Hybrid BlockChain

Budapest, 2018.

Abstract

Can the block chain be scaled to support thousands of transactions per second, like the world's major payment networks?

Ethereum is a public block chain platform that lets anyone build and use decentralized applications. In its current state, Ethereum doesn't scale very well because the history of every transaction ever made needs to be downloaded by the full node i.e synching of blocks. Computational power must also be used to add new blocks.

With the ever increasing use of the platform, you would need a-lot of resources i.e super computers with infinite storage and infinite speed thus becoming too costly for the average user or to reduce the number of transactions that can be processed by the platform which would also affect the performance of the system for the users.

We propose a solution based on layer3 of the Internet's OSI model. The internet is composed of various networks with different hardware protocols and services interconnected to exchange information currently at acceptable speeds.

To simulate a block chain net, we use the proof of authority rinkeby test net. A contract checks if the proof of authority block header of the test net is accepted as a valid header because the signature of the transaction is verified as the address of the miner themselves. We have one validator on this test net, my mining address.

Blocks on this proof of authority net are included without lots of computational power because consensus is based on validation from more than one miner's address picked on a random basis. The term miner is inherited from previous proof of work block chains.

This allows us to explore the possible interoperability between different block chains and the system can still process as many transactions.

The advantages of this design include being inexpensive, high throughput, low latency and anonymity.

Contents

Chapter 1

Introduction

A block chain is defined as a distributed database/ledger where every network node executes and records the same transactions grouped into blocks. Only one block can be added at a time, and every block contains a nonce that verifies that it follows in sequence from the previous block.

This technology was originally used as the underlying peer-to-peer distributed timestamp server for Bitcoin. It's based on cryptographic proof allowing any two willing parties to transact directly with each other without the need for a trusted third party. A transaction is a chain of digital signatures with each owner making transfers to the next owner by digitally signing a hash of the previous transaction and the public key of the next owner. A hash of a block of transactions to be timestamped is published in the peer to peer network. This proves that the data must have existed at that time to get into the hash. Each timestamp will include the previous timestamp in it's hash, forming a chain that reinforces the blocks before it. Implementing this system as a peer to peer network requires a Proof of Work consensus. Proof of Work is a consensus system that relies on computational proof for the chronological order of transactions. CPU/GPU time and electricity are used when work is done with CPU/GPU cycles to scan for a value that when hashed, like with SHA-256, gives a value that begins with a number of zero bits. The block cannot be changed without having to redo this. Miners race to solve this cryptographic problem and are rewarded with BitCoin to validate transactions and create new blocks.[1]

In Ethereum, each and every node of the network runs the Ethereum Virtual Machine and executes the same Contract Accounts. The virtual machine on every node maintains consensus as each and every node connected to the network executes the same instructions. This is used to implement a trustless smart contract platform and allows users to create their own decentralised applications. The sender of a transaction pays with Ether for the program activated as well as computation and memory storage. The transaction fees are

collected by the nodes that validate the network. These nodes receive, propagate, verify and execute transactions by grouping them into blocks. These miners race to solve a proof of Work which is a memory-hard as well as CPU computational problem.[9]

Proof of stake is an alternate consensus system that is based on proof of ownership of the currency i.e someone holding 5 percent of Bitcoin can mine 5 percent of the blocks.[2]

Proof of Authority is an alternate consensus system where instead of miners racing to find a solution to a difficult problem, authorized signers can at any time and own discretion create new blocks.[1.3]

Transitions between consensus systems or changes on the blockchain can be achieved by hard forks or soft forks. A hard fork is permanent, when non-upgraded nodes can't validate blocks created by upgraded nodes on a chain while a soft fork is temporary, non-upgraded nodes don't follow the new consensus rules.

There are three types of blockchains, Public, a "fully public, uncontrolled network and state machines secured by crypto economics", with examples described earlier. Private blockchains however have "write permissions which are centralized to one organization, read permissions may be public or restricted to an arbitrary extent". The third type, Consortium Block Chains have a "consensus process controlled by a pre-selected set of nodes"[3]

While these implementations in block chains allow for integrity and availability on an insecure web, existing block chains have a number of challenges with increasing widespread use and adoption.

Chapter 2

Literature Review

2.1 Scaling Ethereum

Scaling addresses the ability of the system to maintain its quality of service as the overall system load increases. Ethereum aims to allow the processing of a high volume of transactions, maybe 10,000+ transactions per second without requiring a full node running Ethereum to become a super computer or to store a terabyte of data of the blockchain.

2.1.1 Sharding

The design considerations for scaling with sharding aim to ensure that the workload of state storage, transaction processing, downloading and re-broadcasting is spread out across nodes.

Scalability is being able to process O(n) transactions greater than O(c) computational resources. i.e O(n) > O(c) with each participant having O(c) resources and security against attackers up to O(n). O(n) is a linear scale, every time you double n, the amount of work is doubled so that for every element, you are doing a constant number of operations.

The validation effort required per full node simply grows in a linear scale O(n), but with each node participating on the network computing every smart contract transaction, the combined validation effort of all nodes grows by O(n^2) with decentralization being held constant.O(n^2)is a quadratic scale, Every time n doubles, the operation takes four times as long.

Ethereum blockchain shows the state of accounts.There are two types of accounts, Externally Owned Accounts (EOAs) which are controlled by a user's private keys and contract accounts controlled by their contract code that performs instructions based on the Externally Owned Accounts.

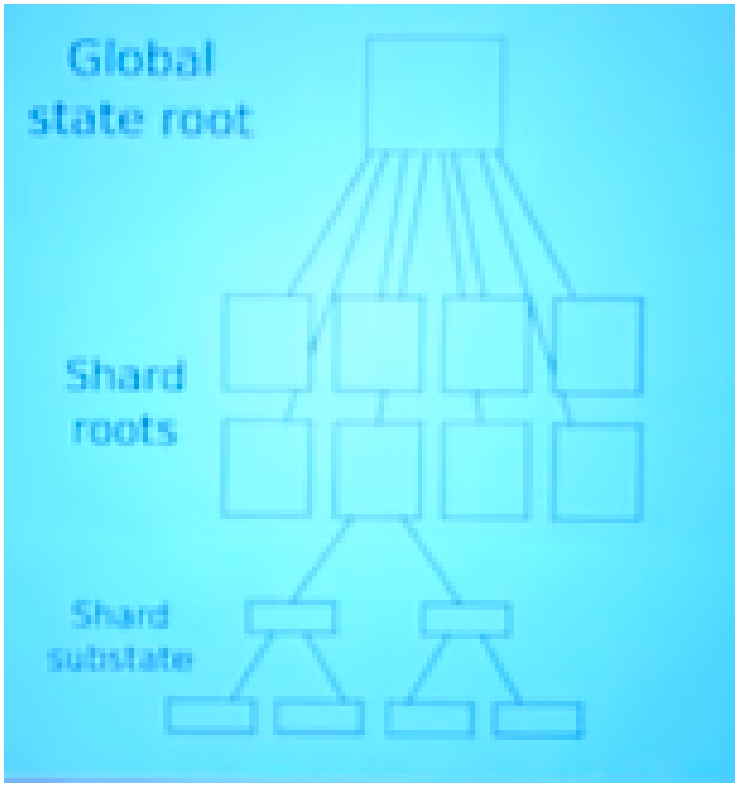

Figure 2.1: Sharding

[1.1]

Sharding is the splitting of the space of possible accounts like contracts into sub spaces. [4] This could be based on the numbering of their addresses.

If the value of the cryptocurrency is O(k log(k)) with k users. A good model to use would be n = O(klog(k)), basing everything off of n and c. O(n log n) operations run in log linear time, O(log n) is performed on each item in your input. c refers to computational resources like computation, bandwidth and storage, n refers to the size of the ecosystem i.e the transaction load, state size and cryptocurrency of the market.

In this solution, nodes might be arranged according to addresses and hold a subset of the state and subsequently of the blockchain thus sharing the load instead of everyone doing the same work. Please see Figure 2.1. Each shard has it's own history and the effects of that shard are limited to it.

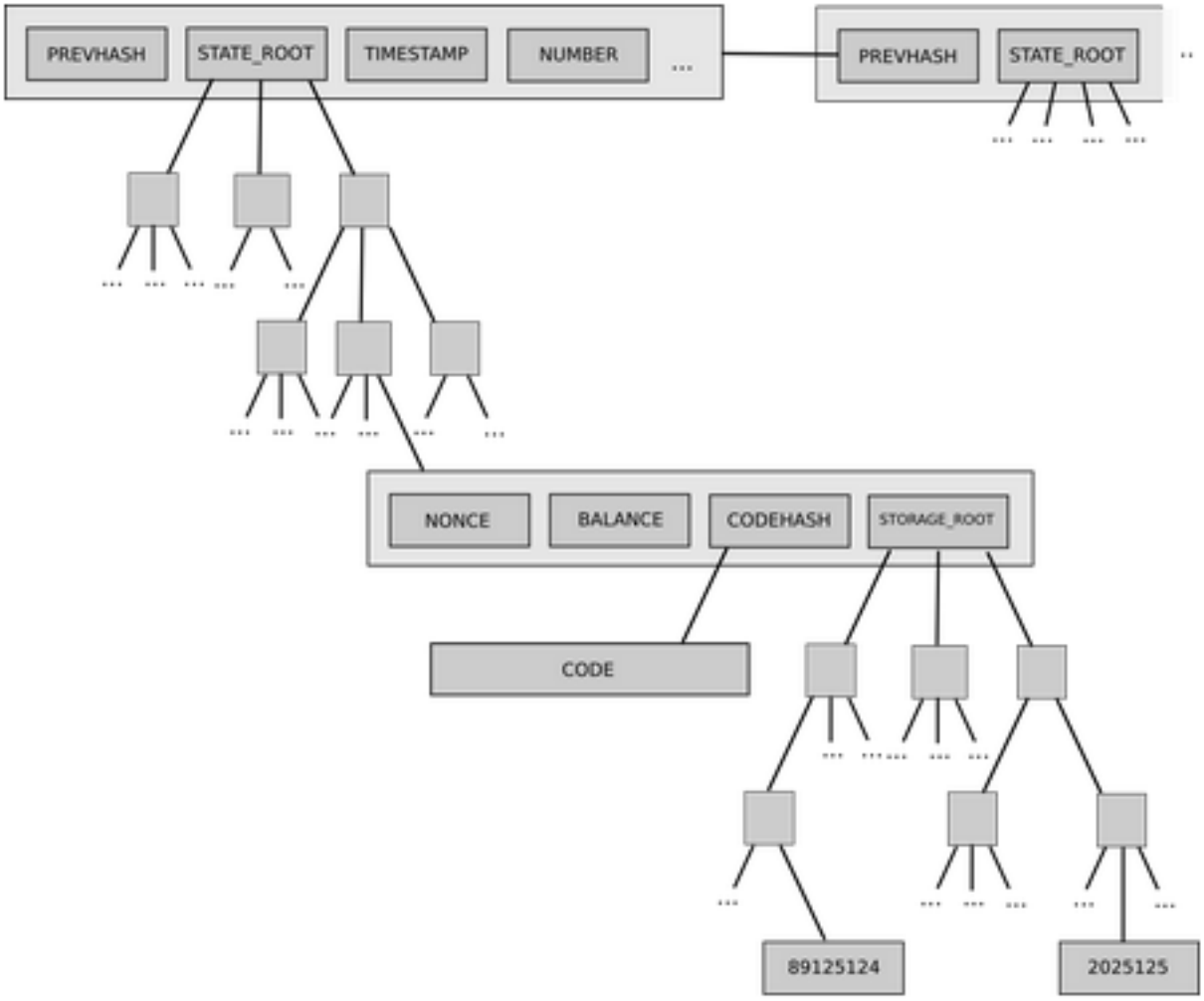

Figure 2.2: EthereumStateTree

[6]

2.1.2 Definitions Ethereum Block Header

2.1.1. Definiton. *The block in Ethereum is the collection of relevant pieces of information (known as the block header), together with information corresponding to the comprised transactions, and a set of other blockheaders, that are known to have a parent equal to the present block's parent's parent (such blocks are known as ommers).*

The block header contains several pieces of information:

2.1.2. Definiton. *parentHash: The Keccak 256-bit hash of the parent block's header, in its entirety.*

2.1.3. Definiton. *ommersHash: The Keccak 256-bit hash of the ommers list portion of this block*

2.1.4. Definiton. *beneficiary: The 160-bit address to which all fees collected from the successful mining of this block are to be transferred*

2.1.5. Definiton. *stateRoot: The Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied*

.

2.1.6. Definiton. *transactionsRoot: The Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list portion of the block*

2.1.7. Definiton. *receiptsRoot: The Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list portion of the block.*

2.1.8. Definiton. *logsBloom: The Bloom filter composed from indexable information (logger address and log topics) contained in each log entry from the receipt of each transaction in the transactions list .*

2.1.9. Definiton. *difficulty: A scalar value corresponding to the difficulty level of this block. This can be calculated from the previous block's difficulty level and the timestamp*

2.1.10. Definiton. *number: A scalar value equal to the number of ancestor blocks. The genesis block has a number of zero .*

2.1.11. Definiton. *gasLimit: A scalar value equal to the current limit of gas expenditure per block .*

2.1.12. Definiton. *gasUsed: A scalar value equal to the total gas used in transactions in this block*

2.1.13. Definiton. *timestamp: A scalar value equal to the reasonable output of Unix's time() at this block's inception. .*

2.1.14. Definiton. *extraData: An arbitrary byte array containing data relevant to this block. This must be 32 bytes or fewer. .*

2.1.15. Definiton. *mixHash: A 256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block*

2.1.16. Definiton. *nonce: A 64-bit hash which proves combined with the mix-hash that a sufficient amount of computation has been carried out on this block*

BlockHeaderDefinitions[1.4]

2.1.3 Proof of Authority

In the Clique proof of authority protocol proposal, authorized signers can at any time create new blocks instead of mining as in the proof of work consensus algorithm.

A Proof of Authority scheme is based on the idea that blocks may only be minted by trusted signers. As such, every block or header that a client sees can be matched against the list of trusted signers.

The list of authorized signers can change in time so the protocol of maintaining the list of authorized signers must be fully contained in the block headers. [1.3]

Chapter 3

Findings / Results / Data Analysis

3.1 Hybrid BlockChain

To simulate a block chain net, we use the proof of authority rinkeby test net from the open-source go-lang implementation of ethereum. We have a script taking the header generated on the test net and checking if the header is indeed signed by the validator on the test net, my mining address.

To verify the signature, we take the following components of the blockheader as defined in Ethereum's Yellow Paper also described as per definition in the chapter before, parenthash, sha3uncles, coinbase address, stateRoot, transactions root, receiptsRoot, logsBloom, difficulty, number, gasLimit, gasUSed, timestamp, extraData, mixHash and nonce.

We then store this information using RLP encoding. Recursive Length Prefix is a data structure used to store objects on the Ethereum blockchain. We then write a decode function to double check our encode function. All this information is then hashed to be used in solidity's ec recover function. Solidity's ec recover function is based on the Elliptic Curve Digital Signature Algorithm for verification.

Solidity contract checks if the proof of authority block header of the test net is accepted as a valid header because the signature of the transaction is verified as the address of the miner themselves. We have one validator on this test net, my mining address. Once this address is verified as the address on the test net, the net is verified and transactions can be submitted.

3.2 Implementation

Experiment carried out for the Hybrid BlockChain - Go-Ethereum- golang implementation of ethereum - A genesis block for a proof of authority test chain with network id 1988

simulating a private block chain. - Mist wallet with geth - A solidity contract that verifies the signature of a proof of authority block header of our private test chain. - An online Remix solidity IDE to write solidity contracts - An online Web3 deploy to remove line breaks - Bash scripts to initiate proof of work or proof of authority block chains - JSON api to interact with test net and deploy the solidity contract on geth

As part of the implementation, I create a genesis block for the private test chain using the Ethereum protocol. I use Mist 0.9.0 Browser that has geth included as part of the package. All these tests are carried out on MacOSX.

[1.2]

3.2.1 Genesis ProofOfWork Block

```
{

"config": {
"chainId": 1987,
"homesteadBlock": 0,
"eip155Block": 0,
"eip158Block": 0
},
    "nonce": "0x0000000000000042",     "timestamp": "0x0",
    "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "extraData": "0x00",     "gasLimit": "0x8000000",     "difficulty": "0x400",
    "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "coinbase": "0x3333333333333333333333333333333333333333",
    "alloc":
    {
     "0xa6c24f36c5463a1850aab17547ea83c99a56e0a6": { "balance": "2000000000000000000" }
     }




}
```

The genesis block is initialized using

/Users/User1/go-ethereum/build/bin/geth –identity "Princess" –rpc –rpccorsdomain

Figure 3.1: GenesisBlock

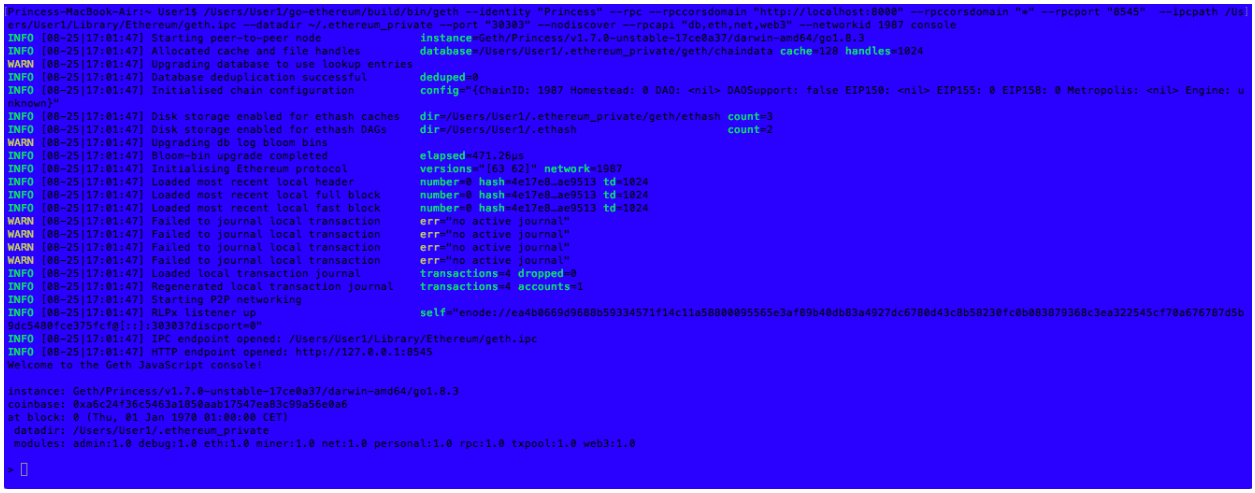

Figure 3.2: GethConsole

"http://localhost:8000" –rpccorsdomain "*" –rpcport "8545" –ipcpath /Users/User1/Library/Ethereum/geth.ipc –datadir /.ethereumprivate –port "30303" –nodiscover –rpcapi "db,eth,net,web3" –networkid 1987 init /Users/User1/Library/Ethereum/CustomGenesis.json

To interact with geth through the console;

/Users/User1/go-ethereum/build/bin/geth –identity "Princess" –rpc –rpccorsdomain "http://localhost:8000" –rpccorsdomain "*" –rpcport "8545" –ipcpath /Users/User1/Library/Ethereum/geth.ipc –datadir /.ethereumprivate –port "30303" –nodiscover –rpcapi "db,eth,net,web3" –networkid 1987 console

Figure 3.3: Accounts

To initialize a new account on the testnest, that will be used as your etherbase (the address that receives mining rewards).

personal.newAccount("password")

To add a new wallet to the Custom Genesis file, adjust the alloc settings in the genesis.json file and reinitate genesis file

Commands run on remote peer node

Set it as the etherbase

miner.setEtherbase(personal.listAccounts[0])

Return account addresses you possess.

eth.accounts

Set primary account

primary = eth.accounts[0]

Check balance

balance = web3.fromWei(eth.getBalance(primary), "ether")

Select

sudo geth –datadir path/to/custom/data/folder –networkid 1987 –bootnodes enode://98aa7ae99c72280ff329eb02c6ed2c6aa49a70c14d1ca41410854cdd46470e28c8ce99b0efb862c55e2c75

mine

–minerthreads=1

Running another private test chain with proof of authority consensus. Using puppeth, tools installed with geth, i generate and initiate a proof of work json file as below

```
Princess-MacBook-Air:~ User1$ /Users/User1/go-ethereum/build/bin/puppeth
+-----------------------------------------------------------+
| Welcome to puppeth, your Ethereum private network manager |
|                                                           |
| This tool lets you create a new Ethereum network down to  |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
|                                                           |
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
+-----------------------------------------------------------+

Please specify a network name to administer (no spaces, please)
> testnet
Sweet, you can set this via --network=testnet next time!

INFO [01-28|17:59:37] Administering Ethereum network           name=testnet
WARN [01-28|17:59:37] No previous configurations found         path=/Users/User1/.puppet

What would you like to do? (default = stats)
 1. Show network stats
 2. Configure new genesis
 3. Track new remote server
 4. Deploy network components
> 2

Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 2

How many seconds should blocks take? (default = 15)
```

```
>

Which accounts are allowed to seal? (mandatory at least one)
> 0x
> 0xbde4f8db67c0024e627929c796ad325156ec9c3b
> 0x

Which accounts should be pre-funded? (advisable at least one)
> 0xbde4f8db67c0024e627929c796ad325156ec9c3b
> 0x


Specify your chain/network ID if you want an explicit one (default = random)
> 1988

Anything fun to embed into the genesis block? (max 32 bytes)
>

What would you like to do? (default = stats)
 1. Show network stats
 2. Save existing genesis
 3. Track new remote server
 4. Deploy network components
> 2

Which file to save the genesis into? (default = testnet.json)
>
INFO [01-28|18:06:05] Exported existing genesis block
```

3.2.2 Genesis ProofOfAuthority Block


```
{
  "config": {
    "chainId": 1988,
    "homesteadBlock": 1,
    "eip150Block": 2,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 3,
    "eip158Block": 3,
    "clique": {
      "period": 15,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x59b116d0",
  "extraData": "0x64616d6e626c6f636b636861696e0000000000000000000000000000000000000000a6c24f
  "gasLimit": "0x47b760",
  "difficulty": "0x1",
  "number": "0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {
    "0000000000000000000000000000000000000000": {
      "balance": "0x1"
    },
    "0000000000000000000000000000000000000001": {
      "balance": "0x1"
    },
    "0000000000000000000000000000000000000002": {
      "balance": "0x1"
    },
    "0000000000000000000000000000000000000003": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000004": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000005": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000006": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000007": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000008": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000009": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000000a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000000b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000000c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000000d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000000e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000000f": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000010": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000011": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000012": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000013": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000014": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000015": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000016": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000017": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000018": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000019": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000001a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000001b": {
```

```
    "balance": "0x1"
  },
  "000000000000000000000000000000000000001c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000001d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000001e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000001f": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000020": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000021": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000022": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000023": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000024": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000025": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000026": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000027": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000028": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000029": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000002a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000002b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000002c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000002d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000002e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000002f": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000030": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000031": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000032": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000033": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000034": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000035": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000036": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000037": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000038": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000039": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000003a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000003b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000003c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000003d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000003e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000003f": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000040": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000041": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000042": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000043": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000044": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000045": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000046": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000047": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000048": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000049": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000004a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000004b": {
```

```
    "balance": "0x1"
  },
  "000000000000000000000000000000000000004c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000004d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000004e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000004f": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000050": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000051": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000052": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000053": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000054": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000055": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000056": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000057": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000058": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000059": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000005a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000005b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000005c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000005d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000005e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000005f": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000060": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000061": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000062": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000063": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000064": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000065": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000066": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000067": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000068": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000069": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000006a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000006b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000006c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000006d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000006e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000006f": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000070": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000071": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000072": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000073": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000074": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000075": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000076": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000077": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000078": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000079": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000007a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000007b": {
```

```
    "balance": "0x1"
  },
  "000000000000000000000000000000000000007c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000007d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000007e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000007f": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000080": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000081": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000082": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000083": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000084": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000085": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000086": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000087": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000088": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000089": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000008a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000008b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000008c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000008d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000008e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000008f": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000090": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000091": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000092": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000093": {
```

```
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000094": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000095": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000096": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000097": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000098": {
    "balance": "0x1"
  },
  "0000000000000000000000000000000000000099": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000009a": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000009b": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000009c": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000009d": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000009e": {
    "balance": "0x1"
  },
  "000000000000000000000000000000000000009f": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a0": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a1": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a2": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a3": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a4": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a5": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a6": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a7": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a8": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000a9": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000aa": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ab": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ac": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ad": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ae": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000af": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b0": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b1": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b2": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b3": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b4": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b5": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b6": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b7": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b8": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000b9": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ba": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000bb": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000bc": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000bd": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000be": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000bf": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c0": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c1": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c2": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c3": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c4": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c5": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c6": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c7": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c8": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000c9": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ca": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000cb": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000cc": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000cd": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ce": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000cf": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d0": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d1": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d2": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d3": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d4": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d5": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d6": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d7": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d8": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000d9": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000da": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000db": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000dc": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000dd": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000de": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000df": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e0": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e1": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e2": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e3": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e4": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e5": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e6": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e7": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e8": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000e9": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ea": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000eb": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ec": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ed": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ee": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ef": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f0": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f1": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f2": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f3": {
```

```
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f4": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f5": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f6": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f7": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f8": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000f9": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000fa": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000fb": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000fc": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000fd": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000fe": {
    "balance": "0x1"
  },
  "00000000000000000000000000000000000000ff": {
```

```
      "balance": "0x1"
    },
    "a6c24f36c5463a1850aab17547ea83c99a56e0a6": {
      "balance": "0x200000000000000000000000000000000000000000000000000000000000000000"
    }
  }
}
```

3.2.3 Solidity IDE

I start the interaction with the blockchain using geth.Please see Appendix A for the bash scripts

/Users/User1/go-ethereum/build/bin/geth –identity "Princess" –rpc –dev –mine -minerthreads 1 –rpccorsdomain "http://localhost:8000" –rpccorsdomain "*" –rpcport "8545" –ipcpath /Users/User1/Library/Ethereum/geth.ipc –datadir /.ethereumprivatepoa –port "30303" –nodiscover –rpcapi "db,eth,net,web3" –networkid 1988 –unlock 0 console 2»geth.log

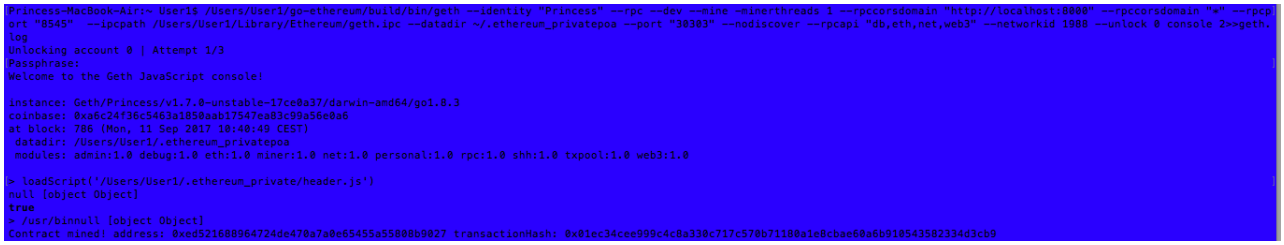

Figure 3.4: ProofofAuthority

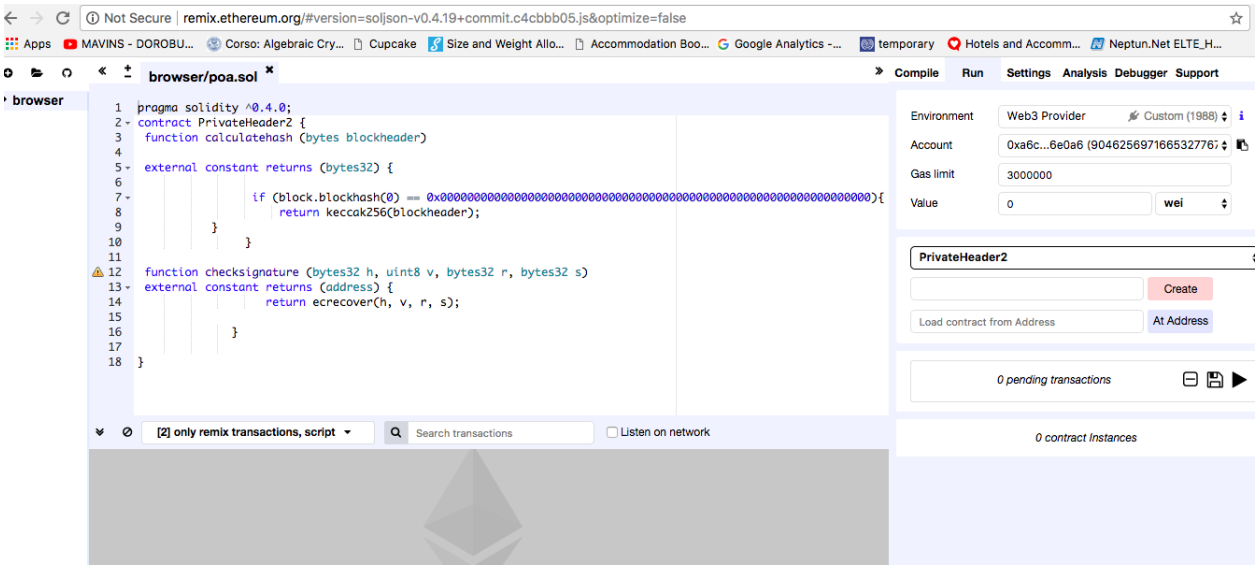

Figure 3.5: RemixIDE

I use the online Remix IDE to write the solidity contracts;

To deploy the contract on geth; I use the Web3 deploy autogenerated script from the contract details on the IDE, remove line breaks and then deploy using the loadScript javascript file command as seen in Figure3.5 above.

3.2.4 PrivateHeader.sol

An account contract that implements a private network by verifying the header. We hash the blockheader fields according to the the order described in Ethereum's yellow paper. We use solidity's ec recover function to verify the blockheader's signature. In POA, with miner.start() on geth, my ethereum account signs the blockhash to authorise it on the network. The resulting signature is 65 bytes containing every header field. Use eth.getBlock(1) on geth to see it embedded into the extra data field as the last 65 bytes.

Please see Appendix A for the source code.

3.2.5 BlockChain Explorer

Using an open source explorer;

```
#git clone https://github.com/etherparty/explorer
cd Library/Ethereum/explorer/
Princess-MacBook-Air:explorer User1$ npm start

EthereumExplorer@0.1.0 prestart /Users/User1/Library/Ethereum/explorer npm install

EthereumExplorer@0.1.0 postinstall /Users/User1/Library/Ethereum/explorer bower install

up to date in 4.63s

EthereumExplorer@0.1.0 start /Users/User1/Library/Ethereum/explorer http-server ./app -a

Starting up http-server, serving ./app on port: 8000 Hit CTRL-C to stop the server [Wed,

#http://localhost:8000/#/
```

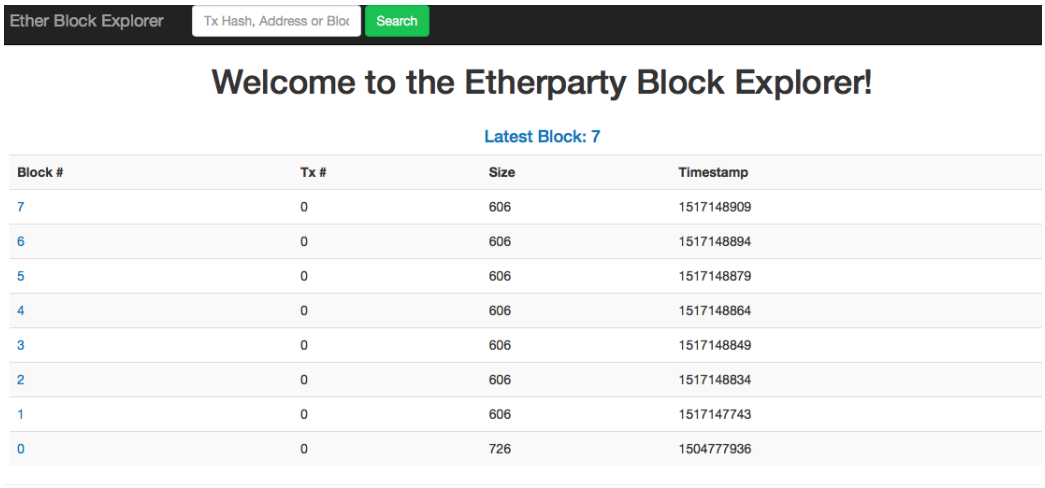

Figure 3.6: PrivateTestNetBlockExplorer

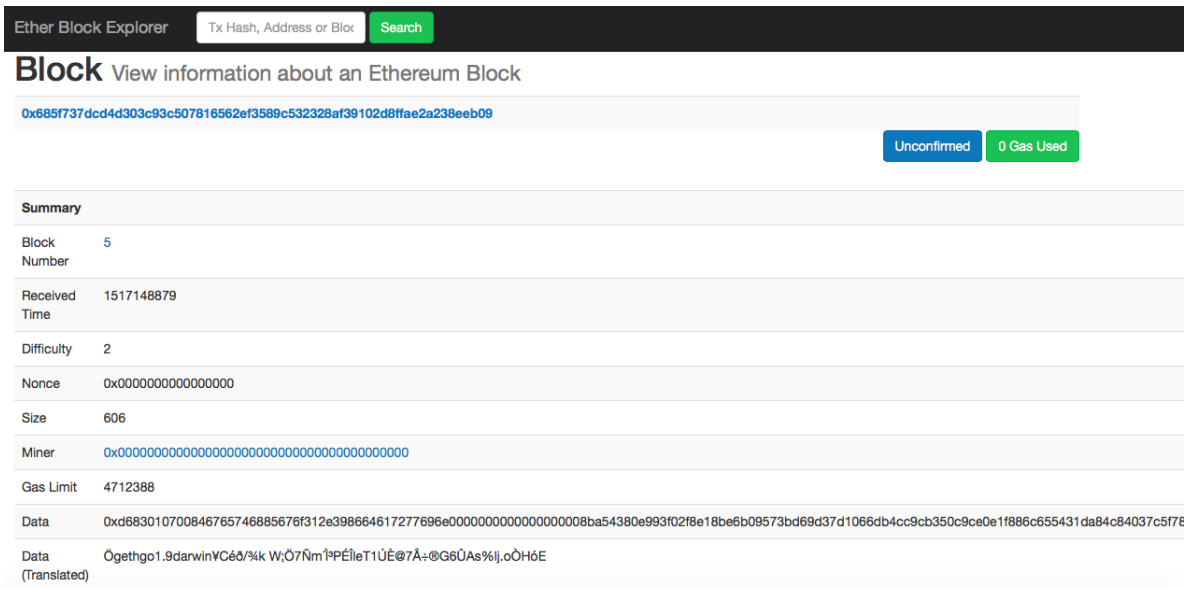

Figure 3.7: BlockInformation

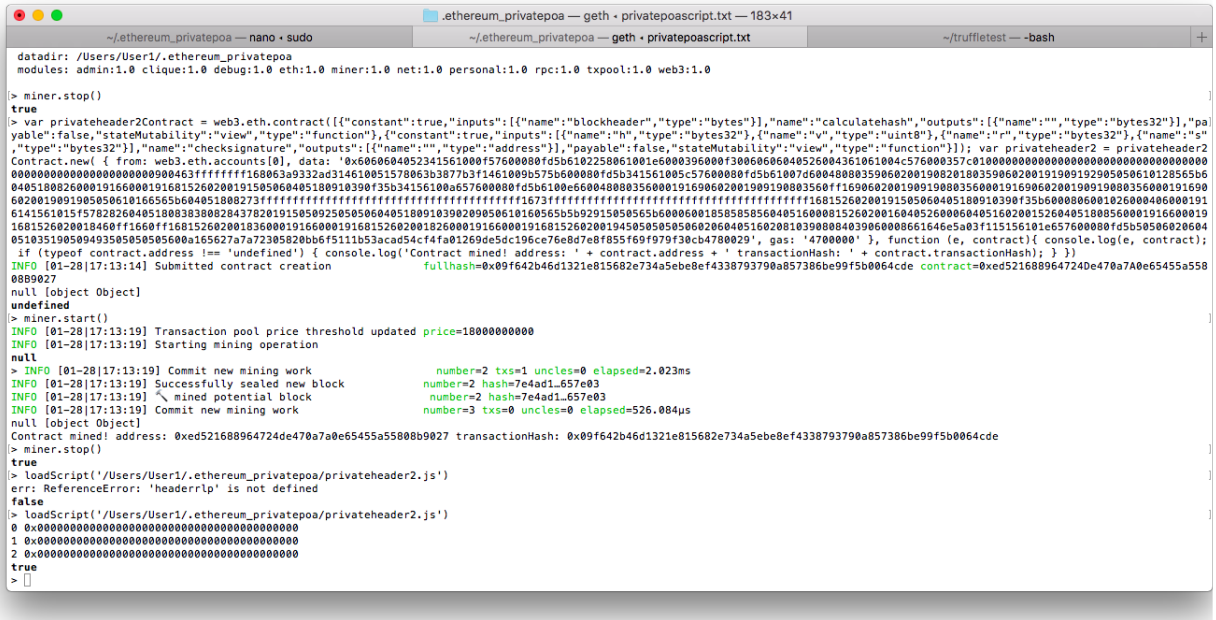

Figure 3.8: Web3Script

3.2.6 Tests

I use truffle to create simple unit tests. I installed truffle using the npm package, I write the .sol contract under contracts, compile it. I copied the web3 script as privateheader.js under migrations. I use migrate to test the network. Finally I use test to deploy and test the scripts.

These last two test functions can be achieved manually in geth by pasting the web3 script directly and using load script for any additional javascript as is seen in Figure 3.8

```
Princess-MacBook-Air:~ User1$ npm install -g truffle@beta
/usr/local/bin/truffle -> /usr/local/lib/node_modules/truffle/build/cli.bundled.js
+ truffle@4.0.0-beta.2
added 91 packages in 11.197s


Princess-MacBook-Air:~ User1$ mkdir truffleproject
Princess-MacBook-Air:~ User1$ cd truffleproject/
Princess-MacBook-Air:truffleproject User1$ truffle init
Downloading project...
Project initialized.
```

```
  Documentation: http://truffleframework.com/docs

Commands:

  Compile: truffle compile
  Migrate: truffle migrate
  Test:    truffle test


Princess-MacBook-Air:truffleproject User1$ ls
contracts migrations test truffle.js


Princess-MacBook-Air:truffletest User1$ cd contracts/
Princess-MacBook-Air:contracts User1$ sudo nano Poa.sol
Princess-MacBook-Air:contracts User1$ ls
Poa.sol

Princess-MacBook-Air:truffleproject User1$ truffle compile /Users/User1/Desktop/smartcon

Compiling ./contracts/Poa.sol...
Writing artifacts to ./build/contracts

Princess-MacBook-Air:truffleproject User1$ ls
build contracts migrations test truffle.js

Princess-MacBook-Air:migrations User1$ sudo nano privateheader.js
Password:
Princess-MacBook-Air:migrations User1$ cd ..


Princess-MacBook-Air:truffletest User1$ truffle test migrations/privateheader.js
Using network 'development'.
```

```
  0 passing (1ms)

Contract mined! address: 0x6f477d90a822d58ca00fdd591fb1f1f454e20949 transactionHash: 0x1
```

Chapter 4

Conclusion

To allow one computing platform that is decentralized in nature to scale, we try to break down the full nodes into components that can verify transactions independently.

We try to adopt the layer3 architecture of the Internet's OSI model in separating public and private transactions.

The information is then stored as rlp encoded information on the blockchain and the contract verifies the header of the test net. The system can then process as many transactions in parallel.

The advantages of this design include being inexpensive, high throughput, low latency and anonymity.

Chapter 5

Bibliography

Bibliography

[1] BitCoin. https://bitcoin.org/bitcoin.pdf. Accessed: 2017-06-14.

[1.1] Mauve Paper. https://scalingbitcoin.org/MauvePaper-VitalikButerin. Accessed: 2017-06-14.

[1.2] Private Network. https://github.com/ethereum/go-ethereum/wiki/Private-network. Accessed: 2017-06-14.

[1.3] Proof of Authority. https://github.com/ethereum/EIPs/issues/225. Accessed: 2017-06-14.

[1.4] Block Header. https://github.com/ethereum/yellowpaper. Accessed: 2017-06-14.

[2] Proof of Stake. https://bitcointalk.org/index.php?topic=27787.0. Accessed: 2017-06-14.

[3] Public and Private BlockChains. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/. Accessed: 2017-06-14.

[4] Sharding FAQ. https://github.com/ethereum/wiki/wiki/Sharding-FAQ. Accessed: 2017-06-14.

[6] Ethereum-Development-Tutorial. https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial. Accessed: 2017-06-14.

[9] Ethereum Introduction. http://ethdocs.org/en/latest/introduction/index.html. Accessed: 2017-06-14.

Appendices

Appendix A

Source code of an Ethereum Solidity Contract to accept a header for a private blockchain

```
pragma solidity ^0.4.0;
contract PrivateHeader2 {
/* Refer to https://github.com/ethereum/EIPs/issues/225, authorising a block */
/* Refer to https://github.com/ethereum/go-ethereum/blob/master/consensus/clique/clique.
/* We use Solidity's ecrecover function to verify signature */
 function checksignature (bytes32 h, uint8 v, bytes32 r, bytes32 s) external constant re
                  return ecrecover(h, v, r, s);

             }

}
```

Web3 deploy of an Ethereum Smart Contract to verify the header on a private blockchain

```
var privateheader2Contract = web3.eth.contract([{"constant":true,"inputs":[{"name":"bloc
```

Script for interacting with blocks generated at runtime

```
"//get elements as they are being mined
getBlockelements = function(number) {
        blockheader = eth.getBlock(number);
        elements = [];
        if (blockheader != null){
          elements.push(/*string*/removehex(blockheader.parentHash),
                       /*string*/removehex(blockheader.sha3Uncles),
```

```
                        /*string*/removehex(web3.eth.coinbase),
                        /*string*/removehex(blockheader.stateRoot),
                        /*string*/removehex(blockheader.transactionsRoot),
                        /*string*/removehex(blockheader.receiptsRoot),
                        /*string*/removehex(blockheader.logsBloom),
                        /*object*/numbertohexstring(blockheader.difficulty),
                        /*number*/numbertohexstring(blockheader.number),
                        /*number*/numbertohexstring(blockheader.gasLimit),
                        /*number*/numbertohexstring(blockheader.gasUsed),
                        /*number*/numbertohexstring(blockheader.timestamp),
                        /*string*/removehex(blockheader.extraData.slice(0,blockheader.ex
                        /*string*/removehex(blockheader.mixHash),
                        /*string*/removehex(blockheader.nonce))
           return elements;
        }
        else
          throw false
         }
//test for getBlockelements
getBlockelementstest = function(){
  testcaseelement = ["0000000000000000000000000000000000000000000000000000000000000000",
  for (t = 0; t<testcaseelement.length; t++){
    if(testcaseelement[t] != getBlockelements(0)[t]) {
          throw "Expected " + testcaseelement[t] + ", got " + getBlockelements(0)[t];
}
}
    return true;
}


//convert number to a hex string for rlp encoding
numbertohexstring = function(number) {
    y = number.toString(16);
    //if (number == 0)  * removed this after testdecode() was failing
      //return "";
 if (((y.length) % 2) != 0){
```

```
    return "0" + y;
    }
  else
      return y;
}



numbertohexstringtest = function() {
  numbertostring =  { 1:"01", 255:"FF", 256:"0100", 65535:"FFFF", 65536:"010000"}
  //keys = Object.keys(numbertostring)
   for(input in numbertostring) {
      console.log("Testing " + input);
      if(numbertostring[parseInt(input)]!= numbertohexstring(parseInt(input)).toUpperCas
      throw "Expected " + numbertostring[parseInt(input)] + ", got " + numbertohexstring
    }
  }
  return true;
}



//match a hexstring
checkstring = function(string){
    hexstring = /([0-9A-F][0-9A-F])+/
    if (hexstring.exec(string) != null){
      return string
    }

    else if (string == ""){
      return string
    }

    else
      throw false
}
```

```
checkstringtest = function() {
   hextest =  {0x00:"0x00", "":""}
   for(input in hextest) {
      console.log("Testing " + input);
      if(hextest[input] != checkstring(input)) {
      throw "Expected " + testcases[input] + ", got " + checkstring(input);
    }
  }
  return true;
}

removehex = function(string){
  return string.replace(/(0x)*/,"")
}

removehextest = function(){
  nohextest = {0x00:"00"}
  for(input in nohextest) {
      console.log("Testing " + input);
      if(nohextest[input] != removehex(input)) {
      throw "Expected " + nohextest[input] + ", got " + removehex(input);
    }
  }
  return true;
}



//test for rlp function
testrlp = function() {
// "input":"expectedoutput"
testcases = {"00":"00", "01":"01", "7F":"7F","81":"8181", "FF":"81FF","0100":"820100", "
/* <55 bytes
rlp("0x951918b572dd2606569e36fc62a6ab95e7642898877794c035fa0f6762356d6d")
"a10x951918b572dd2606569e36fc62a6ab95e7642898877794c035fa0f6762356d6d"
*/
```

```
for(input in testcases) {
  console.log("Testing " + input);
  if(testcases[input] != rlp(input)) {
      throw "Expected " + testcases[input] + ", got " + rlp(input);
}
}
  return true;
}



//rlp encoding:-https://github.com/ethereum/wiki/wiki/RLP
rlp = function(string) {
 checkstring(string);

 if(string.length == 2) {
       z = parseInt(string,16)
       if (0 <= z && z <= 127) {
      return  string;
     }
  }

  else if (string == ""){
      return "80";
     }

  else if ((string.length/2) > 55) {
        return numbertohexstring(0xb7 + numbertohexstring((string.length/2)).length/2) +
    }
        return numbertohexstring((0x80 + string.length/2)) + string;
  }



rlpheader = function(elements){
           headerrlp = [];
           for (n = 0; n<elements.length; n++){
              headerrlp.push(rlp(elements[n]));
```

```
            }
            return headerrlp;
}


  decode = function(string) {
    checkstring(string);
    byt = string.substring(0,2)

    if(parseInt(byt,16) >= 0 && parseInt(byt,16) <= 127) {
               return byt;
     }

    else {
    prefix = parseInt(byt,16);
    }

    if (prefix >= 128 &&  prefix <= 183) {
        shortpayloadlength = prefix - 0x80                //shortpayloadlength = 2digits
        return string.substring(2,2+shortpayloadlength*2)
     }

    if (prefix >= 184 && prefix <= 191){
          longpayloadlength = prefix - 0xb7
          longpayload = parseInt((string.substring(2,2+longpayloadlength*2)),16)
          endposition = 2+longpayloadlength*2
          return string.substring(endposition,endposition+longpayload*2)
    }

    return error;
 }


//test for decode function
testdecode = function() {
// "stringput":"expectedoutput"
```

```
//testencodedcases = {00:"0x00", 15:"0x0f", 1024:"0x82004000","0x00":"0x00", "0x0f":"0x0
testencodedcases = {"00":"00", "01":"01", "7F":"7F","8181":"81","81FF":"FF","820100":"01
for(key in testencodedcases) {
  console.log("Testing " + key);
    if(testencodedcases[key] != decode(key)) {
      throw "Expected " + testencodedcases[key] + ", got " + decode(key);
    }
}
return true;
}


//ecrecover with solidity contract
verifysignature = function (string) {
        headerrlp = rlp(elements)
        h = web3.sha3(headerrlp)
        sig  = blockheader.extraData.slice(blockheader.extraData.length-130)
        r = "0x" + sig.slice(0,64)
        s = "0x" + sig.slice(65,129)
        v = sig.slice(130-132)
        return privateheader2.getadd(h, v, r, s);

}
```

Script for encoding data

```
RLP = require('rlp')
headerrlp = RLP.encode(elements)
```

Bash Script for starting Proof of Authority private test chain

```
#!/bin/bash
echo remove any oldblockchain databases
rm -rf /Users/User1/Library/Ethereum/geth/lightchaindata/*
rm -rf /Users/User1/Library/Ethereum/geth/chaindata/*
rm -rf /Users/User1/.ethereum_privatepoa/geth/lightchaindata/*
rm -rf /Users/User1/.ethereum_privatepoa/geth/chaindata/*
```

```
echo starting GenesisPOA block
/Users/User1/go-ethereum/build/bin/geth --identity "Princess" --rpc --rpccorsdomain "htt

echo starting GenesisPOA console
/Users/User1/go-ethereum/build/bin/geth --identity "Princess" --rpc --mine -minerthreads
```

Bash Script for starting Proof of Work private test chain

```
#!/bin/bash
echo remove any oldblockchain databases
rm -rf /Users/User1/Library/Ethereum/geth/lightchaindata/*
rm -rf /Users/User1/Library/Ethereum/geth/chaindata/*
rm -rf /Users/User1/.ethereum_privatepoa/geth/lightchaindata/*
rm -rf /Users/User1/.ethereum_privatepoa/geth/chaindata/*

echo starting GenesisPOA block
/Users/User1/go-ethereum/build/bin/geth --identity "Princess" --rpc --rpccorsdomain "htt

echo starting GenesisPOA console
/Users/User1/go-ethereum/build/bin/geth --identity "Princess" --rpc --mine -minerthreads
```