ETH Web钱包

无业游民唐朝 suibingyue@gmail.com

初入链圈，多多关照，如有错误|误解，见谅

项目技术栈：

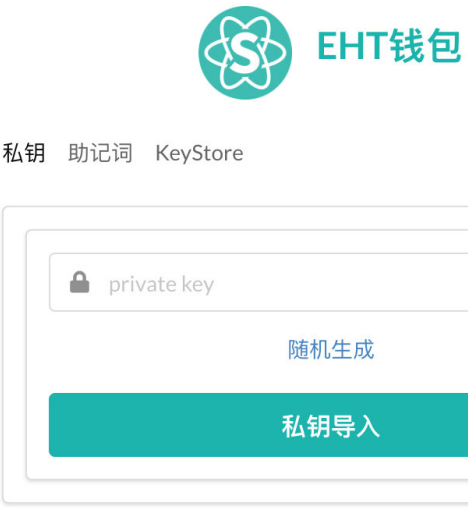
- 公钥、私钥、钱包概念
- HD钱包、BIP32、BIP39、BIP44协议
- nodejs
- html 、css、react
- semantic-react
- ether.js
- golang

环境支持

- ganache客户端
- ganache-cli 命令行客户端 (会通过助记词导入即可)
- geth 可选 （会获取账户列表、余额即可）

1、功能预览

- 网页钱包
- 通过私钥、助记词、创建账户
- 通过私钥，助记词，keyStore导入账户
- 显示钱包信息（地址、余额、交易次数等）
- 钱包转账
- 钱包备份


EHT钱包
私钥 助记词 KeyStore
private key
随机生成
私钥导入


EHT钱包
私钥 助记词 KeyStore
12 words
m/44'/60'/0'/0/0
随机生成
助记词导入

EHT钱包

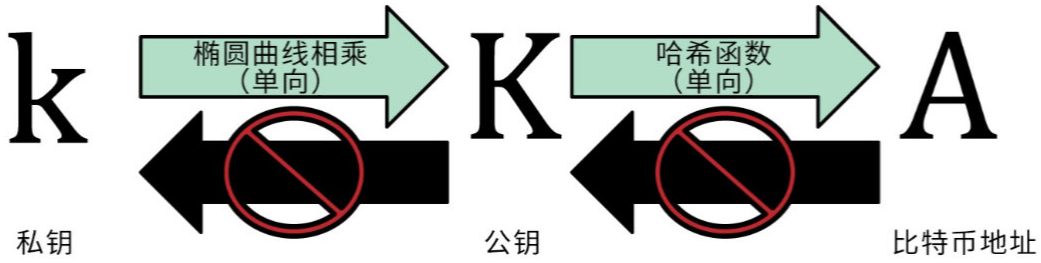
私钥　助记词　KeyStore

keystore为json格式

••••••••••

导入

2、钱包概念



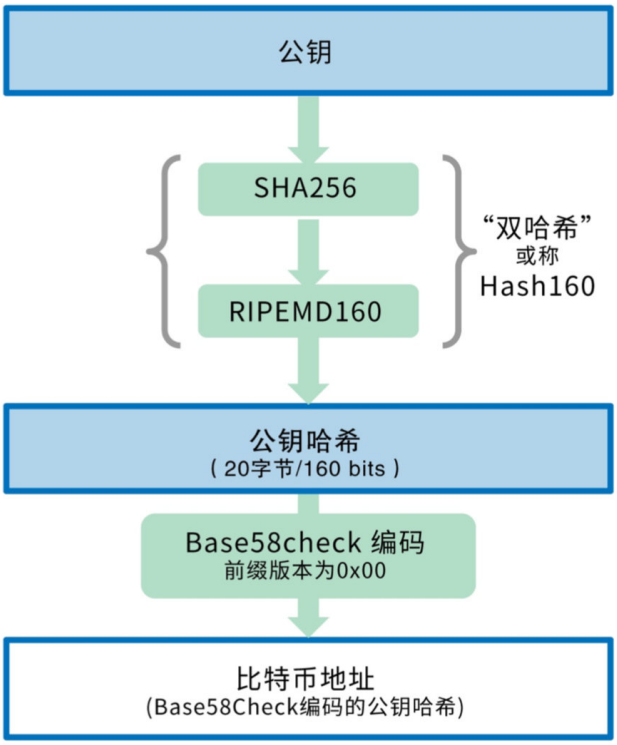
钱包：容器，管理账号私钥的工具

私钥公钥：一个随机数字进行密码学运算得到私钥，公钥

地址：数字字母组成的字符串，由公钥推导而来

3、创建账号复习

比特币钱包创建代码，可查看学过的比特币项目




```
// go比特币中的钱包
type WalletKeyPair struct {
    PrivateKey *ecdsa.PrivateKey
    PublicKey  []byte
}

func newWalletKeyPair() *WalletKeyPair {
    // 椭圆曲线函数得到私钥
    privateKey, err := ecdsa.GenerateKey(elliptic.P256(), rand.Reader)
    if err != nil {
        panic(err)
    }
    // 私钥得到公钥
    publicKey := append(privateKey.X.Bytes(), privateKey.Y.Bytes()...)

    return &WalletKeyPair{PrivateKey: privateKey, PublicKey: publicKey}
}

// 通过公钥得到地址
func (w *WalletKeyPair) getAddress() string {
    publicHash := HashPublicKey(w.PublicKey)
```

```
    // 1字节版本号
    version := 0x00

    // 得到21字节的数据
    payload := append([]byte{byte(version)}, publicHash...)

    // 4字节校验码
    checksum := CheckSum(payload)

    //25字节
    payload = append(payload, checksum...)

    address := base58.Encode(payload)
    return address
}

//ripemd160 进行hash运算
func HashPublicKey(pubKey []byte) []byte {
    hash := sha256.Sum256(pubKey)

    //创建一个hash160对象，向hash160中write数据
    //做哈希运算

    rip160Haher := ripemd160.New()
    _, err := rip160Haher.Write(hash[:])

    if err != nil {
        log.Panic(err)
    }

    publicHash := rip160Haher.Sum(nil)
    return publicHash
}

// 获取校验码
func CheckSum(payload []byte) []byte {
    first := sha256.Sum256(payload)
    second := sha256.Sum256(first[:])
    checksum := second[0:4]
    return checksum
}
```

创建钱包方式

- 随机生成私钥
- HD推导

4、BIP协议

BIP:Bitcoin Improvement Proposals 比特币改进建议

BIP32 通过一个随机种子,提出的为了避免管理一堆私钥的麻烦提出的分层推导方案。HD钱包（Hierarchical Deterministic Wallets）

BIP39 通过定义助记词让种子的备份更友好。

BIP44 BIP32的分层增强了路径定义规范，同时增加了对多币种的支持。

ganache钱包示例，如何备份?

```
Available Accounts
==================
(0) 0xfef3d415f66464c3b38e10fd5f31edbead7be44b (~100 ETH)
(1) 0xfc26f518d2f7091667dbdd81ee04d1f17d122359 (~100 ETH)
(2) 0x5e7363aa3c0669083a554dde5ed548a8ec90ff12 (~100 ETH)
(3) 0x57060a8a16bff2615769282eb83d1b50891f04a9 (~100 ETH)
(4) 0xc1f109c747e70bbc85371bcb6fbdc8fe23219da9 (~100 ETH)
(5) 0x3d25841411dd7917c123d980f4dd33cad101cc31 (~100 ETH)
(6) 0x9b477be361d60597e24dd7838d29f706396a3fa1 (~100 ETH)
(7) 0x8adffcabe036474de3a6d6f513bfd6df19fbcc1f (~100 ETH)
(8) 0x2394c966264c3794247136637e0dc9924dfad3d7 (~100 ETH)
(9) 0xbf513ae069d7a58eb4d0f8c6e17402dbe2cc1bee (~100 ETH)

Private Keys
==================
(0) 0x7c70eaea87e3d568bfcc8758ef9038b2ec640bc86130742a9a5154c5487bb033
(1) 0x7591240850e02d42f00f967aa836bf80c85a8022549a3a37fdc45efdc07db310
(2) 0xd29e3e156168928940d2f2f15a30821b4513919ec0d46aeb3df6c9eb98039ba3
(3) 0xfeb9ea90fe450517044c3d36d199aa2ee8fb81814fce1e1b9f470ebbe7a4fbb0
(4) 0x34c8c7173a5ef0b723aae65a6819aff5c740a8d73a47f1beba8b6e6a31529cc2
(5) 0xfa1cc9d7fd124bfa67b5efc59ef8563139690a634af1e74434902d604228aec8
(6) 0xb98ec5a0902400cc3248046fd869f6e533ec9e05af63f72ca98d2618c6d92b58
(7) 0x3d18f2267377dbafa797aa2e59382e60bee47368c234b5e14e4c3cab40b2adce
(8) 0xc09ceed4ddb30bffe0626c1c2663fe5acf80e0f1eb9fa8eeed09172b84c00faa
(9) 0xcd2d8a951d2d94c264ba8ad86e23d4751a3ede513131bf19b5c7f566443d8af0
```

HD &BIP32 提议

BIP:

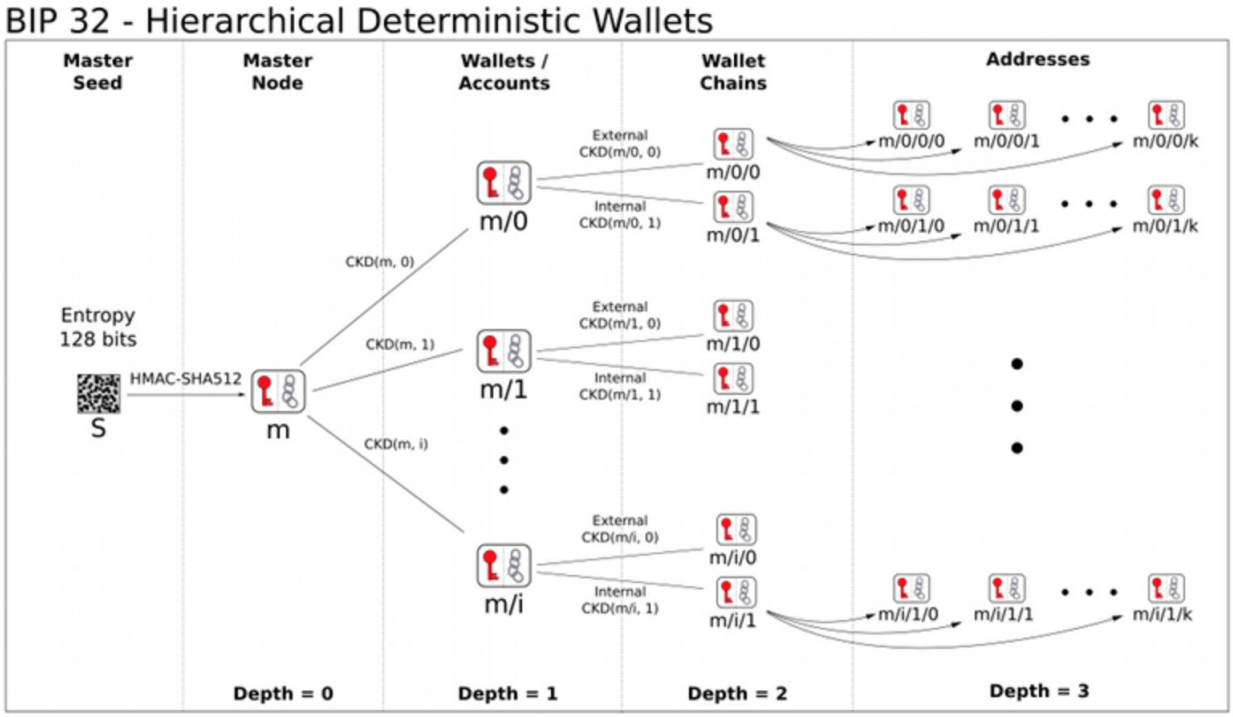
根据一个随机数种子通过推导方式得到n个私钥。存储仅需一个种子就可以推导出私钥

https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

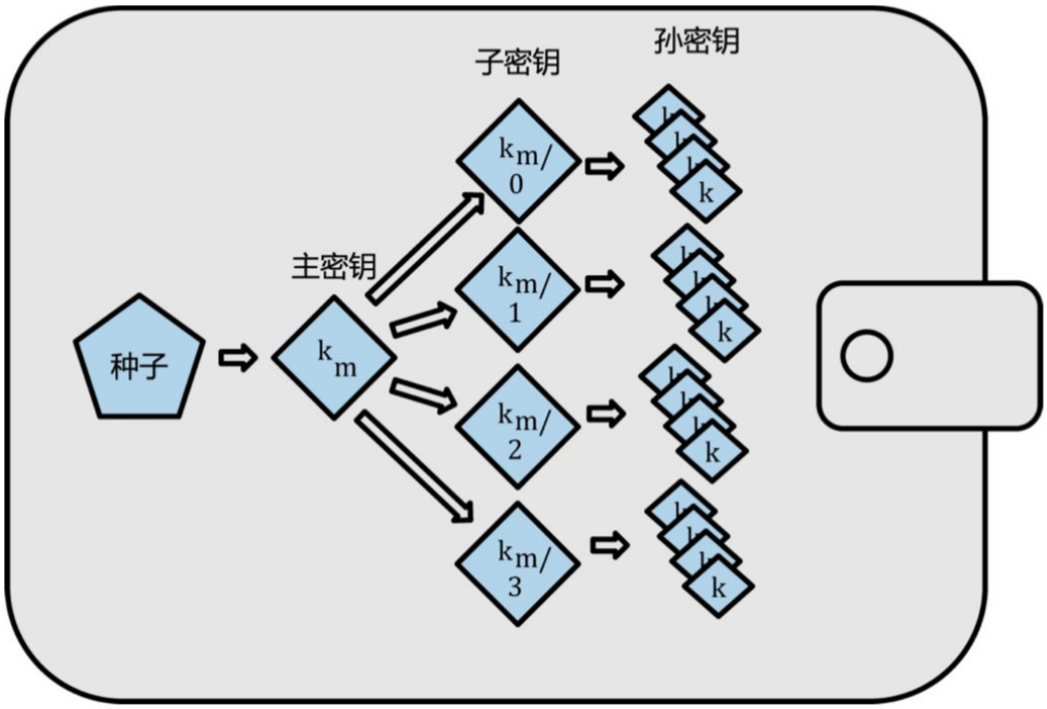
协议名称：Hierarchical Deterministic Wallets-HD

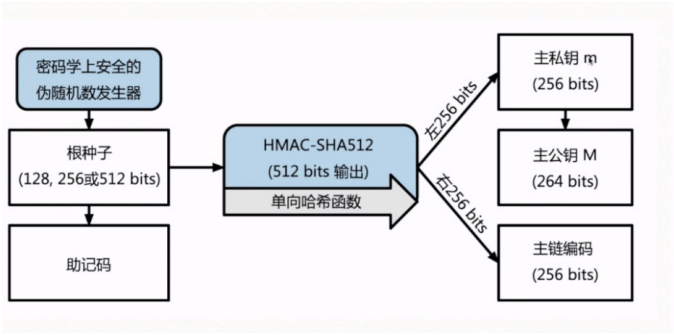
- 更好的保护隐私性

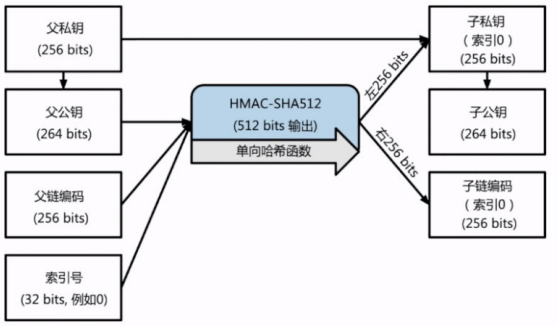
- 多账号备份麻烦，如果钱包100个账号



hd钱包推导过程







BIP39-助记词

https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

BIP32 避免保存一个用户的多个私钥，可以仅保存一个随机种子，而随机种子为一串16进制数据，进行冷备份不变。

BIP39 :使用助记词通过算法生成种子，这样用户只需要记住12个单词即可(12-24个)

随机种子

```
b052ba6a0f2212a003c6b59739ddfff1ee54031041a30a341ab7f26c7319807ca12233e1cc0f8948573e9a83f4e3bd89313627de852f41f719b481af69bb0adb
```

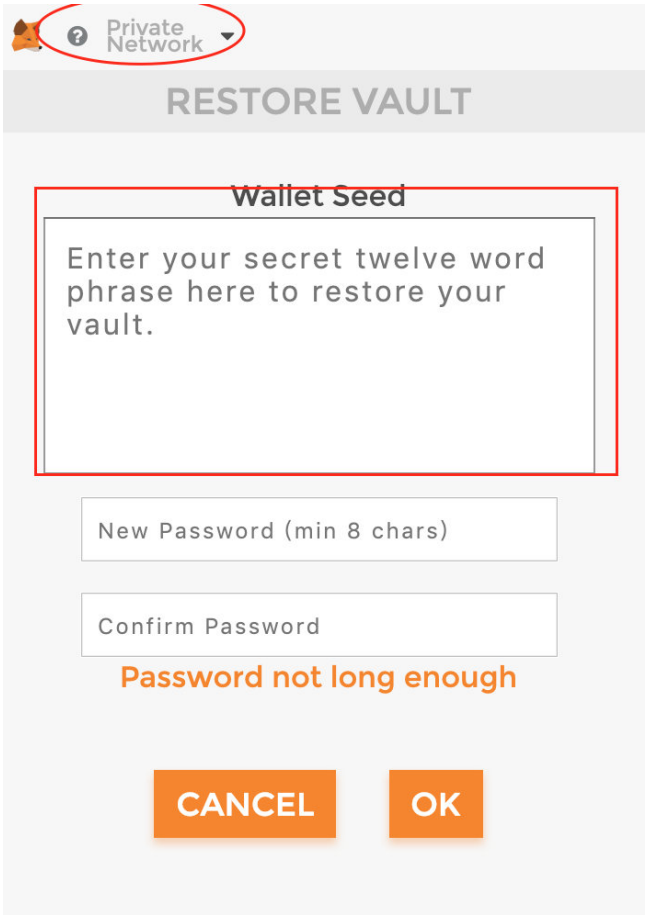
助记词

```
disorder timber among submit tell early claw certain sadness embark neck salad
```

回顾 我们已经用过的,启动ganache-cli命令

```
ganache-cli -m 'disorder timber among submit tell early claw certain sadness embark neck salad'
```

回顾 小狐狸导入账户



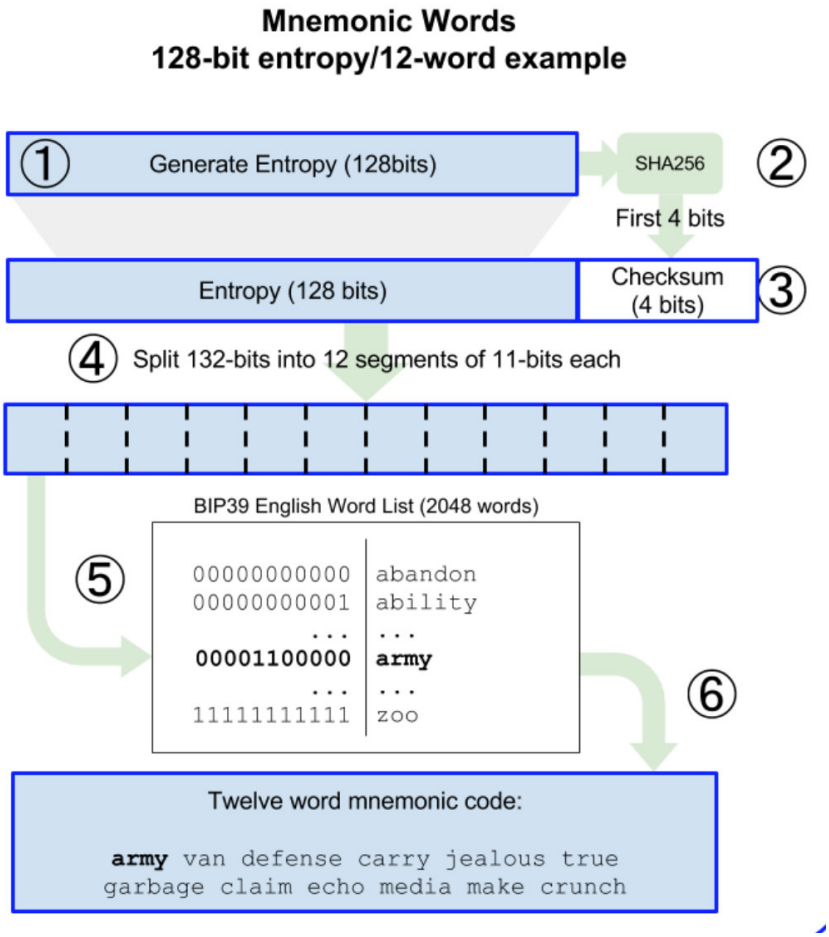
导入成功

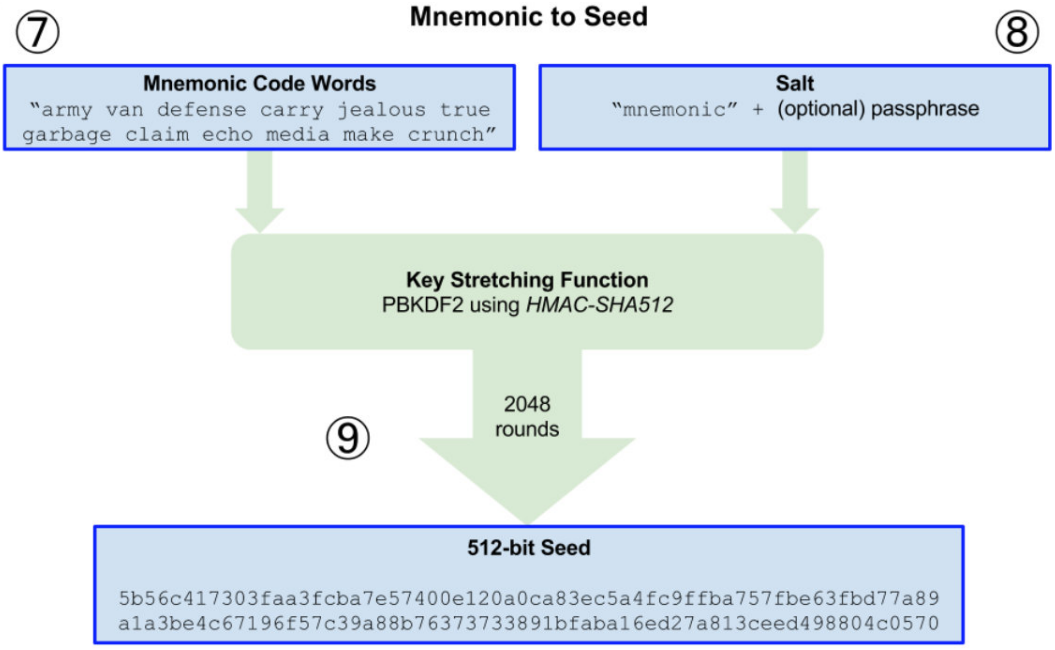
```
➜  ~ ganache-cli -m 'disorder timber among submit tell early claw certain sadness

Ganache CLI v6.2.4 (ganache-core: 2.3.2)

Available Accounts
==================
(0) 0xfef3d415f66464c3b38e10fd5f31edbead7be44b (~100 ETH)
(1) 0xfc26f518d2f7091667dbdd81ee04d1f17d122359 (~100 ETH)
(2) 0x5e7363aa3c0669083a554dde5ed548a8ec90ff12 (~100 ETH)
(3) 0x57060a8a16bff2615769282eb83d1b50891f04a9 (~100 ETH)
(4) 0xc1f109c747e70bbc85371bcb6fbdc8fe23219da9 (~100 ETH)
(5) 0x3d25841411dd7917c123d980f4dd33cad101cc31 (~100 ETH)
(6) 0x9b477be361d60597e24dd7838d29f706396a3fa1 (~100 ETH)
(7) 0x8adffcabe036474de3a6d6f513bfd6df19fbcc1f (~100 ETH)
(8) 0x2394c966264c3794247136637e0dc9924dfad3d7 (~100 ETH)
(9) 0xbf513ae069d7a58eb4d0f8c6e17402dbe2cc1bee (~100 ETH)

Private Keys
==================
(0) 0x7c70eaea87e3d568bfcc8758ef9038b2ec640bc86130742a9a5154c5487bb033
(1) 0x7591240850e02d42f00f967aa836bf80c85a8022549a3a37fdc45efdc07db310
(2) 0xd29e3e156168928940d2f2f15a30821b4513919ec0d46aeb3df6c9eb98039ba3
(3) 0xfeb9ea90fe450517044c3d36d199aa2ee8fb81814fce1e1b9f470ebbe7a4fbb0
(4) 0x34c8c7173a5ef0b723aae65a6819aff5c740a8d73a47f1beba8b6e6a31529cc2
```

助记词流程:

1. 生成一个128位随机数

2. 对随机数做的校验4位
3. 随机数+校验码=132位数
4. 按每11位做切分，得到array[12]
5. 助记词表中查询
6. 得到12个单词为助记词
7. 通过助记词+salt(mnemonic+可选密码)
8. 进行HMAC-512哈希
9. 得到512bit的随机种子





单词表（**支持中文**）

https://github.com/bitcoin/bips/blob/master/bip-0039/bip-0039-wordlists.md

demo

BIP39协议-js

https://www.npmjs.com/package/bip39 6个函数

```
npm install --save bip39
```

demo-01bip39.js

```
const bip39 = require('bip39')

var wordList = bip39.wordlists.chinese_simplified

// 生成助记词
var mmic = bip39.generateMnemonic(128, ctypto.randomBytes, wordList)
console.log("生成的助记词为：", mmic)

var mmicHex = bip39.mnemonicToSeedHex(mmic)
console.log("生成的助记词为：", mmicHex)

// 通过助记词生成种子
var seed = bip39.mnemonicToSeed(mmic, "your password")
console.log("随机种子：", seed)

var validate = bip39.validateMnemonic(mmic,wordList)
console.log("合法？", validate)
console.log("修改后合法？", bip39.validateMnemonic(mmic + " ",wordList))
```

```
// 助记词转换
if (validate) {
    var hex = bip39.mnemonicToEntropy(mmic,wordList);
    console.log("加密后的助记词: ", hex)
    mmic = bip39.entropyToMnemonic(hex,wordList)
    console.log("转换后的助记词: ", mmic)
}
```

执行结果

```
生成的助记词为:  丝 卵 呼 碍 载 快 赖 邦 权 个 督 吾
生成的助记词为:
589044554643555c78b40c7b4aa756f018f03e16d4b2bf5bb1af5a8799b512cec350b129fa
60186267478ebcc462cefbb3bba5e7273017465cd15bf3
随机种子:  <Buffer bf 87 c9 21 49 8a 06 c6 8b eb 0d 96 87 66 ff a1 cf af 14
49 cb 21 99 2d 83 45 53 da0 21 21 4b f7 17 9c 89 68 a3 6d 4c b0 e2 15 39
20 f1 11 a9 d3 ... >
合法?  true
修改后合法?  false
加密后的助记词:  589991cf5cd56671383566256039c5f6
转换后的助记词:  丝 卵 呼 碍 载 快 赖 邦 权 个 督 吾
```

BIP39协议-go-浏览即可

- bip32 分层确定性钱包 创建账号
- bip39 助记词 备份账号
- eth账户导入导出的功能
- go语言编写，基础回顾

BIP32相关函数

```
// 生成随机数种子
seed, _ := bip32.NewSeed()

// 生成 masterkey
masterKey, _ := bip32.NewMasterKey(seed)

// 通过masterkey得到privatekey
childIdx = 0;
privateKey, _ := masterKey.NewChildKey(childIdx)
//通过childkey 得到publickKey
publicKey := privateKey.PublicKey()
```

demo-bip32.go

```
/*
@Time   :   2018-12-27 01:32
```

```go
@Author :   suibingyue
@File   :   demo-bip32 分层确定性钱包
*/

package main

import (
    "fmt"
    "github.com/tyler-smith/go-bip32"
)

func main() {
    // 随机种子
    seed, _ := bip32.NewSeed()
    // 生成 masterkey
    masterKey, _ := bip32.NewMasterKey(seed)
    fmt.Println("masterKey:", masterKey)

    accounts0 := genAccountsFromMasterKey(masterKey)
    genAccountsFromMasterKey(accounts0[0])
}

// 根据种子生成账户
func genAccountsFromMasterKey(masterKey *bip32.Key) []*bip32.Key {

    // 存储账户key
    var privKeys []*bip32.Key
    var pubKeys []*bip32.Key

    for i := 0; i < 3; i++ {

        // 得到childkey
        privKey, _ := masterKey.NewChildKey(uint32(i))
        //通过childkey 得到publickKey
        pubKey := privKey.PublicKey()

        privKeys = append(privKeys, privKey)
        pubKeys = append(pubKeys, pubKey)
    }

    fmt.Println("\n Public keys, 深度为", pubKeys[0].Depth)
    for index, key := range pubKeys {
        fmt.Printf(" %d %x\n", index, key.Key)
    }

    fmt.Println("\n Private keys 深度为", pubKeys[0].Depth)
    for index, key := range privKeys {
        fmt.Printf(" %d %x\n", index, key.Key)
    }
```

```
    return privKeys
}
```

bip39相关函数

```
    // 生成随机数
    entropy, _ := bip39.NewEntropy(256)
    // 生成助记词
    mmic, _ := bip39.NewMnemonic(entropy)
    // 根据助记词密码生成种子
    seed := bip39.NewSeed(mmic, "suibingyue")
```

demo-bip39.go

```
package main

import (
    "fmt"
    "github.com/tyler-smith/go-bip32"
    "github.com/tyler-smith/go-bip39"
)

func main() {
    // 生成随机数
    entropy, _ := bip39.NewEntropy(256)

    // 生成助记词
    mmic, _ := bip39.NewMnemonic(entropy)
    fmt.Println("生成的助记词 ", mmic)

    //根据助记词密码生成随机种子
    seed := bip39.NewSeed(mmic, "")
    fmt.Printf("生成的 seed %x \n", seed)
    // 根据随机种子生成主key
    masterKey, _ := bip32.NewMasterKey(seed)
    fmt.Printf("生成的 masterKey %x \n", masterKey.Key)

    fmt.Println("===电脑不安全，写进我的日记本里====")

    fmt.Println("导入的助记词 ", mmic)
    seed2 := bip39.NewSeed(mmic, "")
    fmt.Printf("导入的seed %x \n", seed2)

    masterKey2, _ := bip32.NewMasterKey(seed2)
    fmt.Printf("导入的 masterKey %x \n", masterKey2.Key)
}
```

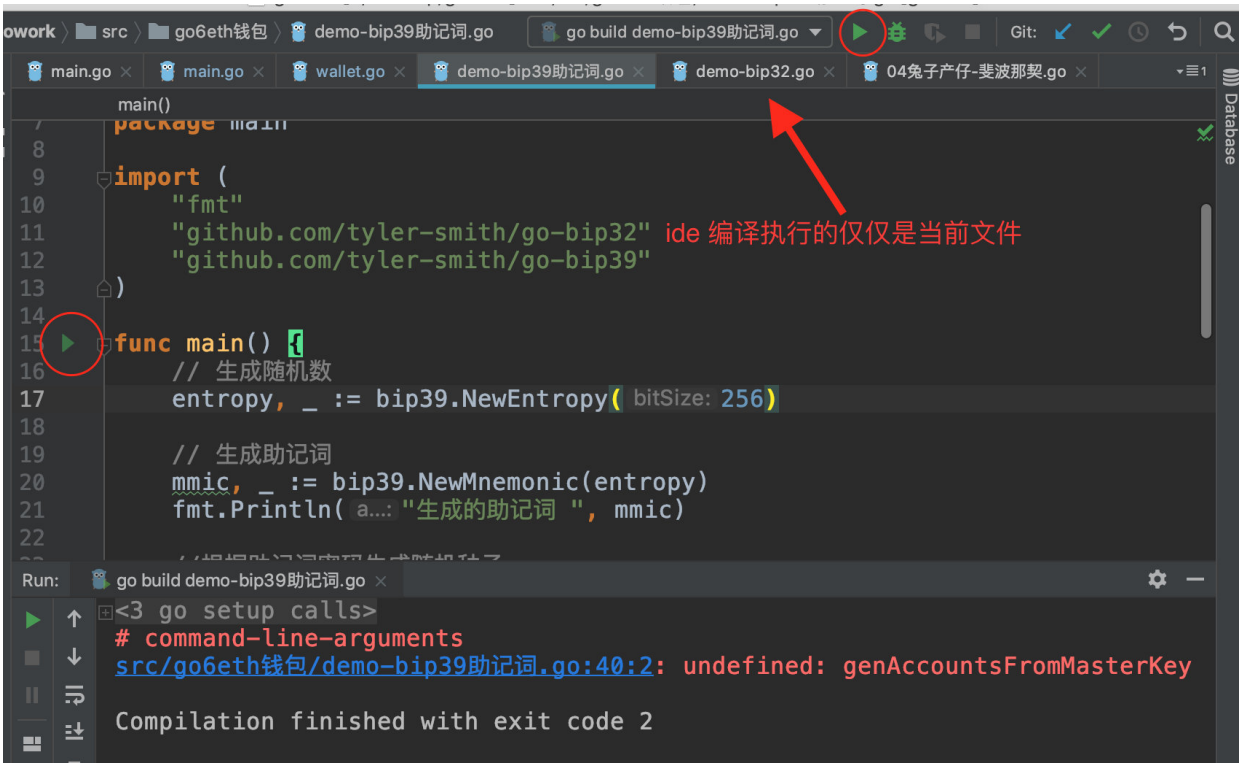
执行结果

有了masterKey，我们就可以使用BIP32协议，根据mastKey创建HD分层账号

同一个包下，执行失败?

```
<3 go setup calls>
# command-line-arguments
src/go6eth钱包/demo-bip39助记词.go:40:2: undefined: genAccountsFromMasterKey

Compilation finished with exit code 2
```

原因分析：go文件编译运行过程 build -> run



解决办法 进入到根目录，执行**go build *.go**，执行完成后，会生成两个可执行文件，执行即可

或者**go run *.go**

注意，**同一个包下只能有一个main函数**

```
➜  gowork cd src/go6eth钱包
➜  go6eth钱包 go build *.go
➜  go6eth钱包 ls
demo-bip32              demo-bip39助记词.go
demo-bip32.go           go6eth钱包
➜  go6eth钱包 ./go6eth钱包
```

BIP44 规范

为分层确定性钱包定义的一个逻辑层次结构规范，大部分币种，基于BIP32

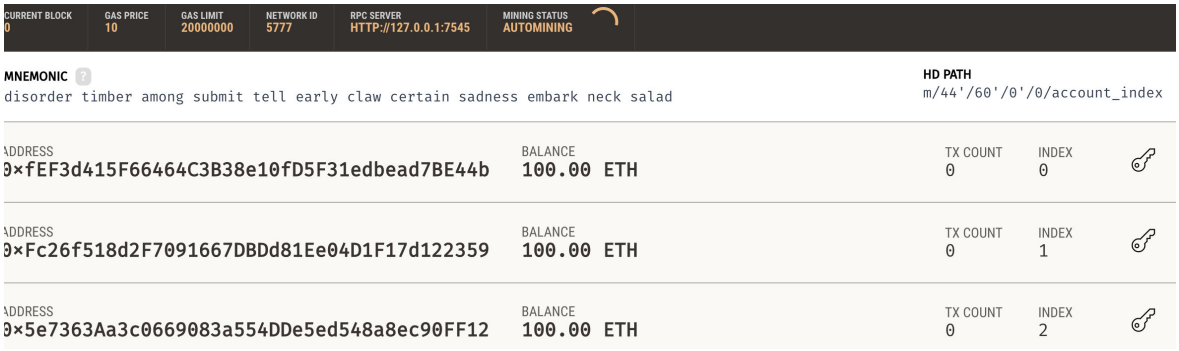
https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki

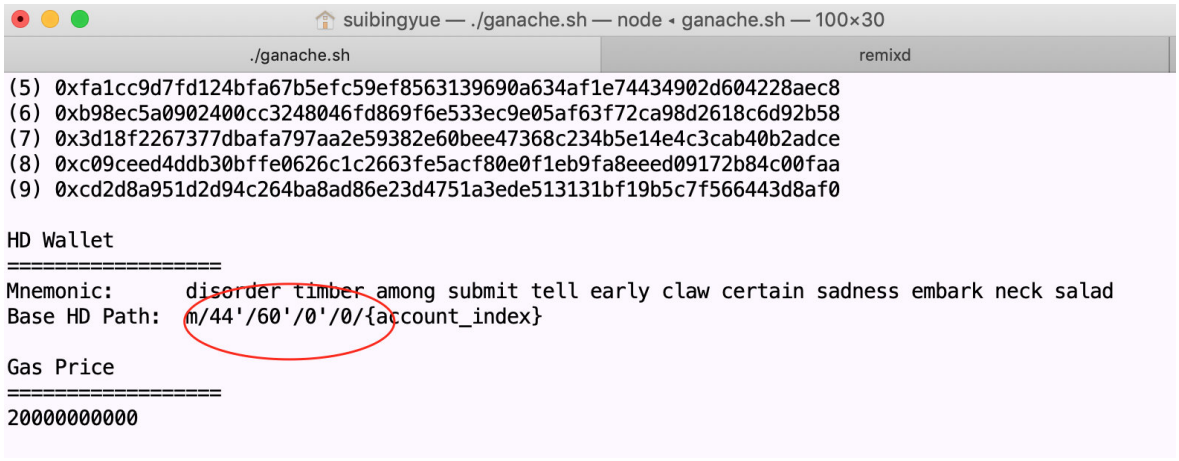
Path levels:

```
m / purpose' / coin_type' / account' / change / address_index
```

- purpose' : 44
- coin_type: 币种 **btc 01；eth 60，61**
- account: 账号分割标识
- change：0 外部账户 1 内部账户

  coinType查询：https://github.com/satoshilabs/slips/blob/master/slip-0044.md





有了HD分层钱包，钱包备份就无需备份多个私钥，只需备份一份随机种子的文件即可。

5、GO语言钱包

项目源码

https://github.com/miguelmota/go-ethereum-hdwallet

——HD-wallet go语言版本

如何学习源码

readme.md

- 找到github项目页面阅读 readme
- 查找关键词
- Install 安装
- Documenation 文档
- Getting Started 或者Quick Start 快速入门
- Testing

```
/*
@Time   :   2018-12-27 07:28
@Author :   suibingyue
@File   :   demo-hdwallet-test
*/

package main

import (
    "fmt"
    "go-ethereum-hdwallet"
    "log"
)
```

```
func main() {
    // 助记词
    mnemonic := "disorder timber among submit tell early claw certain
sadness embark neck salad"
    // 通过助记词创建了一个hd钱包
    wallet, err := hdwallet.NewFromMnemonic(mnemonic)
    if err != nil {
        log.Fatal(err)
    }

    var accounts []string
    var privateKeys []string

    count := 10
    for i := 0; i < count; i++ {
        temp := fmt.Sprintf("m/44'/60'/0'/0/%d", i)
        path := hdwallet.MustParseDerivationPath(temp)
        account, err := wallet.Derive(path, false)
        if err != nil {
            log.Fatal(err)
        }
        // 16进制string
        accounts = append(accounts, account.Address.Hex())
        // 16进制字符串
        privKey, _ := wallet.PrivateKeyHex(account)
        privateKeys = append(privateKeys, privKey)
    }

    fmt.Println("Available Accounts\n==================")
    for i := 0; i < count; i++ {
        fmt.Printf("(%d) %s\n", i, accounts[i])
    }
    fmt.Println("Private Keys\n==================")
    for i := 0; i < count; i++ {
        fmt.Printf("(%d) %s\n", i, privateKeys[i])
    }
}
```

下载源码后导入到IDE

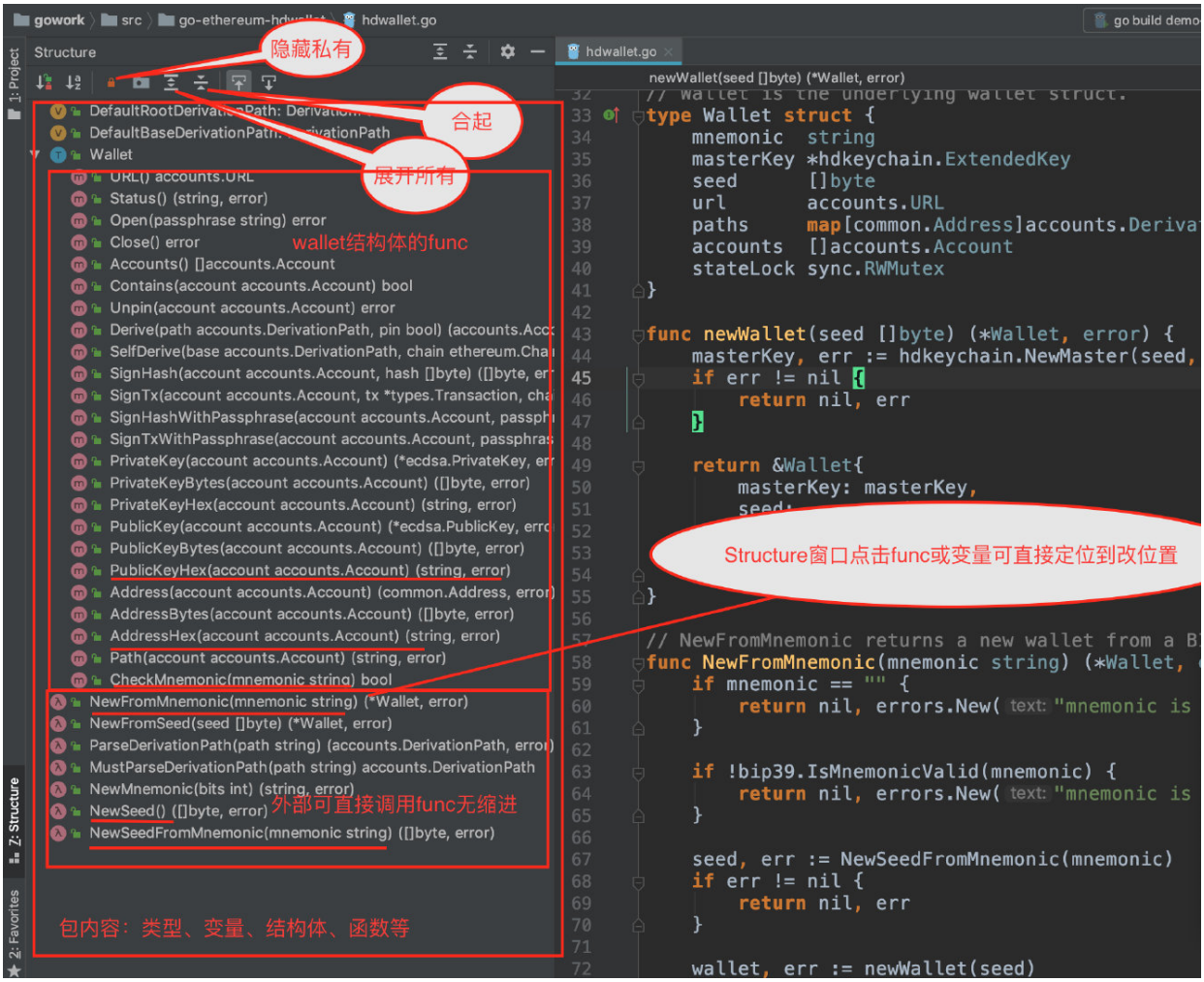
- run(如果项目有main入口)
- 查看包结构
- 查看test 文件(如有)
- 查看demo、example文件(如有)

- 自己写demo

查看结构

定位到源码位置，查看package结构，了解大体模块及作用

打开goland Structcture

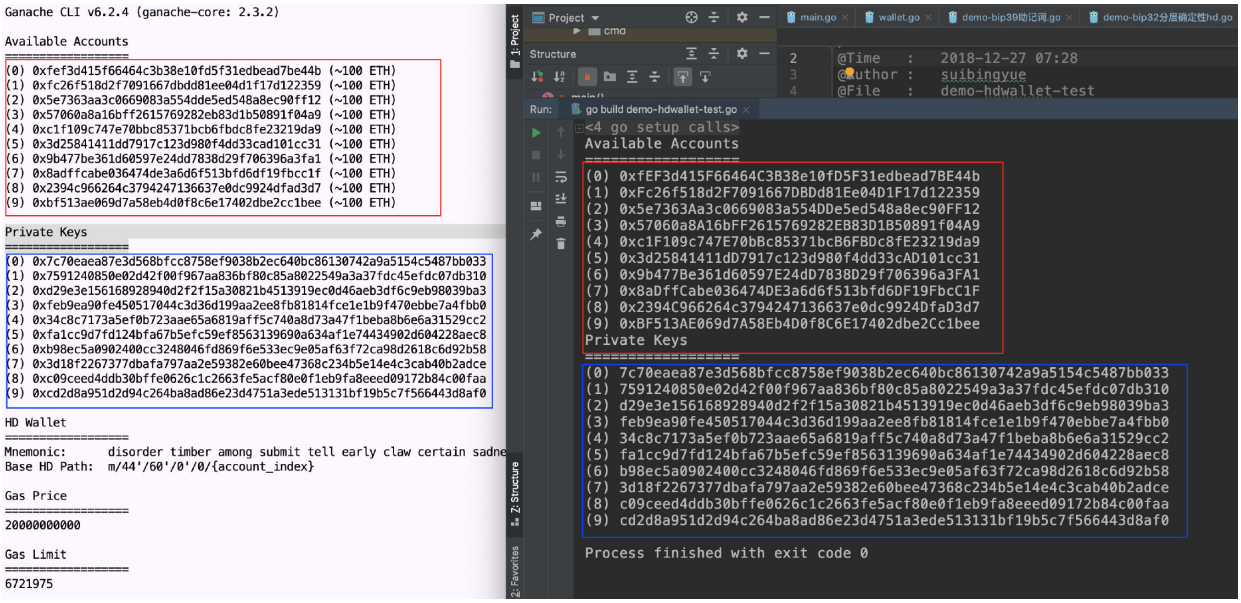


demo ETH助记词

模拟ganache或者metamask 助记词导入，显示账户地址和私钥信息

```
func main() {
    // 助记词使用巧克力
    mnemonic := "disorder timber among submit tell early claw certain
sadness embark neck salad"
    // 通过助记词创建了一个hd钱包
    wallet, err := hdwallet.NewFromMnemonic(mnemonic)
    if err != nil {
        log.Fatal(err)
    }

    var accounts []string
    var privateKeys []string

    count := 10
    for i := 0; i < count; i++ {
        // 路径模板
        temp := fmt.Sprintf("m/44'/60'/0'/0/%d", i)
        // 解析路径
        path := hdwallet.MustParseDerivationPath(temp)
        // 加载account
        account, err := wallet.Derive(path, false)
        if err != nil {
            log.Fatal(err)
        }
        // 16进制string
        accounts = append(accounts, account.Address.Hex())
        // 16进制字符串
        privKey, _ := wallet.PrivateKeyHex(account)
        privateKeys = append(privateKeys, privKey)
    }

    fmt.Println("Available Accounts\n==================")
    for i := 0; i < count; i++ {
        fmt.Printf("(%d) %s\n", i, accounts[i])
    }
    fmt.Println("Private Keys\n==================")
    for i := 0; i < count; i++ {
        fmt.Printf("(%d) %s\n", i, privateKeys[i])
    }

}
```

查看结果，与我们在控制台测试环境导入的信息一样



6、ethers.js

https://docs.ethers.io/ethers.js/html/

和以太坊系统进行交互的库：保存私钥、导入导出钱包、BIP39协议实现、合约交易、以太坊交互等

安装&导入

```
npm install --save ethers
import { ethers } from 'ethers';
```

四大模块

- Wallets and Signers 钱包签名
- Providers 以太环境
- Contracts 合约交互
- Utilities工具类

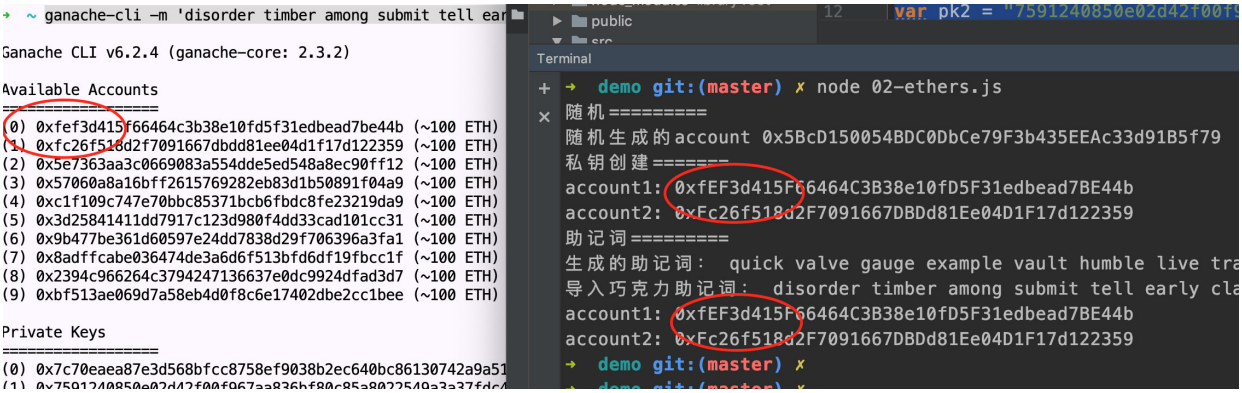
创建钱包方式

- 随机创建
- 通过私钥创建钱包
- *JSON*文件导入
- 通过助记词创建钱包

```
const ethers = require('ethers')

const accont1 = "m/44'/60'/0'/0/0"
const accont2 = "m/44'/60'/0'/0/1"

console.log("随机==========")
var wallet = ethers.Wallet.createRandom()
console.log("随机生成的account", wallet.address)
```

```
console.log("私钥创建=======")
var pk1 =
"7c70eaea87e3d568bfcc8758ef9038b2ec640bc86130742a9a5154c5487bb033"
var pk2 =
"7591240850e02d42f00f967aa836bf80c85a8022549a3a37fdc45efdc07db310"
var wallet1 = new ethers.Wallet(pk1)
var wallet2 = new ethers.Wallet(pk2)

console.log("account1:", wallet1.address)
console.log("account2:", wallet2.address)


console.log("助记词=========")
//随机gen助记词
var random = ethers.utils.randomBytes(16) //eth 128位
var mmic = ethers.utils.HDNode.entropyToMnemonic(random)
console.log("生成的助记词: ", mmic)

mmic = "disorder timber among submit tell early claw certain sadness
embark neck salad"
console.log("导入巧克力助记词: ", mmic)

//创建账号
wallet1 = ethers.Wallet.fromMnemonic(mmic, accont1)
wallet2 = ethers.Wallet.fromMnemonic(mmic, accont2)

console.log("account1:", wallet1.address)
console.log("account2:", wallet2.address)

// console.log(wallet1)
```

查看结果和命令行导入的一样



7、web钱包

1、创建工程

```
create-react-app webwallet
```

导入引用的包 bip、ethers、file-saver、pubsub、semantic

```
 "dependencies": {
    "bip39": "^2.5.0",
    "ethers": "^4.0.20",
    "file-saver": "^2.0.0",
    "json-format": "^1.0.1",
    "pubsub-js": "^1.7.0",
    "react": "^16.7.0",
    "react-dom": "^16.7.0",
    "react-scripts": "2.1.2",
    "semantic-ui-css": "^2.4.1",
    "semantic-ui-react": "^0.84.0"
  }
```

创建目录

```
➜  src git:(master) ✗ tree
├── App.css
├── App.js
├── service
│   └── service.js
├── serviceWorker.js
├── utils
└── view
    ├── login
    │   ├── login.js
    │   ├── tab_keystore.js
    │   ├── tab_mnemonic.js
    │   └── tab_private.js
    └── wallet
        ├── tab_account.js
        ├── tab_settings.js
        ├── tab_transaction.js
        └── wallet.js
```

2、首页

首页引入PubSub.js,事件发布订阅,如果未登录，显示导入页面，否则显示钱包页面

代码

```
import React, {Component} from 'react';
import PubSub from "pubsub-js";
import './App.css';
```

```
import LoginForm from './view/login/login'
import Wallet from './view/wallet/wallet'
import {Container} from "semantic-ui-react";

class App extends Component {
    state = {
        login: false,
        loading: false,
        loginEvent: '',
        wallets: []
    }
    // 页面加载，订阅消息，保存订阅id
    componentWillMount() {
        let loginEvent = PubSub.subscribe("onLoginSucc", this.onLoginSucc)
        this.setState({loginEvent})
    }

    onLoginSucc = (msg, data) => {
        console.log("登陆成功")
        console.log(data)
        this.setState({
            login: true,
            wallets: data
        })
    }
    // 页面卸载 取消消息订阅
    componentWillUnmount() {
      PubSub.unsubscribe(this.state.loginEvent)
    }

    render() {
        let {login} = this.state
        let content = login ? <Wallet wallets={this.state.wallets}/> :
<LoginForm/>
        return (
            <Container>
                {content}
            </Container>
        );
    }
}

export default App;
```

3、登陆页面

3.1 登录首页

登陆页面为3个tab,点击切换导入方式

登陆成功后发布消息

login.js

```
import React,{Component} from 'react'

import {Grid,Header, Image, Tab} from 'semantic-ui-react'
import PrivateLogin from "./tab_private"
import MmicLogin from "./tab_mnemonic"
import KeyStoreLogin from "./tab_keystore"

const panes = [
    {menuItem: '私钥', render: () => <Tab.Pane attached={false}>
<PrivateLogin/></Tab.Pane>},
    {menuItem: '助记词', render: () => <Tab.Pane attached={false}>
<MmicLogin/></Tab.Pane>},
    {menuItem: 'KeyStore', render: () => <Tab.Pane attached={false}>
<KeyStoreLogin/></Tab.Pane>},
]

export default class Login extends Component {
    render() {
        return (
            <Grid textAlign='center'  verticalAlign='middle'>
                <Grid.Column style={{maxWidth: 450, marginTop: 100}}>
                    <Header as='h2' color='teal' textAlign='center'>
                        <Image src='images/logo.png'/> EHT钱包
                    </Header>
                    <Tab menu={{text: true}} panes={panes} style=
{{maxWidth: 450}}/>
                </Grid.Column>
            </Grid>
        )
    }
}
```

3.2 私钥新建或导入

新建账号，随机一个私钥，

对用户输入的私钥进行校验 **service.checkPrivate(key)**

登陆成功后发布消息

```
import {Button, Form, Segment} from 'semantic-ui-react'
```

```
import React, {Component} from 'react'
import PubSub from 'pubsub-js'

let service = require('../../service/service')

export default class PrivateLogin extends Component {

    state = {
        privateKey: "",
    }
    // 随机创建
    handleCreateClick = () => {
        let privateKey = service.newRandomKey()
        this.setState({privateKey})
    }
    // 处理输入绑定
    handleChange = (e, {name, value}) => {
        this.setState({[name]: value})
    }
    // 私钥登陆
    onPrivateLoginClick = () => {
        let key = this.state.privateKey
        let err = service.checkPrivate(key)
        if (err !== "") {
            alert(err)
            return;
        }
        if (key.substring(0, 2).toLowerCase() !== '0x') {
            key = '0x' + key;
        }
        console.log("开始创建钱包", key)
        let wallets = service.newWalletFromPrivateKey(key)
        if (wallets) {
            PubSub.publish("onLoginSucc", wallets)
        } else {
            alert("导入出错")
        }
    }

    render() {
        return (
            <Form size='large'>
                <Segment>
                    <Form.Input
                        fluid icon='lock' iconPosition='left'
                        placeholder='private key'
                        name="privateKey"
                        value={this.state.privateKey}
                        onChange={this.handleChange}/>
```

```
                    <a href='#' onClick={this.handleCreateClick}>随机生
成</a>
                    <br/>
                    <br/>
                    <Button
                        color='teal' fluid size='large'
                        onClick={this.onPrivateLoginClick}>
                        私钥导入
                    </Button>
                </Segment>
            </Form>
        )
    }
}
```

```
// 私钥校验
function checkPrivate(key) {
    if (key === '') {
        return "不能为空"
    }
    if (key.length != 66 && key.length != 64) {
        return false, "秘钥长度必须为66或者64"
    }
    if (!key.match(/^(0x)?([0-9A-fa-f]{64})+$/)) {
        return "秘钥为16进制表示[0-9A-fa-f]"
    }
    return ""
}

// 随机私钥
function newRandomKey() {
    let randomByte = ethers.utils.randomBytes(32)
    let randomNumber = ethers.utils.bigNumberify(randomByte);
    return randomNumber.toHexString()
}

// 通过私钥创建钱包
function newWalletFromPrivateKey(privateKey) {
    let wallets = []
    let wallet = new ethers.Wallet(privateKey)
    wallets.push(wallet)
    return wallets
}
```

3.3 助记词新建或导入

助记词导入执行bip44协议，定义路径，默认**"m/44'/60'/0'/0/0"**，也可以进行多账号导入，修改index值即可

```
import {Button, Loader,Form, Grid, Header, Image, Message, Segment} from
'semantic-ui-react'
import PubSub from 'pubsub-js'
import React, {Component} from 'react'

let service = require('../../service/service')
// disorder timber among submit tell early claw certain sadness embark
neck salad

export default class MmicLogin extends Component {

    state = {
        privateKey: "",
        mmic: "",
        pwd: "",
        path: "m/44'/60'/0'/0/0",
    }
    // 处理输入文本绑定
    handleChange = (e, {name, value}) => {
        this.setState({[name]: value})
    }

    // 生成助记词
    handleGenMicc = () => {
        let mmic = service.genMmic()
        this.setState({mmic})
    }

    // 助记词导入
    onMMICClick = () => {
        let {mmic, path} = this.state
        let wallets = service.newWalletFromMmic(mmic, path)
        PubSub.publish("onLoginSucc", wallets)
    }

    render() {
        return (
            <Form size='large' onSubmit={this.onMMICClick}>
                <Segment stacked>
                    <Form.TextArea
                        placeholder='12 words'
                        name="mmic"
```

```
                        value={this.state.mmic}
                        onChange={this.handleChange}/>
                    <Form.Input
                        fluid
                        icon='user'
                        iconPosition='left'
                        type='path'
                        value={this.state.path}
                        onChange={this.handleChange}
                    />
                    <a onClick={this.handleGenMicc}>随机生成</a>
                    <br/>
                    <br/>
                    {/*<Form.Input*/}
                    {/*fluid*/}
                    {/*icon='lock'*/}
                    {/*iconPosition='left'*/}
                    {/*placeholder='Password'*/}
                    {/*type='password'*/}
                    {/*value={this.state.pwd}*/}
                    {/*onChange={this.handleChange}*/}
                    {/*/>*/}

                    <Form.Button color='teal' fluid size='large'>
                        助记词导入
                    </Form.Button>

                </Segment>
            </Form>
        )
    }
}
```

```
// 生成助记词
function genMmic() {
    let words =
ethers.utils.HDNode.entropyToMnemonic(ethers.utils.randomBytes(16));
    return words
}

// 通过助记词创建钱包
function newWalletFromMmic(mmic, path) {
    let wallets = []
    for (let i = 0; i < 10; i++) {
        path = PATH_PREFIX + i
        let wallet = ethers.Wallet.fromMnemonic(mmic, path)
```

```
            wallets.push(wallet)
            console.log(i, wallets)
        }
        return wallets
    }
```

3.4 keystore导入

wallet 需要连接provider 才可以使用

wallet balance为Object类型，金额需要手工转换(ethers.utils)

导出需要密码，

```
import {Button, Form, Grid, Header, Image, Loader, Message, Segment} from
'semantic-ui-react'
import PubSub from 'pubsub-js'
import _ethets2 from "ethers"
import React, {Component} from 'react'

let service = require('../../service/service')

export default class KeyStoreLogin extends Component {

    state = {
        keyStore: "",
        pwd: '',
        loading:false
    }

    handleChange = (e, {name, value}) => {
        this.setState({[name]: value})
    }

    // 处理导入
    handleKeyImport = () => {
        let {keyStore, pwd} = this.state
        if (keyStore==""){
            return
        }
        console.log(service.checkJsonWallet(keyStore))
        this.setState({loading:true})
        service.newWalletFromJson(keyStore, pwd).then(wallets => {
            PubSub.publish("onLoginSucc", wallets)
            this.setState({loading:false})
        }).catch(e => {
            console.log(e)
            alert("导入出错" + e)
```

```
            this.setState({loading:false})
        })
    }

    onFileChooseClick = ()=>{
    }

    render() {
        return (
            <Form size='large'>
                <Loader active={this.state.loading} inline />
                <Segment>
                    <Form.TextArea
                        placeholder='keystore为json格式'
                        name="keyStore"
                        value={this.state.keyStore}
                        onChange={this.handleChange}/>

                    <Form.Input
                        fluid
                        icon='lock'
                        iconPosition='left'
                        placeholder='Password'
                        type='password'
                        name = "pwd"
                        value={this.state.pwd}
                        onChange={this.handleChange}
                    />
                    <Button
                        color='teal' fluid size='large'
                            onClick={this.handleKeyImport}>
                        导入
                    </Button>
                </Segment>
            </Form>
        )
    }
}
```

```
// 从keystore导入钱包，需要密码
function newWalletFromJson(json, pwd) {

    return new Promise(async (resolve, reject) => {
        try {
            let wallets = []
            let wallet = await ethers.Wallet.fromEncryptedJson(json, pwd,
false)
            wallets.push(wallet)
```

```
            resolve(wallets)
        } catch (e) {
            reject(e)
        }
    })
}
```

通过keystorejson文件校验是否包含地址

```
// 校验地址
function checkJsonWallet(data) {
    return ethers.utils.getJsonWalletAddress(data)
}
```

4、钱包页面

获取钱包信息需要连接以太环境，请提前确认开启端口

钱包字段信息获取为异步，**注意**不可以直接调用显示

转账对地址及金额进行校验

```
import React, {Component} from 'react'
import {Grid, Form, Header, Loader, Button, Loading, Segment, Image} from
'semantic-ui-react'

let ethers = require('ethers')
let service = require('../../service/service')
let fileSaver = require('file-saver');

export default class Wallet extends Component {

    state = {
        wallets: [],// 支持多账户，默认第0个
        selectWallet: 0,
        provider: "http://127.0.0.1:8545", //环境
        walletInfo: [], // 钱包信息，获取为异步，单独存储下
        activeWallet: {}, // 当前活跃钱包
        txto: "", // 交易接收地址
        txvalue: "", // 转账交易金额
        pwd: "", // 导出keystore需要密码

        // UI状态表示
        txPositive: false, //
        loading: false,
        exportLoading: false,
```

```
    }

    constructor(props) {
        super(props)
        this.state.wallets = props.wallets
        this.state.selectWallet = props.wallets.length == 0 ? -1 : 0
    }

    // 更新钱包信息
    updateActiveWallet() {
        if (this.state.wallets.length == 0) {
            return null
        }
        let activeWallet = this.getActiveWallet()
        this.setState({activeWallet})
        this.loadActiveWalletInfo(activeWallet)
        return activeWallet
    }

    // 获取当前的钱包
    getActiveWallet() {
        let wallet = this.state.wallets[this.state.selectWallet]
        console.log("wallet", wallet)
        // 激活钱包需要连接provider
        return service.connectWallet(wallet, this.state.provider)
    }

    // 加载钱包信息
    async loadActiveWalletInfo(wallet) {
        let address = await wallet.getAddress()
        let balance = await wallet.getBalance()
        // 获取交易次数
        let tx = await wallet.getTransactionCount()
        this.setState({
            walletInfo: [address, balance, tx]
        })
    }

    // 发送交易
    onSendClick = () => {
        let {txto, txvalue, activeWallet} = this.state
        // balance 为Object类型

        console.log("balance", activeWallet)
        // 地址校验
        let address = service.checkAddress(txto)
        if (address == "") {
            alert("地址不正确")
```

```
            return
        }
        console.log(txvalue, isNaN(txvalue))
        if (isNaN(txvalue)) {
            alert("转账金额不合法")
            return
        }
        // 以太币转换，发送wei单位
        txvalue = ethers.utils.parseEther(txvalue);
        console.log("txvalue", txvalue)

        // 设置加载loading，成功或者识别后取消loading
        this.setState({loading: true})

        service.sendTransaction(activeWallet, txto, txvalue)
            .then(tx => {
                console.log(tx)
                alert("交易成功")
                this.updateActiveWallet()
                this.setState({loading: false, txto: "", txvalue: ""})
            })
            .catch(e => {
                this.setState({loading: false})
                console.log(e);
                alert(e);
            })
    }

    // 查看私钥
    onExportPrivate = () => {
        alert(this.getActiveWallet().privateKey)
    }
    // 导出keystore
    onExportClick = () => {
        let pwd = this.state.pwd;
        if (pwd.length < 6) {
            alert("密码长度不能小于6")
            return
        }
        this.setState({exportLoading: true})
        // 通过密码加密
        this.getActiveWallet().encrypt(pwd, false).then(json => {
            let blob = new Blob([json], {type: "text/plain;charset=utf-
8"})
            fileSaver.saveAs(blob, "keystore.json")
            this.setState({exportLoading: false})
        });
    }
```

```
    // 页面加载完毕，更新钱包信息
    componentDidMount() {
        this.updateActiveWallet()
    }

    handleChange = (e, {name, value}) => {
        this.setState({[name]: value})
    }

    render() {
        // 金额显示需要手工转换
        let wallet = this.state.walletInfo
        if (wallet.length == 0) {
            return <Loader active inline/>
        }
        let balance = wallet[1]
        let balanceShow = ethers.utils.formatEther(balance) + "(" +
balance.toString() + ")"
        return (
            <Grid textAlign='center' verticalAlign='middle'>
                <Grid.Column style={{maxWidth: 650, marginTop: 10}}>
                    <Header as='h2' color='teal' textAlign='center'>
                        <Image src='images/logo.png'/> EHT钱包
                    </Header>
                    <Segment stacked textAlign='left'>
                        <Header as='h1'>Account</Header>
                        <Form.Input
                            style={{width: "100%"}}
                            action={{
                                color: 'teal',
                                labelPosition: 'left',
                                icon: 'address card',
                                content: '地址'
                            }}
                            actionPosition='left'
                            value={wallet[0]}
                        />
                        <br/>
                        <Form.Input
                            style={{width: "100%"}}
                            action={{
                                color: 'teal',
                                labelPosition: 'left',
                                icon: 'ethereum',
                                content: '余额'
                            }}
                            actionPosition='left'
                            value={balanceShow}
                        />
```

```
                        <br/>
                        <Form.Input
                            actionPosition='left'
                            action={{
                                color: 'teal',
                                labelPosition: 'left',
                                icon: 'numbered list',
                                content: '交易'
                            }}
                            style={{width: "100%"}}
                            value={wallet[2]}
                        />
                    </Segment>
                    <Segment stacked textAlign='left'>
                        <Header as='h1'>转账|提现</Header>
                        <Form.Input
                            style={{width: "100%"}}
                            action={{
                                color: 'teal',
                                labelPosition: 'left',
                                icon: 'address card',
                                content: '地址'
                            }}
                            actionPosition='left'
                            defaultValue='52.03'
                            type='text' name='txto' required value=
{this.state.txto}
                            placeholder='对方地址' onChange=
{this.handleChange}/>
                        <br/>
                        <Form.Input
                            style={{width: "100%"}}
                            action={{
                                color: 'teal',
                                labelPosition: 'left',
                                icon: 'ethereum',
                                content: '金额'
                            }}
                            actionPosition='left'
                            defaultValue='1.00'
                            type='text' name='txvalue' required value=
{this.state.txvalue}
                            placeholder='以太' onChange=
{this.handleChange}/>
                        <br/>
                        <Button
                            color='twitter'
                            style={{width: "100%"}}
                            size='large'
```

```
                            loading={this.state.loading}
                            onClick={this.onSendClick}>
                            确认
                        </Button>
                    </Segment>
                    <Segment stacked textAlign='left'>
                        <Header as='h1'>设置</Header>
                        <Form.Input
                            style={{width: "100%"}}
                            action={{
                                color: 'teal',
                                labelPosition: 'left',
                                icon: 'lock',
                                content: '密码'
                            }}
                            actionPosition='left'
                            defaultValue='1.00'
                            type='pwd' name='pwd' required value=
{this.state.pwd}
                            placeholder='密码'
                            onChange={this.handleChange}/>
                        <br/>
                        <Button
                            color='twitter'
                            style={{width: "48%"}}
                            onClick={this.onExportPrivate}>
                            查看私钥
                        </Button>
                        <Button
                            color='twitter'
                            style={{width: "48%"}}
                            onClick={this.onExportClick}
                            loading={this.state.exportLoading}>
                            导出keystore
                        </Button>
                    </Segment>
                </Grid.Column>
            </Grid>
        )
    }
}
```

5、service.js

```
import {BlockTag, Provider, TransactionRequest, TransactionResponse} from
"ethers/providers";
```

```
import {Arrayish, BigNumber, ProgressCallback} from "ethers/utils";

let ethers = require('ethers')

// 默认路径
const PATH_PREFIX = "m/44'/60'/0'/0/"

// 私钥校验
function checkPrivate(key) {
    if (key === '') {
        return "不能为空"
    }
    if (key.length != 66 && key.length != 64) {
        return false, "秘钥长度必须为66或者64"
    }
    if (!key.match(/^(0x)?([0-9A-fa-f]{64})+$/)) {
        return "秘钥为16进制表示[0-9A-fa-f]"
    }
    return ""
}

// 校验地址
function checkJsonWallet(data) {
    return ethers.utils.getJsonWalletAddress(data)
}

// 连接provider
function connectWallet(wallet, providerurl) {
    let provider = new ethers.providers.JsonRpcProvider(providerurl);
    return wallet.connect(provider);
}

// 发送交易
function sendTransaction(wallet,to,value) {
    return wallet.sendTransaction({
        to: to,
        value: value,
    })
}
```

6 、多账户钱包

TODO，页面中为wallet[] 数组，想实现账号的，增加 账号选择模块即可

8 环境问题

ganache provider是可以获取钱包信息，但转账会提示余额不足

大家使用 ganache-cli 开启eth测试环境即可

【可选】为了便于查看转账及余额信息，大家可以开启geth命令行，开启时添加端口

- geth attach provider 开启geth命令行
- eth.accounts 获取当前账号信息
- eth.getBalance(eth.accounts[8]) 获取某一个账号余额

```
➜  00-wallet git:(master) ✗ geth attach http://localhost:8545
Welcome to the Geth JavaScript console!
> eth.accounts
["0xfef3d415f66464c3b38e10fd5f31edbead7be44b",
"0xfc26f518d2f7091667dbdd81ee04d1f17d122359",
"0x5e7363aa3c0669083a554dde5ed548a8ec90ff12",
"0x57060a8a16bff2615769282eb83d1b50891f04a9",
"0xc1f109c747e70bbc85371bcb6fbdc8fe23219da9",
"0x3d25841411dd7917c123d980f4dd33cad101cc31",
"0x9b477be361d60597e24dd7838d29f706396a3fa1",
"0x8adffcabe036474de3a6d6f513bfd6df19fbcc1f",
"0x2394c966264c3794247136637e0dc9924dfad3d7",
"0xbf513ae069d7a58eb4d0f8c6e17402dbe2cc1bee"]
> eth.getBalance(eth.accounts[0])
100000000000000000000
> eth.getBalance(eth.accounts[8])
120000000000000000000
```

9 源码地址

无业游民-唐朝

课件及源码 https://github.com/bigsui/eth-webwallet

分享是没得，但别忘了点👍赞 start，fork，后期会更新

邮箱：suibingyue@gmail.com

微信: SuiaBing

问题咨询：71小时支持(23：00以后)

10、相关链接

官方GitHub网址：https://github.com/ethereum Geth的Github地址：https://github.com/ethereum/go-ethereum 官方不同系统安装Geth指南地址：https://github.com/ethereum/go-ethereum/wiki/Building-Ethereum

官方Geth命令行参数说明：https://github.com/ethereum/go-ethereum/wiki/Command-Line-Options