项目使用truffle为框架，使用python的web3.py进行交互，使用ganache-cli作为客户端。

**环境需求：**

truffle 4.1.8

ganache-cli

python >= 3.6

python的Crypto库、solc库与web3库

NIZK库来源：https://github.com/Juzix/BlockChain-Demo

**环境安装指令：**

Truffle: npm install -g truffle@4.1.8

ganache-cli npm install -g ganache-cli

pip换源 ~/pip/pip.ini

Python相关库： pip install Crypto

pip install web3

pip install py-solc

**文件结构：**

AttriChain

--app 与合约交互代码

----DS.py ds数字签名的python实现

----main.py 与区块链交互的主体代码

--build 编译输出目录

----contracts

------合约编译结果

--contracts 合约目录

----nizk NIZK具体实现

------LibNIZK.sol NIZK库主体，所有可以调用的4个函数都在这里

------LIBNizkParam.sol NIZK工具函数，不要从外部调用

----utillib NIZK库的底层工具库

------LibInt.sol

------LibString.sol

----ChainInit.sol 链初始化合约

----LibDTBE.sol DTBE库

----test.sol 测试用合约

--migration 部署脚本

----1\_initial\_migration.js 这个脚本空着不用

----2\_depoly\_contracts.js 在这里写部署代码

**使用流程**

1. 在配置好环境的平台上，先启动ganache，

注意！！！必须使用-db参数导入已有的账户

ganache-cli -db ./ -g 100000000 -l 4294967295;

绝对不要直接使用ganache-cli直接启动客户端，会覆盖本地存储的账户！！

1. 编译并部署合约到链上

truffle migrate --reset;

1. 启动python应用

python3 main.py

**代码详细说明**

密码学工具的实现：

NIZK为github的库，DTBE自己实现，还在写，数字签名DS与TS都是用python完成，使用预留的接口对链上数据进行初始化。

1. Python部分：

DS.py使用实现sha256算法进行签名，封装好了3个函数。keyGen(), sign(), verify()。

TS的实现使用t+m=m`，然后将m`作为密文进行签名。

DS.py的signature\_test函数是一个简单的密钥分配，签名，验签的流程演示。

main.py中前26行完成了对运行在localhost:8545的区块链网络的连接，后续对区块链的操作都使用这一对象进行。

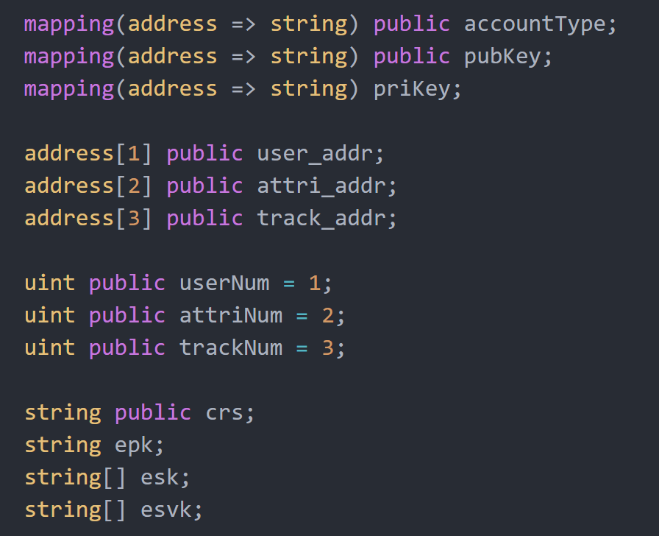
account\_list列表是本地存储的账户，注释已经写明了账户分别是什么类型的。

具体的账户与合约操作看注释。

1. Solidity部分

所有的链初始化代码均在ChainInit合约中，其中全局初始化与最终机构初始化在合约中完成，属性机构和用户初始化留好接口，在python中完成再通过调用接口将值写入。

合约变量说明：

accountType是从地址到类型的mapping，key为账户地址，value为三种取值 user, attri, track。

pubkey为各个账户的公钥

prikey为各个账户的私钥

user\_addr为所有用户账户的地址列表

attri\_addr为所有属性机构账户的地址列表

track\_addr为所有追踪机构账户的地址列表

三个num分别是三种账户的数量，

crs是NIZK初始化时输出的参数，epk, esk, esvk为DTBE初始化时输出的参数。

合约库说明

写在合约中的库只有nizk和dtbe，如果要使用ds或ts，请留好函数接口，在需要计算的时候交由web3.py返回相应数值。nizk和dtbe的使用直接import然后using …for \*。