实现动态口令验证，有无图形化界面均可

 程序a 模拟电子令牌每分钟变换显示口令

 程序b 模拟服务器端，通过网络接口接收口令并验证

 程序c 模拟客户端，点击验证后（或输入验证命令后），

1. 实现原理
2. 令牌生成

由程序a来实现，程序a在发放给用户使用前，会先和用户的账号进行一个绑定。服务器端会在该用户名下生成一个电子令牌seed，并将该seed写入到该电子令牌中。

电子令牌会产生6位数字，每分钟进行一次更新，令牌生成的的动态密码是由seed和当下时间哈希后得到的，即：

动态密码 = 取后六位(hash[种子+时间])

1. 服务端验证

由程序b实现。服务端收到来自，客户端的验证请求，先根据用户的id，找到对应用户密码哈希值和用户令牌种子，仅用户pin和动态密码拼接，让后进行哈希运算，再与用户发来的认证信息比较，如果一致则认证成功。

1. 客户端验证

由程序c实现。客户端会先将用户的pin进行一次哈希，让后拼接上令牌产生的动态密码，再次哈希，连同用户id一同发给服务器进行验证，即用户发送给服务器的认证值是如下形式：

认证值 = hash[ hash[PIN] + 动态密码]

1. 具体实现
2. 令牌生成

Seed ：一个11位的十进制整数，这里我们取做19300240012

时间： 选用1970纪元后经过的分钟数

Hash算法 ：MD5

生成过程就是利用seed和时间拼接为一个字符串，让后用MD5算法获得128位hash值，将该值转换为整数，取末尾6个数字即我们所需要的动态密码。

1. 服务端验证

PIN的hash生成算法：sha256

PIN的hash值和动态密码拼接后的hash生成算法： sha256

动态密码Hash算法：MD5

服务端与客户端连接连接方式：TCP

等候端口号为：12000

服务端要做的就是用自己存储用户信息，实时生成用户的认证值，让后和用户发来的认证值进行比对。

为了体现认证过程，服务端存放了一组用户信息信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 用户id | 时间seed | PIN的hash值 | PIN（服务端不存储） |
| 0 | yjr | 19300240012 | 2c70e12b7a0646f92279f427c7b38e7334d8e5389cff167a1dc30e73f826b683 | key |
| 1 | yjr1 | 19300240013 | 2c70e12b7a0646f92279f427c7b38e7334d8e5389cff167a1dc30e73f826b683 | key |
| 2 | yjr2 | 19300240014 | 2c70e12b7a0646f92279f427c7b38e7334d8e5389cff167a1dc30e73f826b683 | key |

1. 客户端验证

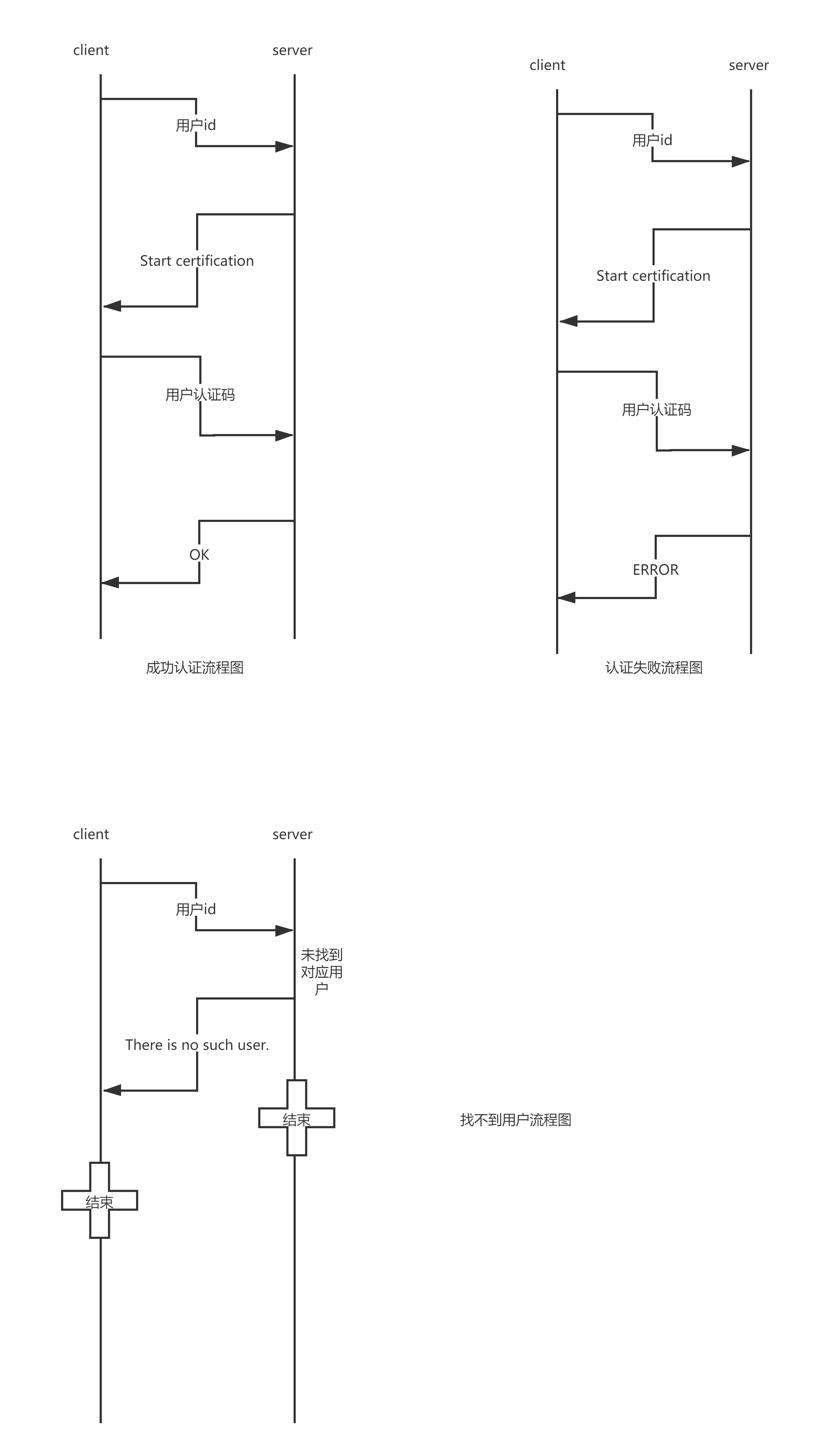
PIN的hash生成算法：sha256

PIN的hash值和动态密码拼接后的hash生成算法： sha256

服务端与客户端连接连接方式：TCP

客户端的逻辑和服务端的是一一对应的，只是不同计算时间令牌尝试的动态密码，但是需要接受用户的输入。

1. 服务端与客户端交互逻辑



1. 图形化界面实现

图形化框架：PySide2

实现的程序：

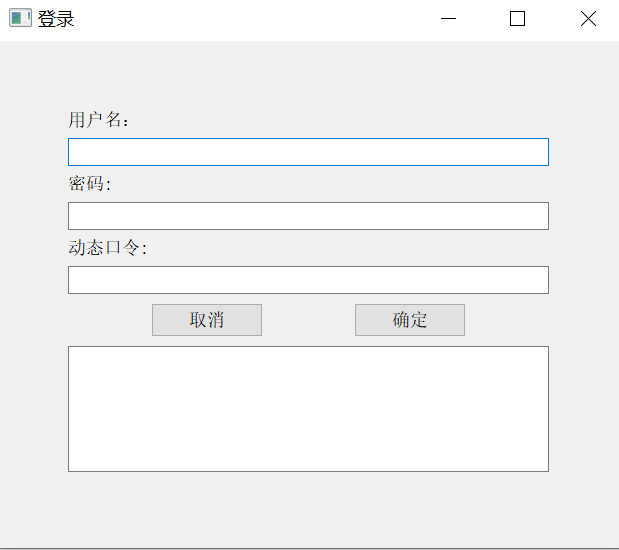
1. 电子令牌 token\_GUI.exe



该动态口令每秒刷新，实时显示出来。

另外该程序是与用户名为yjr的账号，绑定的，内置了yjr的时间seed，只能由yjr使用。

1. 客户端 client\_GUI.exe



注：服务端没有实现图形化界面，因为对服务端来说图形化界面没有必要。