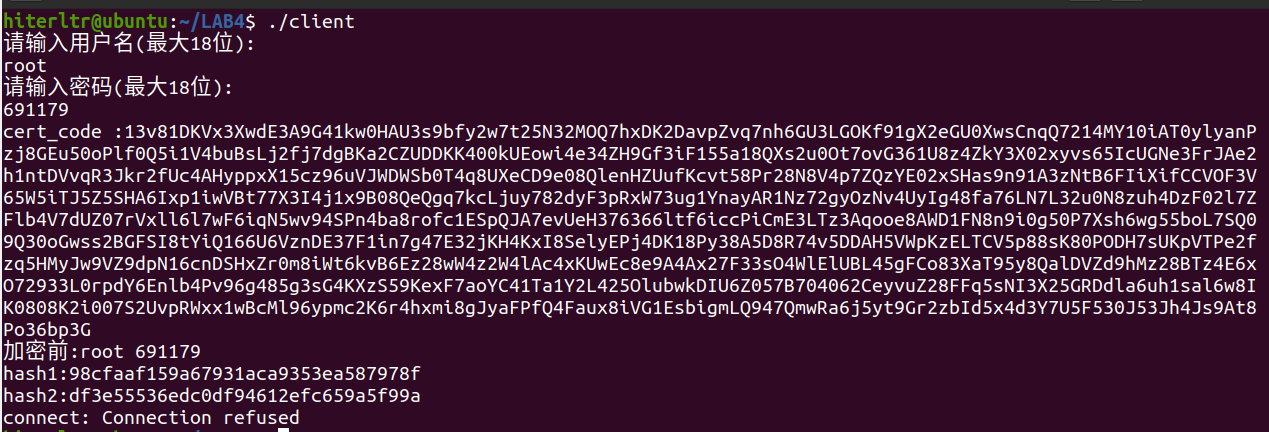
# 四、实验结果

服务器端的原虚拟机IP地址为192.168.176.132，所连接的本地数据库为MySQL，User库里存储用户名称为root，密码为691179，同时在服务器端运行启动程序./server.c如下图所示：

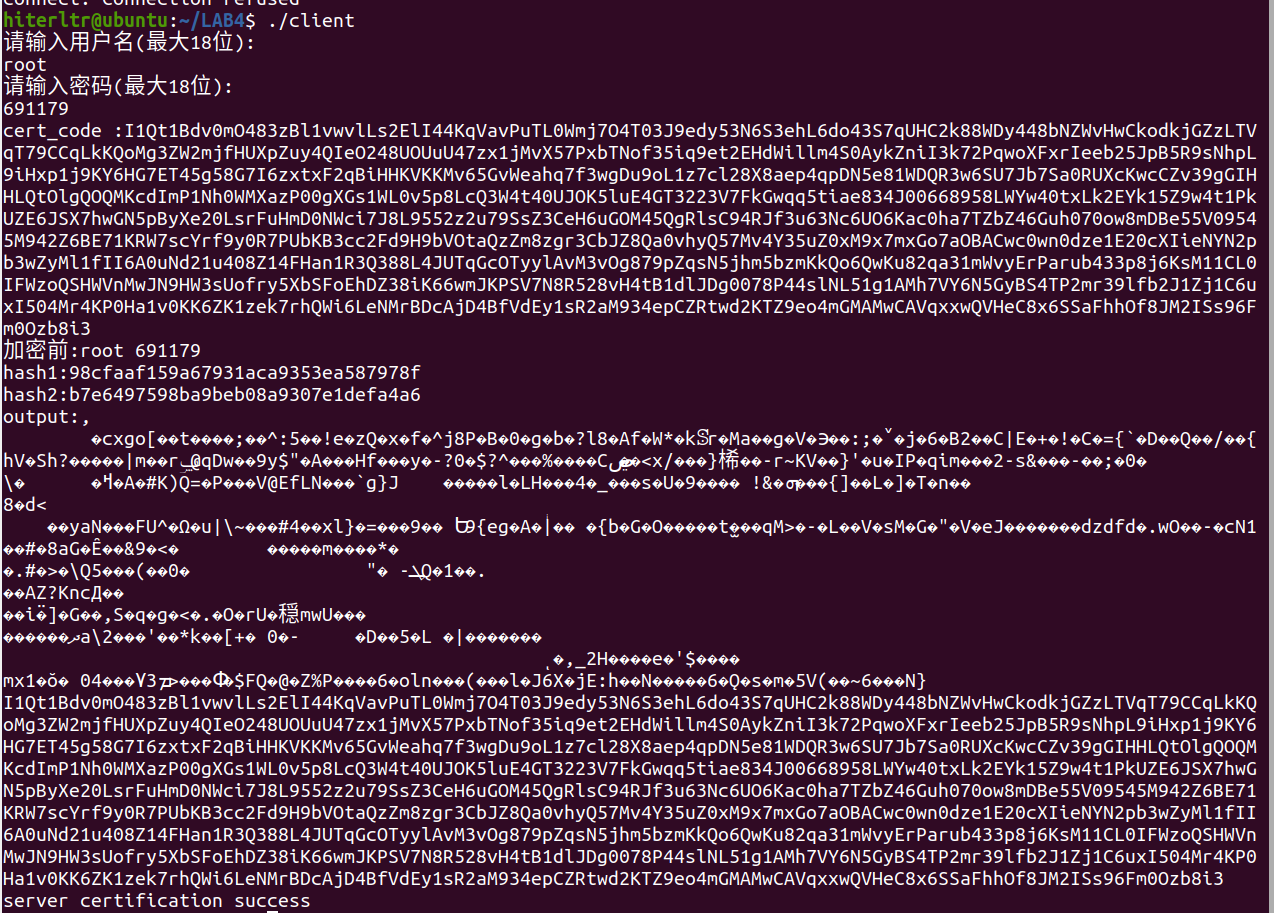


并且所存储的散列值表为98cfaaf159a67931aca9353ea587978f；

客户端的克隆虚拟机IP地址为192.168.176.130，运行启动服务端的程序./client，输入用户名称和密码（之后也可以随时对该存储的用户密码进行修改），可以得到加密前的散列值1（此时服务器端尚未启动，连接失败，但是可以获得加密前的散列值1并且需要将其存储到服务器端的MySQL数据库User中，即为哈希列表里的98cfaaf159a67931aca9353ea587978f），如下图所示：

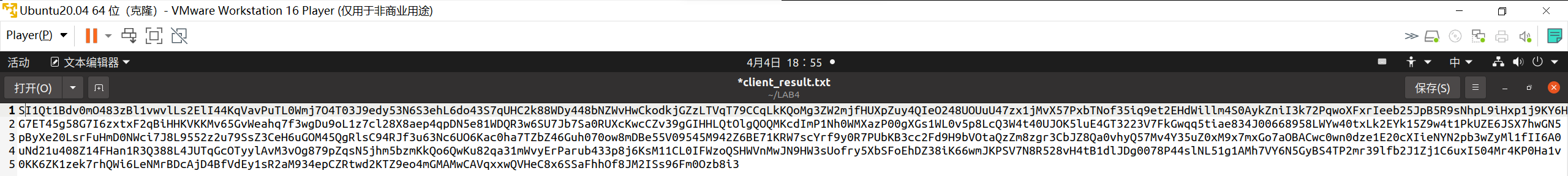


启动服务器端后，则如下图所示：



最后验证成功，在客户端的相同路径下也得到了所需输出的文本文件client\_result.txt，内含解密之后的认证码，如下图所示：





并且也可以得到服务器端的终端输出乱码结果如下图所示（注：server.c程序中所设置的多余printf只是为了更方便地查看通信双方的数据是否一致，完全可以注释掉）



# 五、心得体会 （出现问题分析）

通过本次实验，我不仅学习到了随机函数的编写，了解了散列函数、对称加密函数的具体操作过程中的使用方法，并且我还了解到了Openssl()库函数。除此之外，我通过定义数据包的格式与发送数据包的顺序来进行服务器端与客户端之间的多端通讯，从中甚至额外还掌握了源代码程序与本地MySQL数据库的链接，可谓收获颇丰。

出现问题分析：我在进行散列函数结果格式分析时发现结果为16位的无符号字符unigned char，即总共大小为16\*8位=128位，该得到的结果不并能作为字符串string进行输出，而需要再通过相关函数转为标准字符串，否则就会出现以上截图里的乱码情况。