NetworkStatusBlockchain System

xyangxi5@gmail.com

1 Account

1.1 Generate key

输入一个 seed(`[]byte`), 返回一个 ed25519 的私钥.

$$\text{private key} = \text{NewKeyFromSeed}(\text{Sha256}(\mathsf{header}, \mathsf{seed}))$$

注,ed25519 是 eddsa 的 curve25519 实现:

EdDSA

ed25519

Things that use ed25519

1.2 Signature

$$\text{signature} = \text{Sign}(\text{private key}, \text{Sha512}(\mathsf{header}, \mathsf{message}))$$

1.3 Check

可以使用 Hash 过的消息，也可以使用明文.

1.4 Store

暂时以明文存放

2 Transaction

传输采用字节流:

$$\text{bytesJson} = \text{Type}|\text{Content}$$

例如 Type 为 Contract Invoke 时:

$$\text{Content} = \text{FunctionName}|\text{Header}$$

2.1 Check

暂时把所有逻辑放到 Deliver 中

2.2 ContractFunctions/CreateContracts

2.2.1 prepare

解读 Header.

```
type TransactionHeader struct {
    From            []byte        `json:"from"`
    ContractAddress []byte        `json:"to"`
    JsonParas       []byte        `json:"data"`
    Value           *math.Uint256 `json:"value"`
    Nonce           *math.Uint256 `json:"nonce"`
    Signature       []byte        `json:"signature"`
}
```

得到 Contract Environment

```
type ContractEnvironment struct {
    Storage         *localstorage.LocalStorage

    from            []byte
    fromInfo        *AccountInfo
    contractAddress []byte
    toInfo          *AccountInfo

    Data            []byte
    value           *math.Uint256
}
```

其中 Storage 的实现基于 MerkMap. fromInfo 和 toInfo 来自 StateRoot. Data 为函数输入.

2.2.2 contract functions invoke

contract 的语言为 go, 只允许利用 Storage 存取数据库. 如何禁止 golang 具有危险性的行为是个问题.

可能的解决办法:

1 只允许有一个 package(即不允许带有目录结构)

2 只允许使用指定库 (绝对禁止 IO 库等)

传参的办法, 以下为唯一的函数格式:

```
type contractFunc func([]byte) (*ContractCallBackInfo)
```

任何字节流协议都可以用作解析格式. 其中 ContractCallBackInfo 用于回调, 做那些 Contract 没有权限的善后处理.

2.3 contract functions types

只有 Registered 过的 Function 才能调用，其他函数被视为内部函数.

3 LocalStorage

3.1 MerkMap

与 Solidity 的 mapping 一样,MerkMap 接受一个 offset, 将目标数据存储在 Keccak256(slot, offset) 中.

3.2 ArrangeSlot

ArrangeSlot 使得 MerkMap 成为能够完全管理一个独立存储空间的数据结构.

以 stateTrie 对应的 stateMap 为例,stateMap 是所有 Map 中唯一 slot 为空的 map,txMap 是 stateMap 进行 slot 偏移 (`txMap = stateMap.ArrangeSlot([]byte("tx:"))`),accMap 是 stateMap 进行另一个 slot 偏移 (`accMap = stateMap.ArrangeSlot([]byte("acc:"))`). 本质 stateMap,txMap 和 accMap 在同一颗 Trie(stateTrie) 上.

3.3 LocalStorage

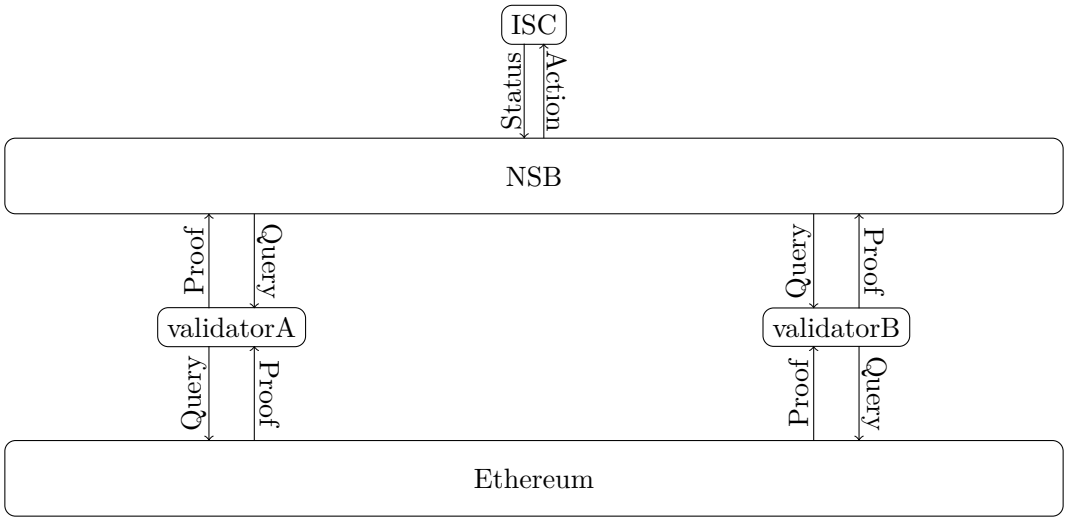
LocalStorage 封装了这种特性, 对于合约存储空间. 任意 Map 都以 Sha256(accountaddress, mapname) 为 slot. 并且任意非 Map 变量，都以空 slot(Sha256(accountaddress)) 为 slot. 下面是对 LocalStorage 的测试:

冲突测试

可持久化测试

4 Cross-Blockchain MerkleProof

4.1 单链模型



Tendermint 有 SetOption, 用于设定非 Consensus 的链参数.

比如 `set_ethereum_rpc = http://localhost:8545`, 要求每个 validator 只使用一个 rpc, 防止一个不可信 rpc 被多个 validator 相信.

这种 NSB 的实现内嵌 Ethereum 的 MerkleProof 验证逻辑.

但是这样所有 validator 都必须同时拥有多条链的账户. 如果一个用户不拥有某条链的账户, 那么它的投票不应该算在最终的结果中. 但事实上 Tendermint 的投票机制的要求所有节点参与的, 这是共识的达成. 如何知道 validator 确实拥有多条链? 现在不知道如何解决.

如何知道 validator 不是来自一个用户?Ethereum 和 Bitcoin 目前来看都有 PoW 作支撑, 但是 Tendermint 的投票机制 (每个 PublicKey 对应一个 Power) 给人带来疑问.

5 todo

1 application

a nsb action 这周 → 下周

b nsb merkle proof 这周 → 下周

c isc insurance claim 下周 → 下周

d isc settle contract 下周 → 下周

2 contract

a create 新增需求 → doing

b invoke 新增需求 → doing

c storage 新增需求 → OK

3 MPT 下周 → OK

4 HyperService wrapping 下周 → 下周.