# Wanchain早早期接口手册

## 账号地址管理

#### 基于account命令行的账号管理

./geth account new

./geth account list

#### 基于geth console的账号管理

> personal.newAccount("passwd")

生成的wanchain格式的keystore文件里面多个一个crypto2字段.

#### 新地址的编码格式

以太坊原生的地址其实是哈希, 但是我们需要的是两个公钥. 两个公钥可以简单的理解为A.X A.Y B.X B.Y. 因为椭圆曲线根据Y轴对称,所以根据X可以计算出Y(实际是根据Y轴对称的两个Y), 所以新的地址格式可以简化为0x + A.X + B.X + (A.Y的正负号)+(B.Y的正负号)+checksum.

另外这个新地址要储存在keystore文件里. 否则计算此地址需要解锁私钥太麻烦了.

具体的工作留到下一期. 目前直接使用A.X A.Y B.X B.Y 用’+’简单拼接起来.

## 原生币的隐私转账

只支持基于console的命令行模式

#### 产生一次性地址

目前产生一次性地址比较复杂,具体步骤如下

1. 获取一个账号的主地址 GetPublicKeysRawStr()

> personal.unlockAccount("0x2d0e7c0813a51d3bd1d08246af2a8a7a57d8922e")

Unlock account 0x2d0e7c0813a51d3bd1d08246af2a8a7a57d8922e

Passphrase:

true

> eth.getPublicKeysRawStr("0x2d0e7c0813a51d3bd1d08246af2a8a7a57d8922e")

"0xd7dffe5e06d2c7024d9bb93f675b8242e71901ee66a1bfe3fe5369324c0a75bf+0x6f033dc4af65f5d0fe7072e98788fcfa670919b5bdc046f1ca91f28dff59db70+0x4d67960ead4e4d7ae4c454b3938ffd638aa617b617e0e8467b6369b213fcb7d6+0x30bfb791e895b6a53cf6d6f9ae4570f007c23b0d1ffdc85cb9f20d11fac38eeb"

>

说明: 原来的以太坊地址是一个哈希. 我们的地址是两个公钥. 目前格式是简单的把A.X A.Y B.X B.Y简单的拼接起来. 以后要给出一个编码格式.

1. 根据主地址生成一次性地址(GenerateOneTimeAddress)

> eth.generateOneTimeAddress("0xd7dffe5e06d2c7024d9bb93f675b8242e71901ee66a1bfe3fe5369324c0a75bf+0x6f033dc4af65f5d0fe7072e98788fcfa670919b5bdc046f1ca91f28dff59db70+0x4d67960ead4e4d7ae4c454b3938ffd638aa617b617e0e8467b6369b213fcb7d6+0x30bfb791e895b6a53cf6d6f9ae4570f007c23b0d1ffdc85cb9f20d11fac38eeb")

"0xe39d139cc5fb43ec102fd8cc00a08ebb85abd0457f10f5cd559baef5bebe0839+0xa31f876213dcf3866c4f7502650a086f9abe33a9f92269efd61286d63fd4c66a+0xb3a4dc5d4c38417b472098416db828d52d986bbcbc825d15bb07949e42f3beb0+0x9b953f1028ded8704b5120a24bb2084dbb0ec76dc10ff8e1c3e0441caf3e3182"

>

#### 原生币隐私转账用到的合约

合约可以有三种生成方式

1. 预编译在程序中.
2. 通过solidity编译生成
3. 用solidity编译,但是调用预编译在程序中的函数.

目前采用第一种方式

#### 原生币转账函数接口

(老贾)

#### 扫描属于某账户的一次性地址

ScanOTAbyAccount(address)

#### 获取某一面值的一系列一次性地址

(Frank)

#### 提币函数接口