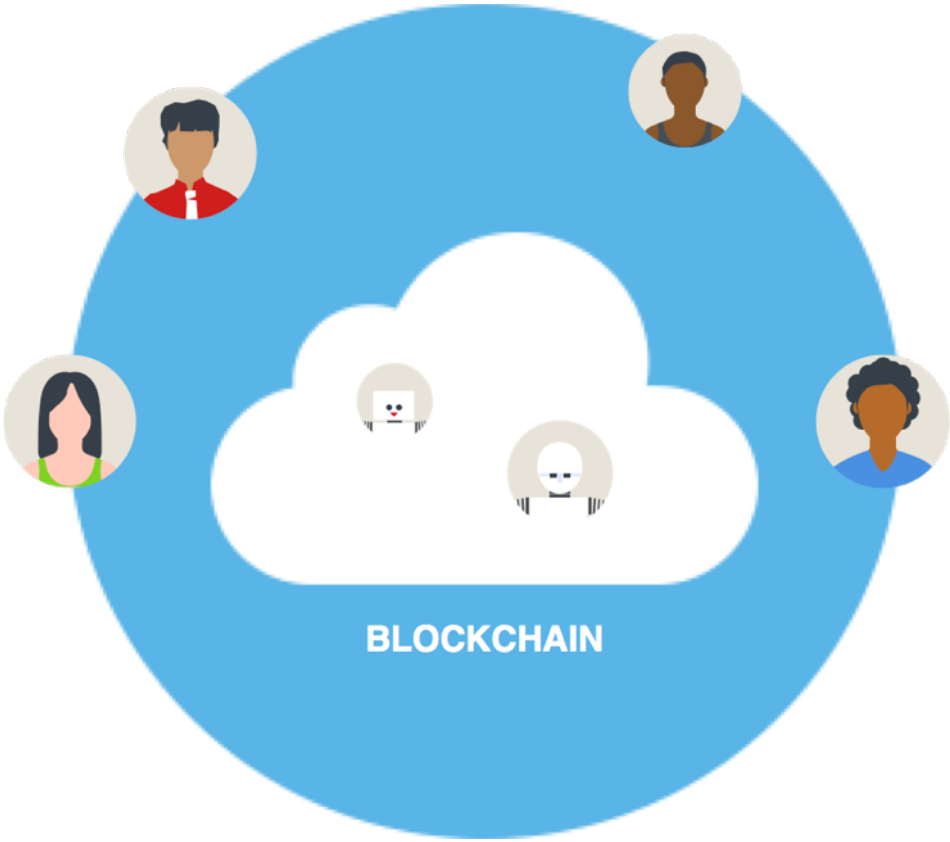
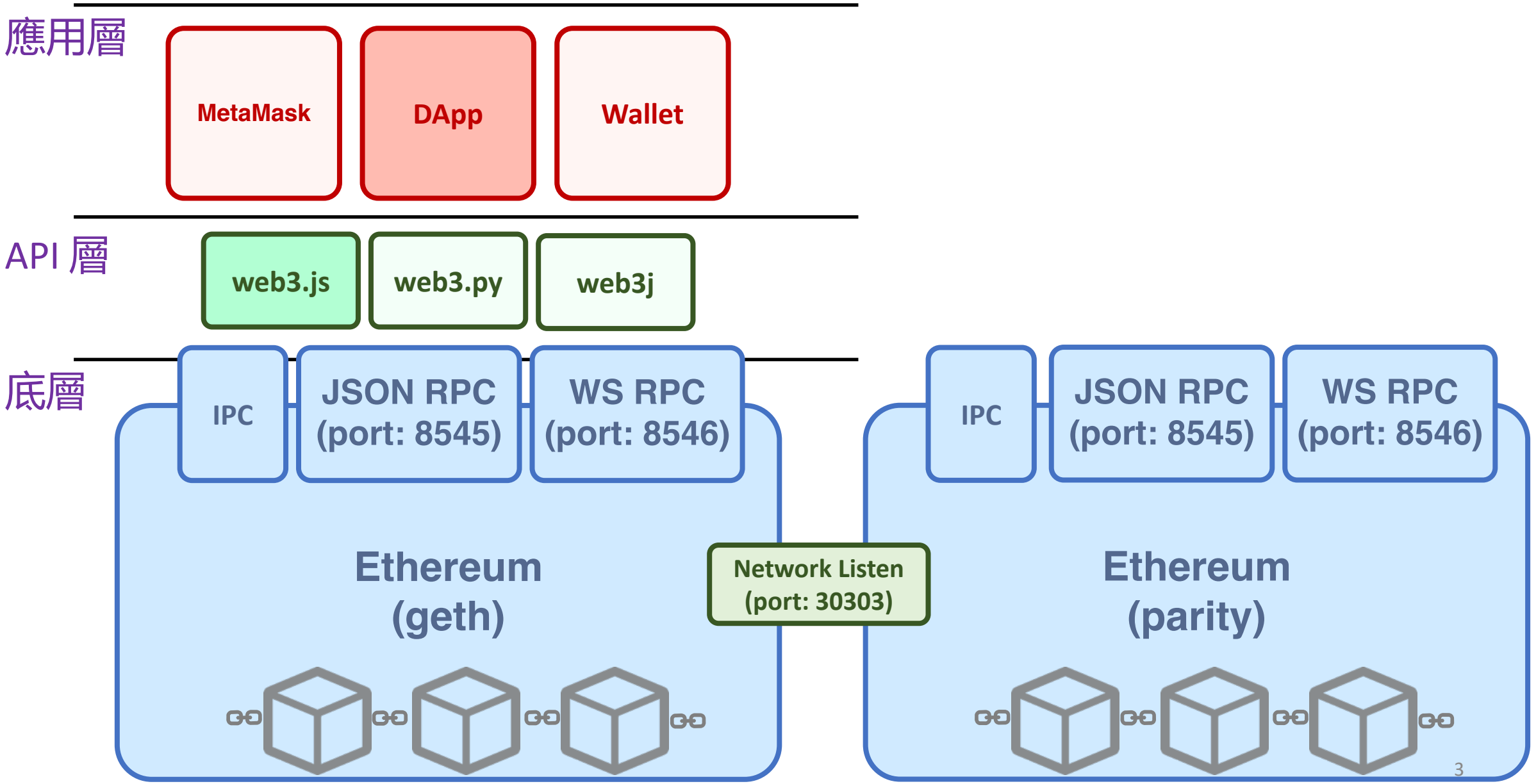
Ethereum protocol

以太坊協議

Why Ethereum?

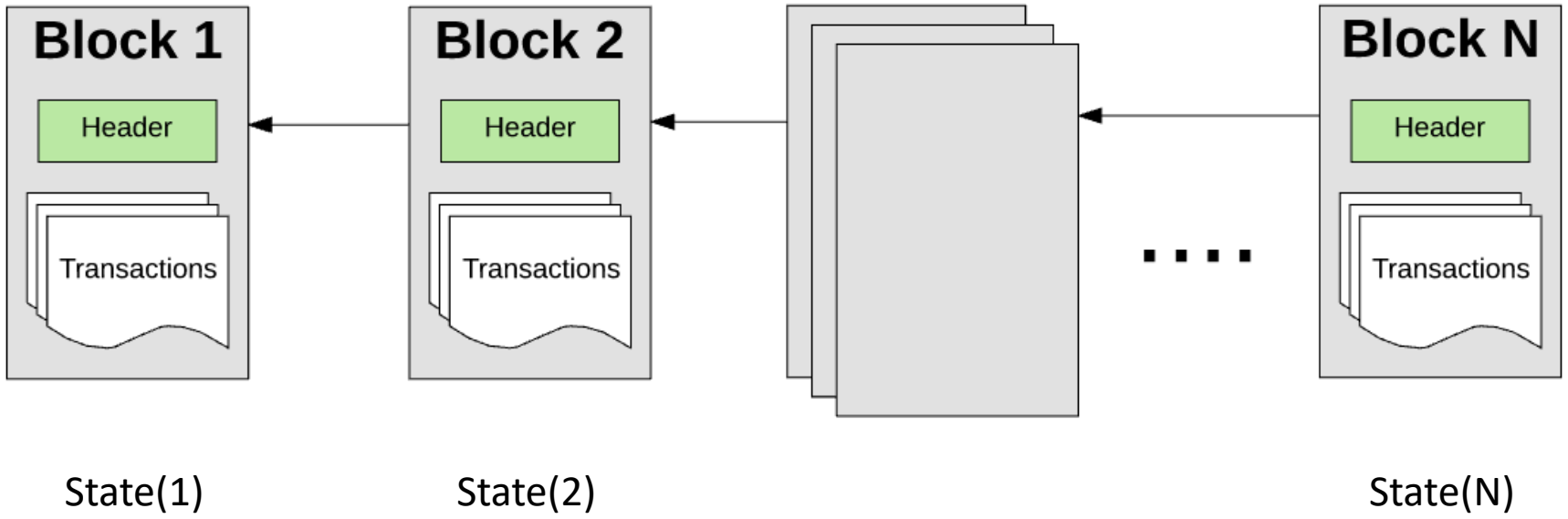
- 受比特幣啟發 (電子貨幣)
- 不只是電子貨幣，打造 World Computer
- 分散式應用平台，以區塊鏈為底
- 開發任何智 Dapp (智能合約)
  - Token is one of the DApp



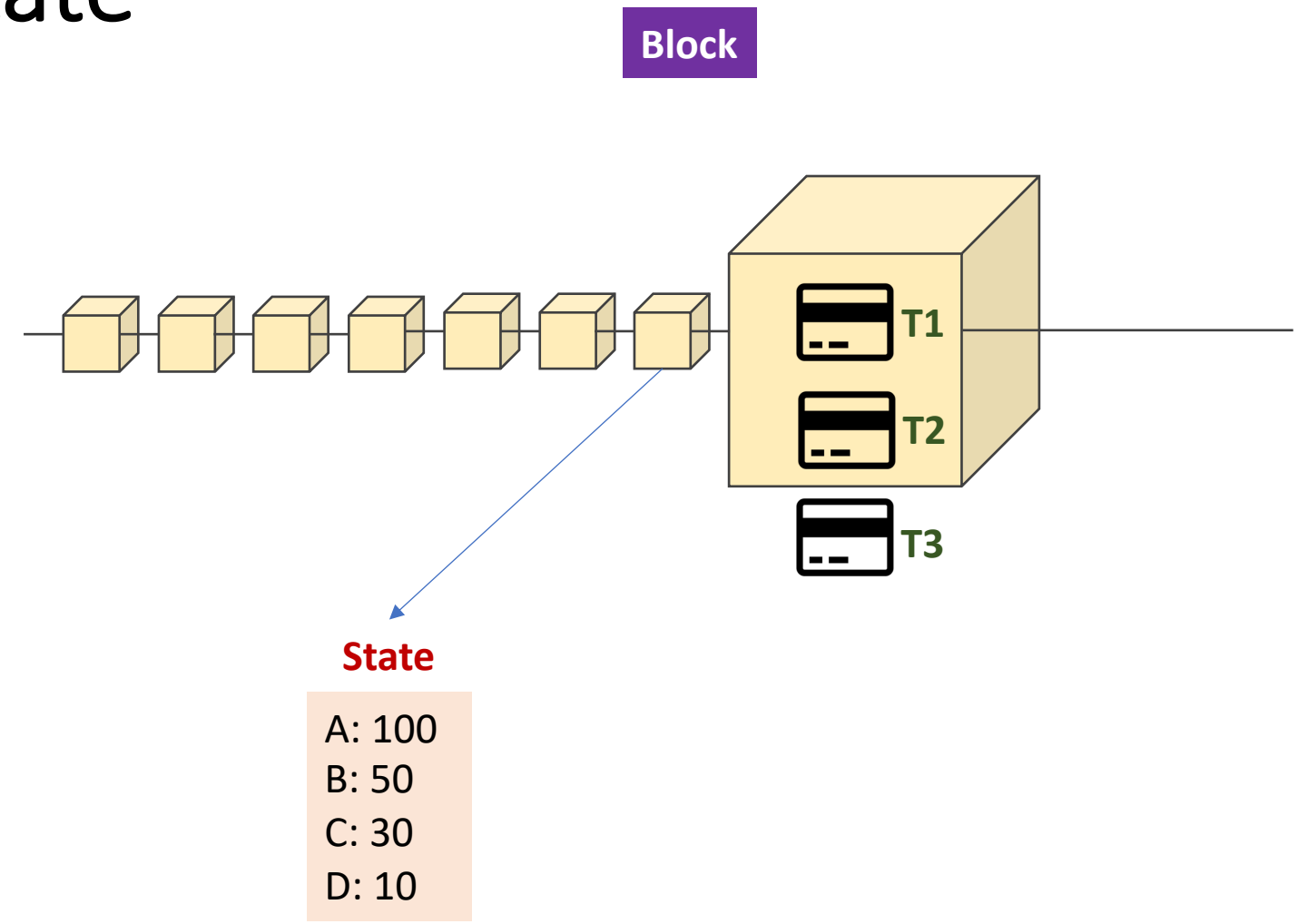

應用層
MetaMask
DApp
Wallet
API 層
web3.js
web3.py
web3j
底層
IPC
JSON RPC
(port: 8545)
WS RPC
(port: 8546)
Ethereum
(geth)
Network Listen
(port: 30303)
IPC
JSON RPC
(port: 8545)
WS RPC
(port: 8546)
Ethereum
(parity)

帳戶

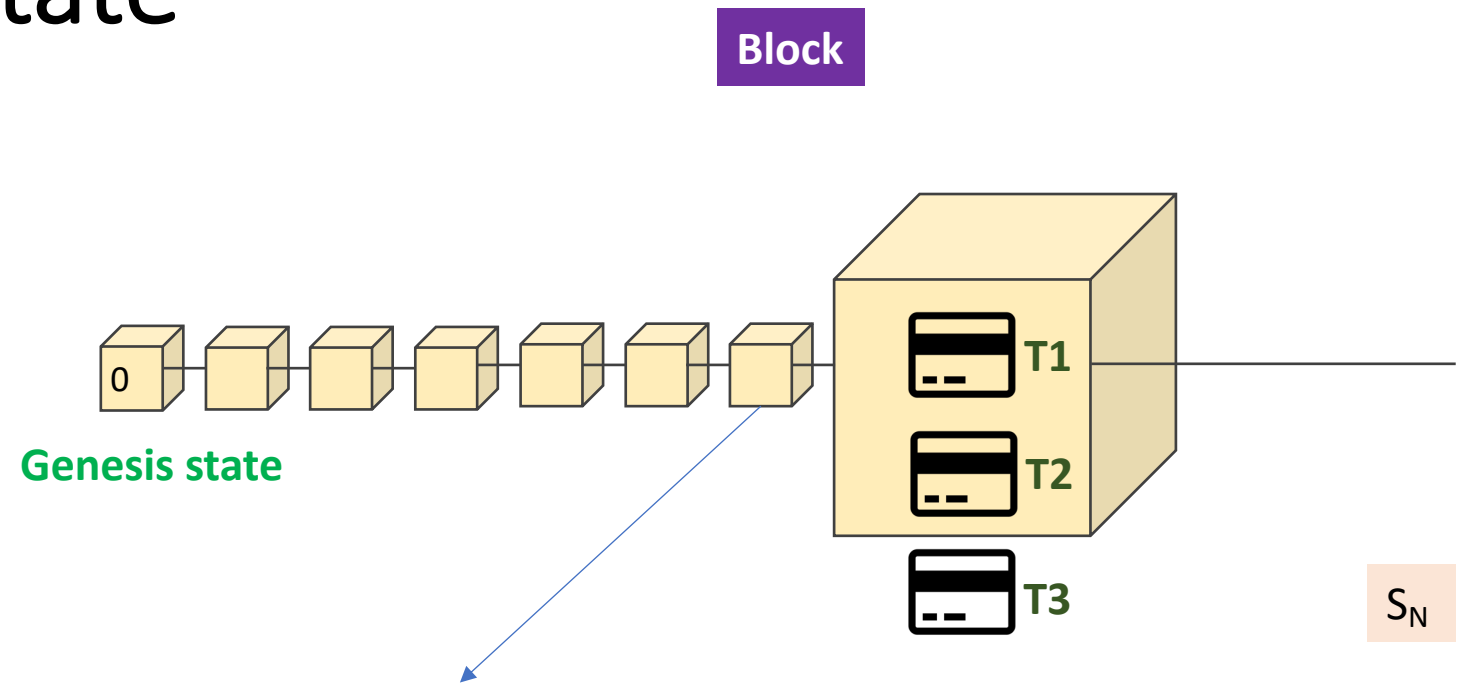
- 在以太坊中，全局狀態 (Global State) 是由許多的帳戶物件 (Account Object) 所組成
- 帳戶之間可以透過訊息 (Message) 傳遞進行互動
- 每個帳號會關聯到一個狀態 (State) 與一個 160 位元的地址(Address)



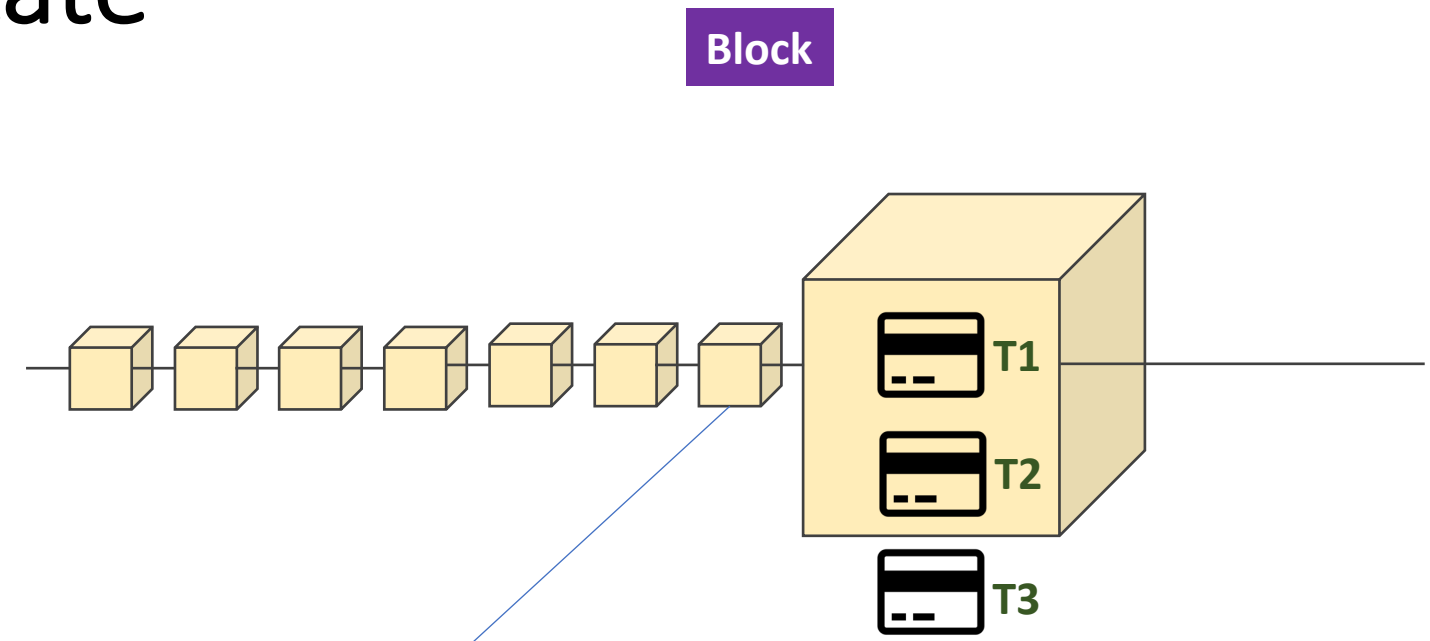
Global state



Global state



Global state



State

A: 100
B: 50
C: 30
D: 10

T1: A → B: 20

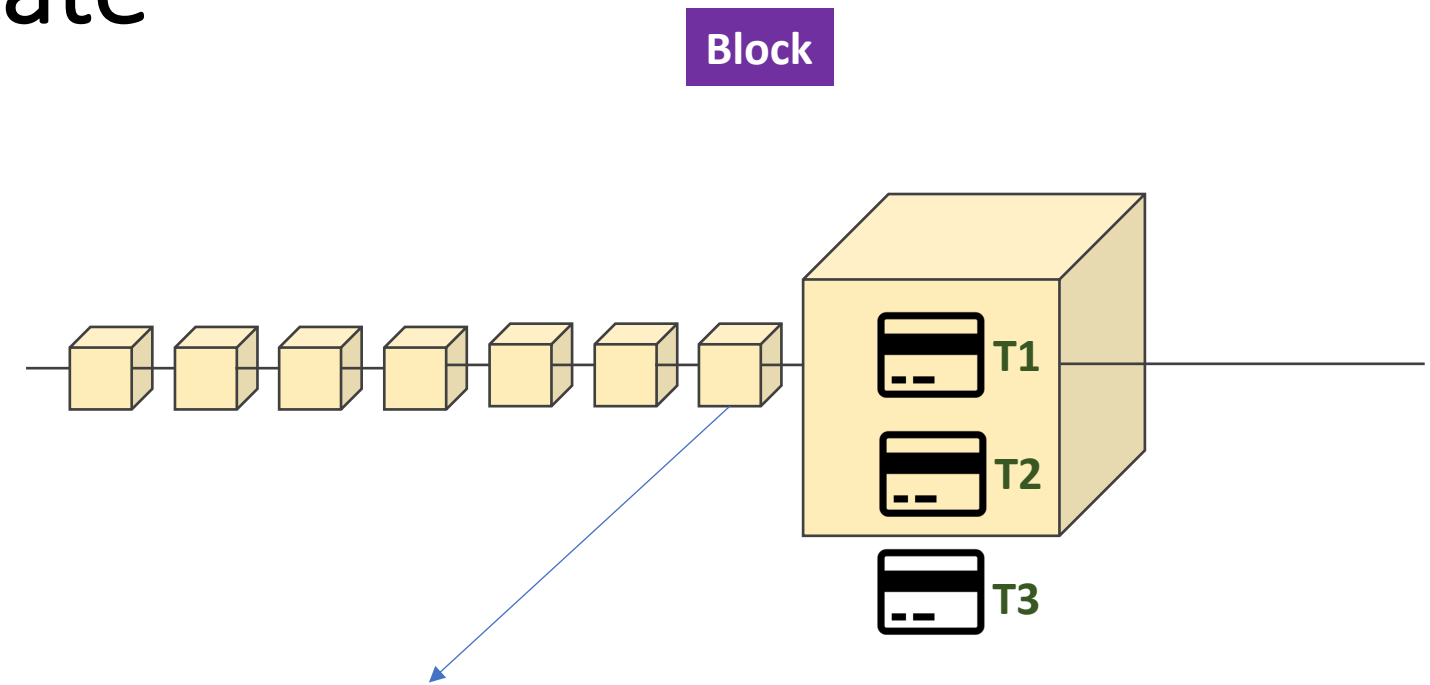
T2: A → D: 20

A: 80
B: 70
C: 30
D: 10

State''

A: 60
B: 70
C: 30
D: 30

Global state



State

A: 100
B: 50
C: 30
D: 10

T1: A → B: 20

T2: A → D: 20

T3: C → D: 10

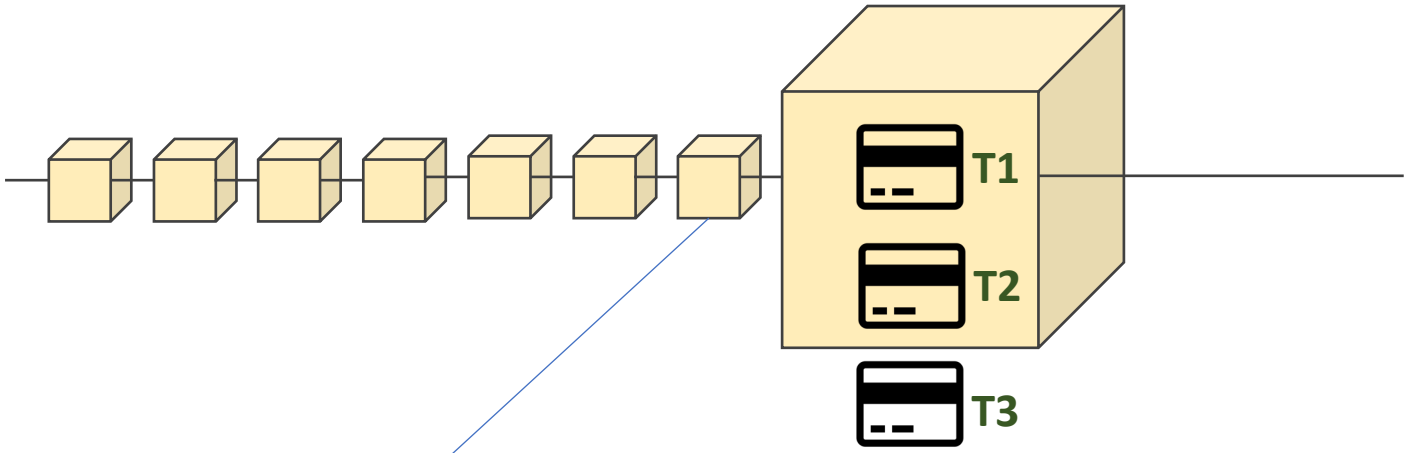
A: 80
B: 70
C: 30
D: 10

A: 60
B: 70
C: 30
D: 30

State'''

A: 60
B: 70
C: 20
D: 40

Global state



Ledger state

State

A: 100
B: 50
C: 30
D: 10

T1: A → B: 20

T2: A → D: 20

T3: C → D: 10

A: 80
B: 70
C: 30
D: 10

A: 60
B: 70
C: 30
D: 30

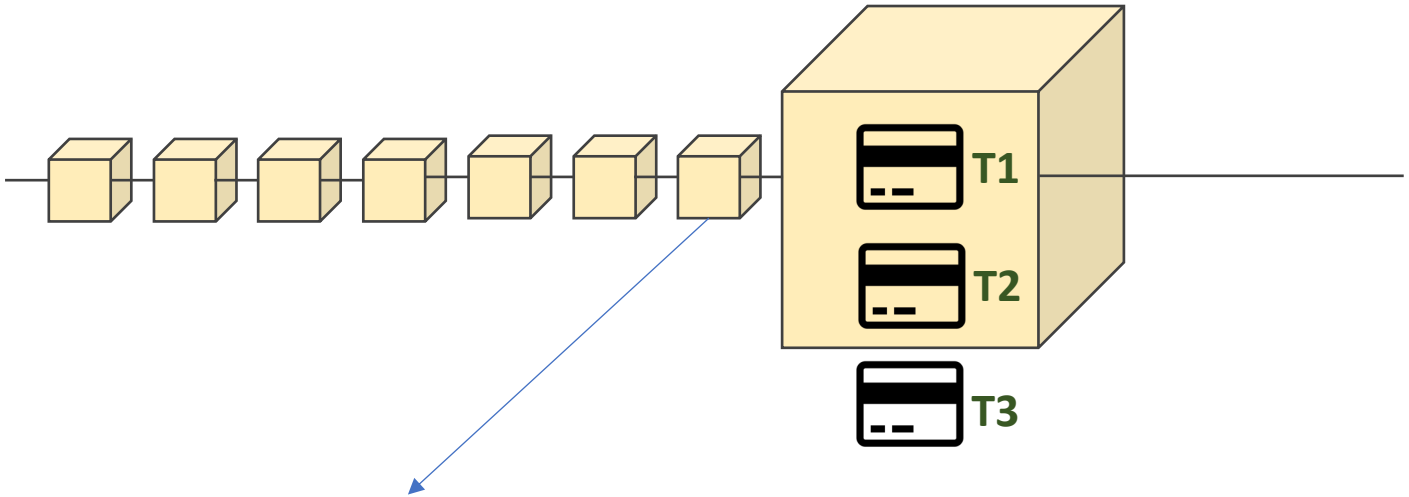
State'''

A: 60
B: 70
C: 20
D: 40

An ordering of transactions in the blockchain

Global state

1. 每個新區塊就是當前的 global state 有什麼好處?
2. 這類 state machine 設計的好處在哪?



Ledger state

State

A: 100
B: 50
C: 30
D: 10

T1: A → B: 20

T2: A → D: 20

T3: C → D: 10

A: 80
B: 70
C: 30
D: 10

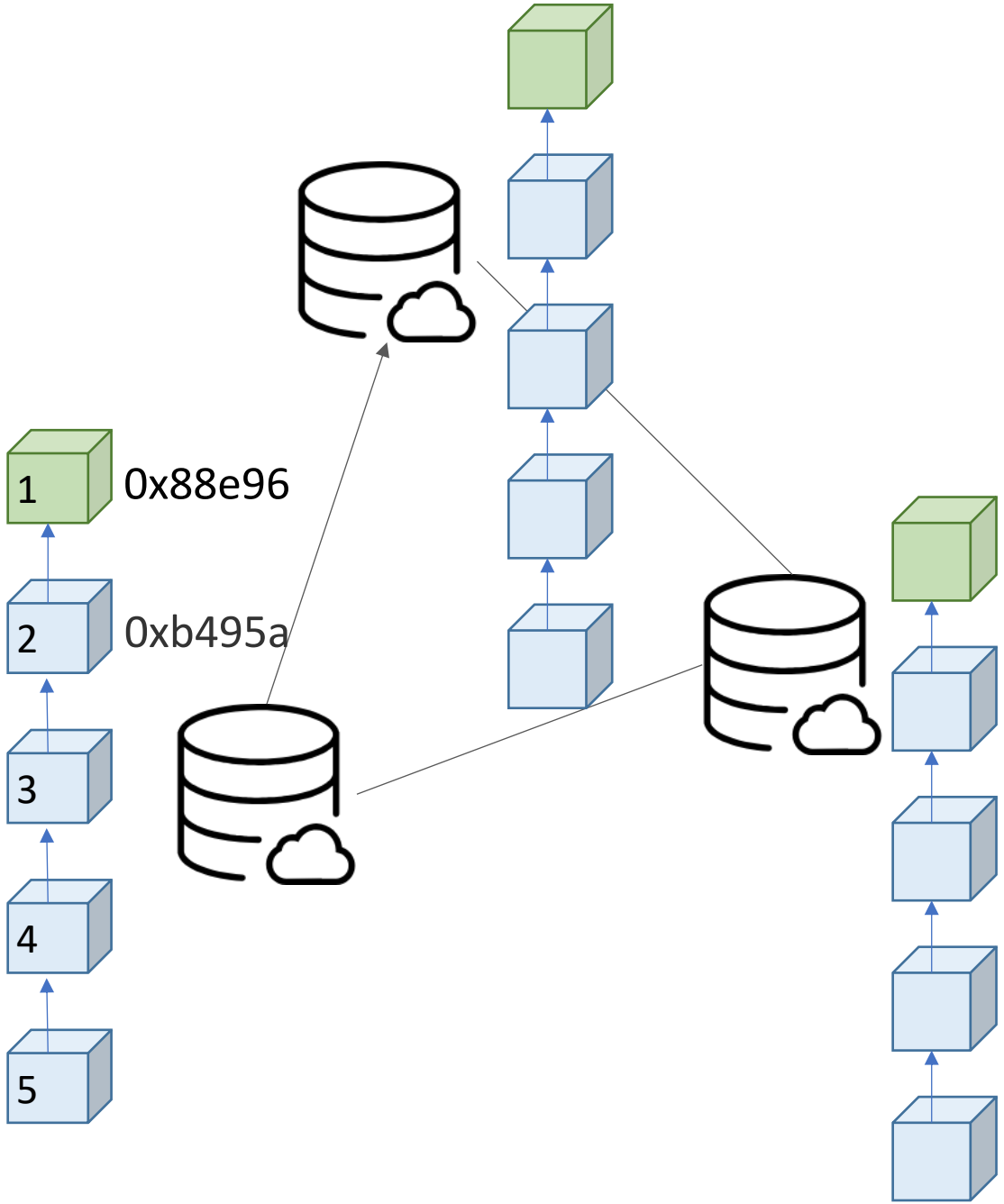
A: 60
B: 70
C: 30
D: 30

State'''

A: 60
B: 70
C: 20
D: 40

Synchronization

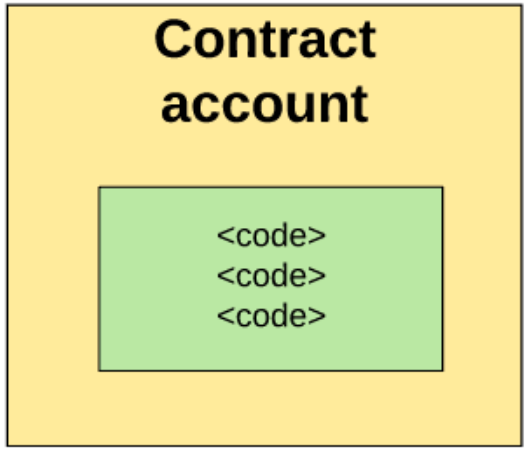
- 區塊同步
- 快速同步
- 復原同步



https://github.com/ethereum/go-ethereum/wiki/Connecting-to-the-network

帳戶類型

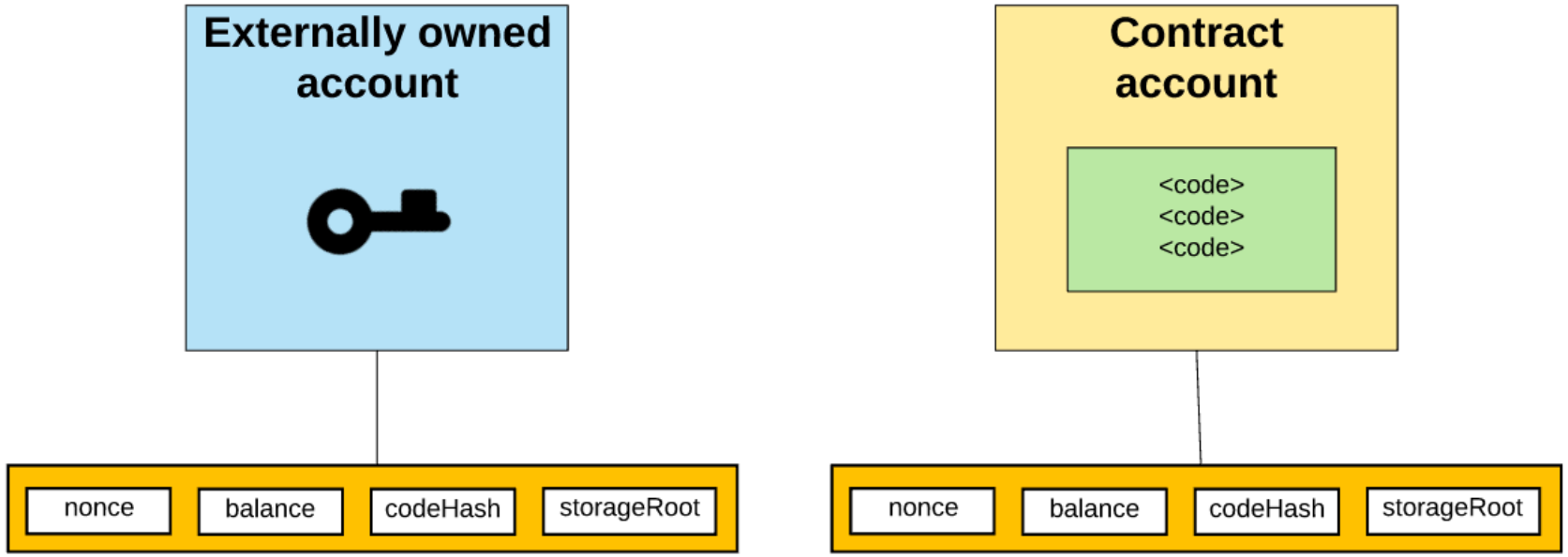
- 在以太坊中，帳戶主要分兩種類型
- 外部帳戶 (External owned account, EOA), 由私鑰進行控制
- 合約帳戶 (Contract account), 與合約程式碼相關聯



帳戶狀態 (Account state)

- Nonce
- Balance
- codeHash
- storageRoot

怎麼分辨帳戶是 EOA 還是合約地址?



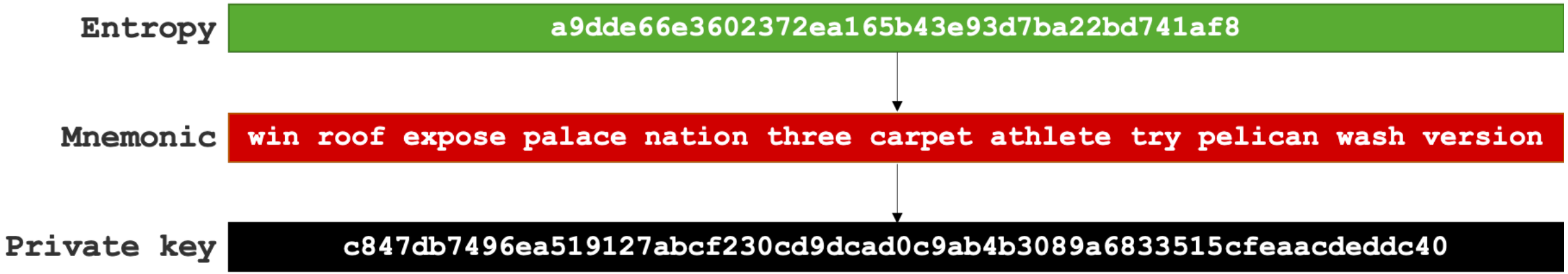
> eth.getCode("0xfec2394d5952828b00f4420caa7d0cd887846b09")

https://gist.github.com/changwu-tw/2d312757589fac5e4a94a9dddf9c0edd

EOA

1. Create a keypair of private/public key
2. public_key = ECDSA(private_key)
3. public_key_hash = Keccak-256(public_key)
4. address = '0x' + last 20 bytes of public_key_hash

Mnemonic Phrase (BIP39)



1. Create a keypair of private/public key
2. public_key = ECDSA(private_key)
3. public_key_hash = Keccak-256(public_key)
4. address = '0x' + last 20 bytes of public_key_hash

Contract address

1. Create a keypair of private/public key
2. public_key = ECDSA(private_key)
3. public_key_hash = Keccak-256(public_key)
4. address = '0x' + last 20 bytes of public_key_hash

contract_address = sha3(rlp_encode([address, nonce]))[14:]

ECDSA

- Private key: **d**
- Public key: **d*G**

p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
a = 0
b = 7
G_x = 0x79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
G_y = 0x483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8
n = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

Sign

- Hash(message): e
- Secret number: **k**
- $r_x = k*G$
- $s = \mathbf{k^{-1}} *(e+d*r_x)$
- sig: (r_x, s)

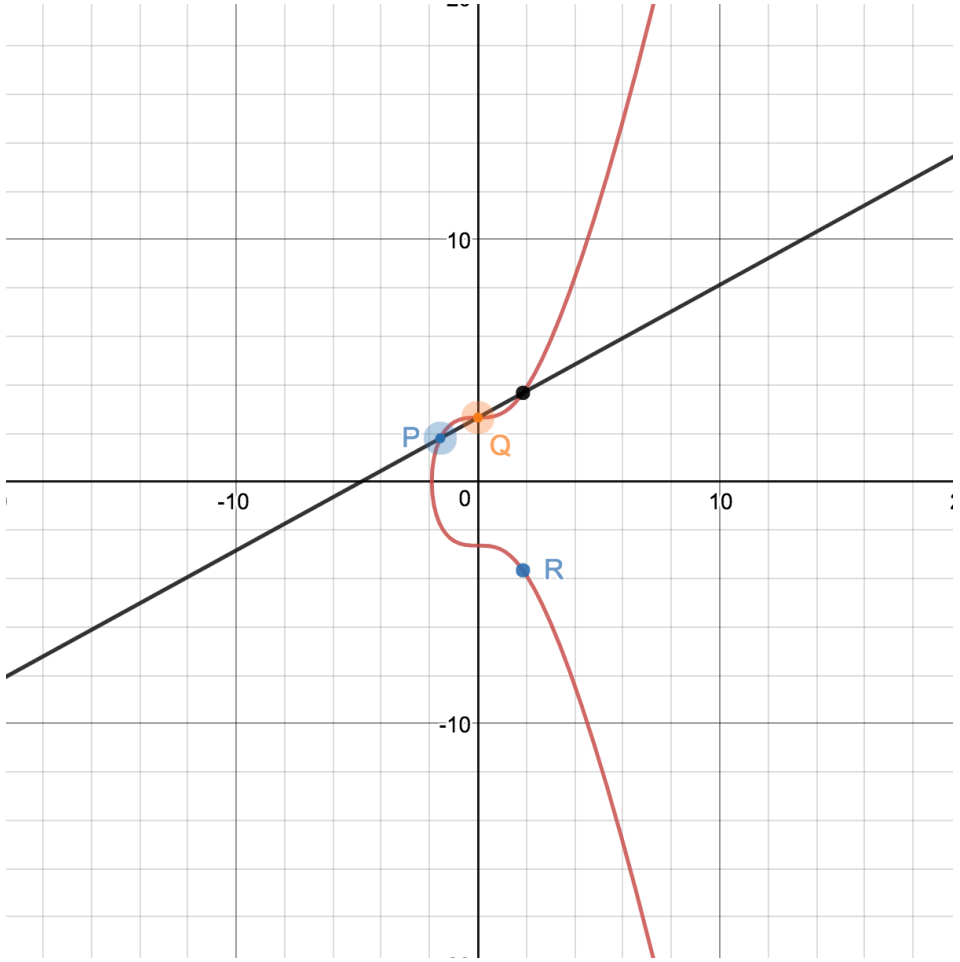
Verify

- Compute s^{-1}
- $r_x = (e*s^{-1}) * G + (r*s^{-1}) *(\mathbf{d*G})$
- Compare r == r_x

secp256k1

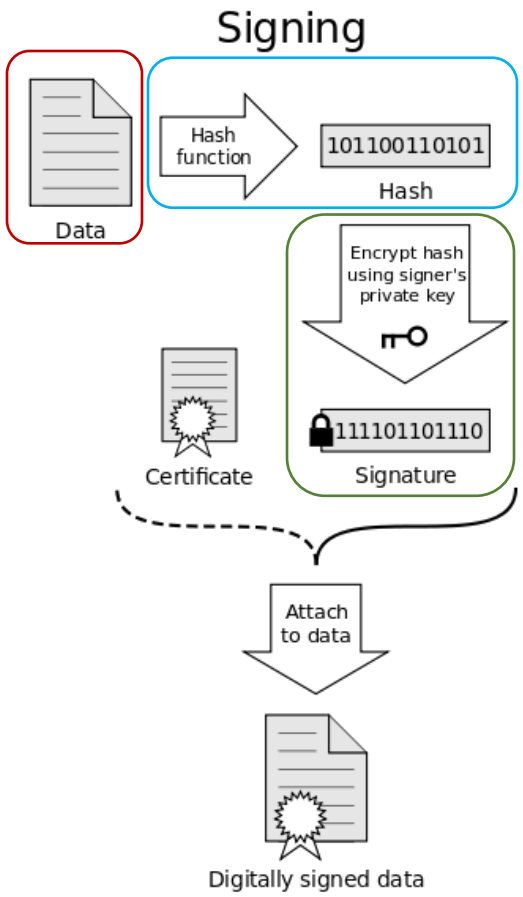
$$y^2 = x^3 + ax + b \pmod{p}$$

$$y^2 = x^3 + 7$$



p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
a = 0
b = 7
G_x = 0x79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
G_y = 0x483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8
n = 0xFFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

ECDSA on Ethereum



EIP#155

```
transaction = {
    nonce: web3.toHex(9),
    gasPrice: web3.toHex(20000000000),
    gasLimit: web3.toHex(21000),
    to: '0x3535353535353535353535353535353535353535',
    value: web3.toHex(1000000000000000000),
    data: '',
    v: web3.toHex(1),
    r: 0x0,
    s: 0x0
}
```

rlp + hash

Hash(message): e

```
0xdaf5a779ae972f972197303d7b574746c7ef83eadac0f2791ad23db92e4c8e53
```

sign with privateKey
and modify v

```
v:  '0x25'
r:  '0x28ef61340bd939bc2195fe537567866003e1a15d3c71ff63e1590620aa636276'
s:  '0x67cbe9d8997f761aecb703304b3800ccf555c9f3dc64214b297fb1966a3b6d83'
```

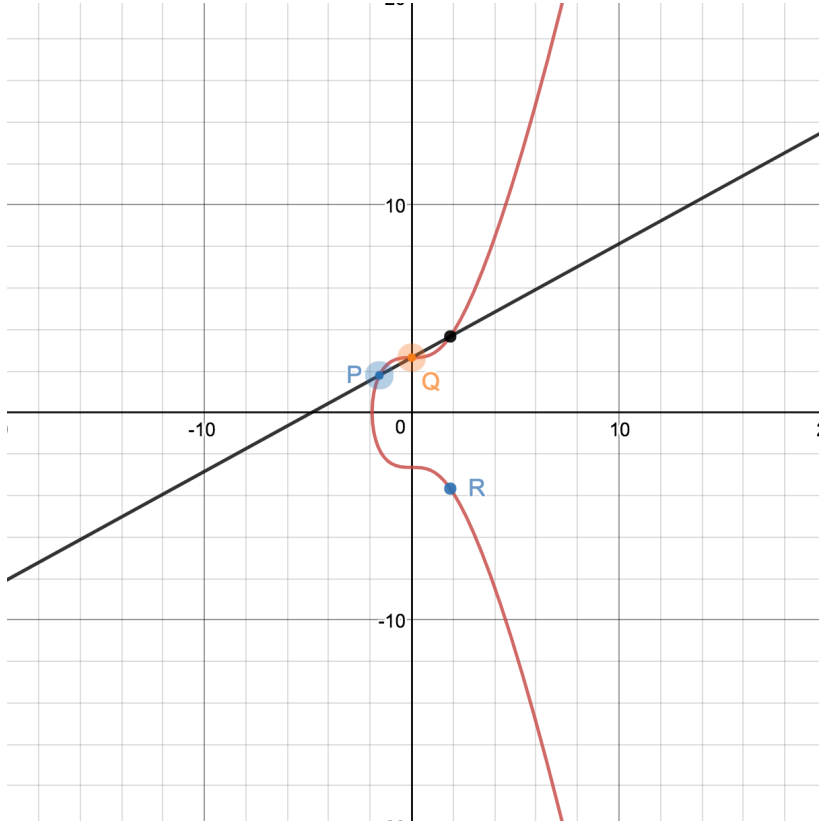
signature

(r, s): each 32 bytes

ECDSA (簽名)

- d_A: Private key (隨機數)
- Q_A: Public key (曲線上的點)
- $Q_A = d_A \times G$ (G 是曲線上的某個點)

- 假設要對檔案進行簽名，會用到
  1. 計算檔案的 Hash 演算法 (ex: SHA256, KECCAK-256)
  2. 用私鑰簽名得到的簽章
  3. 數位簽章包含兩個部分: r 與 s

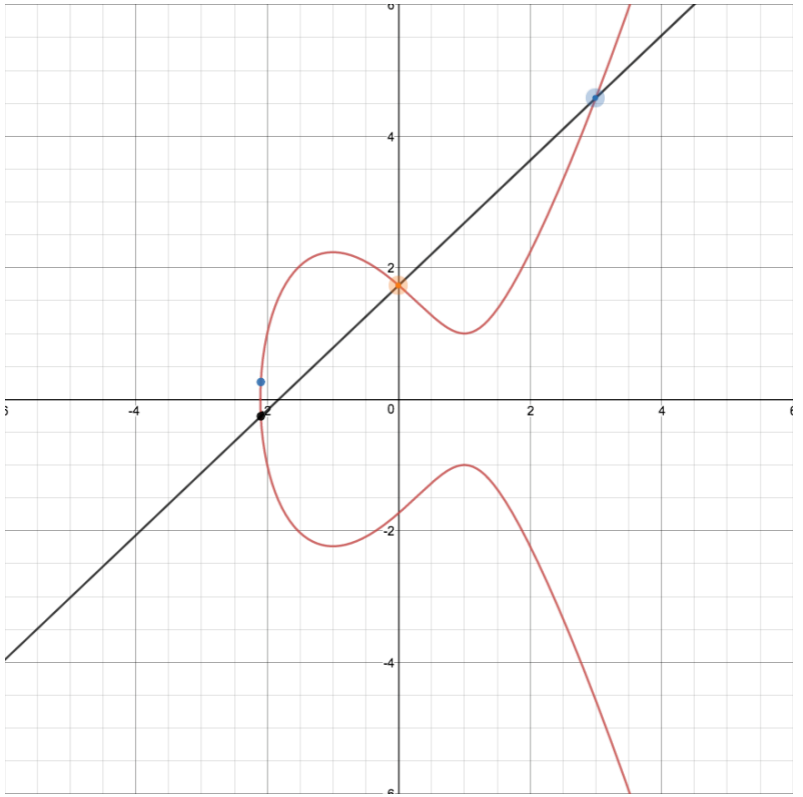


$$y^2 = x^3 + 7$$

ECDSA (驗證)

- ~~d_A: Private key (隨機數)~~
- Q_A: Public key (曲線上的點)
- $Q_A = d_A \times G$ (G 是曲線上的某個點)

- 假設要對數位簽章進行驗證
  1. 從簽章中取出 s
  2. 代回橢圓曲線計算，得到 r
  3. 若 r 有效，則簽章合法



$$y^2 = x^3 - 3x + 3$$

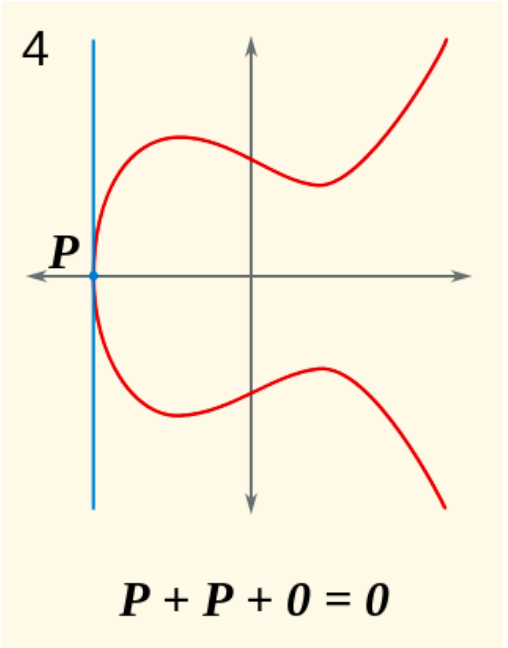
ps. 目前沒有方法能從公鑰快速推得私鑰

Trap-door function

- 巧門函數
  - 單向驗證計算簡單，但逆推困難

為了求 k, 目前的算法最快是 $O(\sqrt{n}) \approx O(2^{128})$

- RSA
  - 給定一個大數，試找出兩個質數的乘積等於這個大數？
- ECC
  - 給定曲線上的起點與終點，試問經過多少個 hop？
  - $P + P + P + P + \cdots + P = k * P$



知道 k 要驗算相對容易，但要找到 k 很困難

* Discrete Logarithm Problem (DLP)

Transaction Hash=Keccak-256(RLP(nonce, gasPrice, gasLimit, to, value, data, v, r, s))

Raw transaction format

```
transaction = {
    nonce: web3.toHex(9),
    gasPrice: web3.toHex(20000000000),
    gasLimit: web3.toHex(21000),
    to: '0x3535353535353535353535353535353535353535',
    value: web3.toHex(1000000000000000000),
    data: '',
    v: web3.toHex(1),
    r: 0x0,
    s: 0x0
}
```

Data

rlp + hash

0xdaf5a779ae972f972197303d7b574746c7ef83eadac0f2791ad23db92e4c8e53

sign with privateKey and modify v

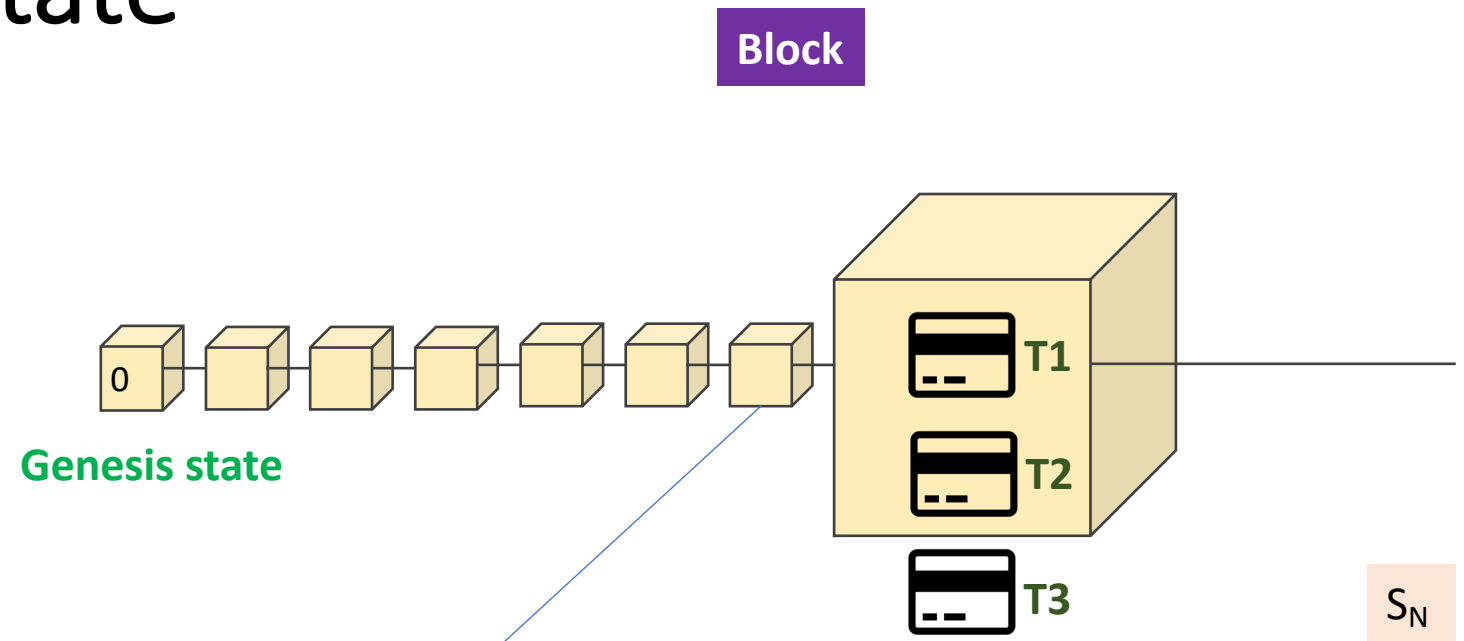
Signature

```
v:  '0x25'
r:  '0x28ef61340bd939bc2195fe537567866003e1a15d3c71ff63e1590620aa636276'
s:  '0x67cbe9d8997f761aecb703304b3800ccf555c9f3dc64214b297fb1966a3b6d83'
                              signature
```

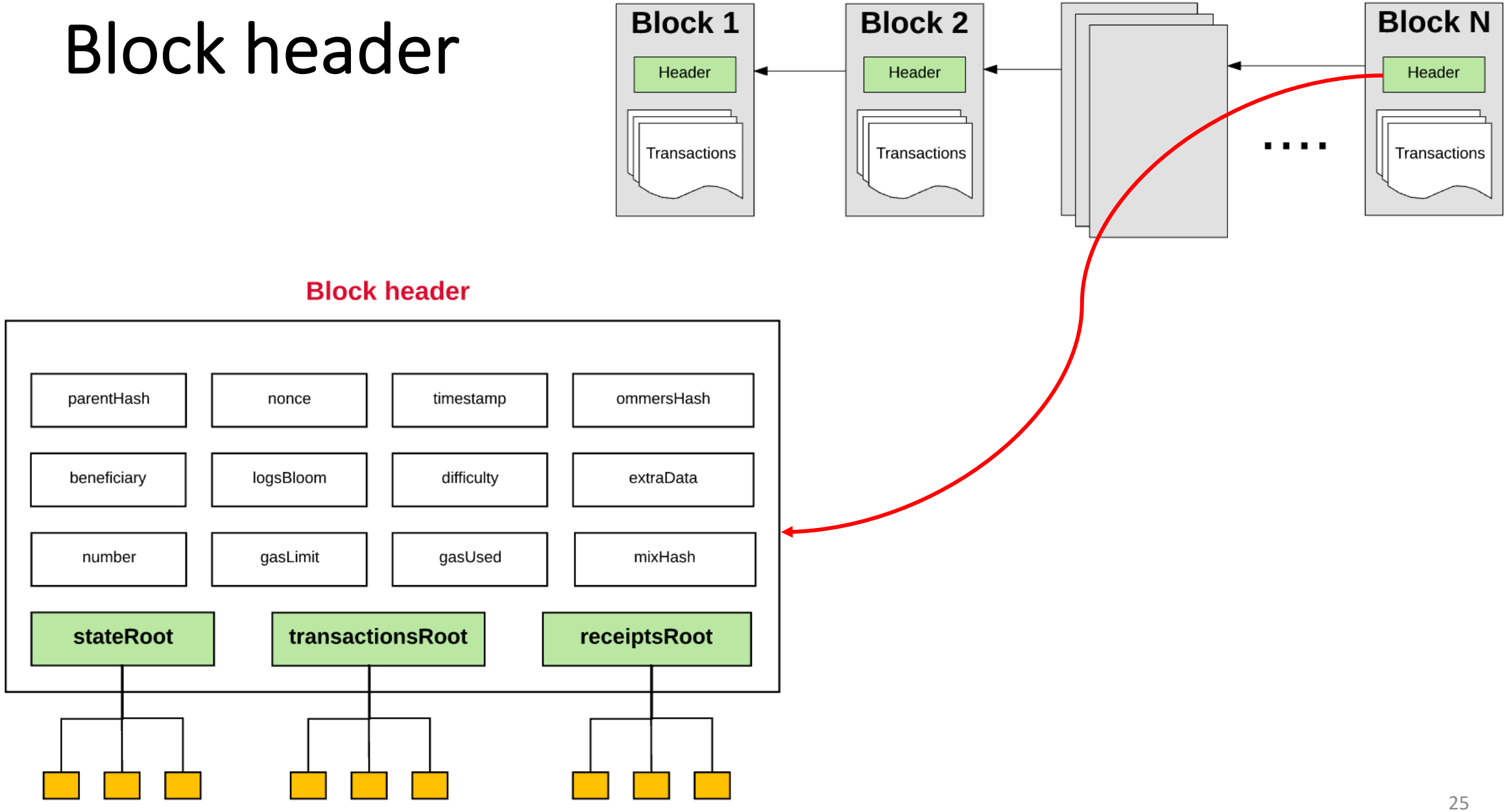
```
nonce: web3.toHex(9),
gasPrice: web3.toHex(20000000000),
gasLimit: web3.toHex(21000),
to: '0x3535353535353535353535353535353535353535',
value: web3.toHex(1000000000000000000),
data: '',
v:  '0x25'
r:  '0x28ef61340bd939bc2195fe537567866003e1a15d3c71ff63e1590620aa636276'
s:  '0x67cbe9d8997f761aecb703304b3800ccf555c9f3dc64214b297fb1966a3b6d83'
                              signature
```

Source: https://medium.com/@codetractio/walkthrough-of-an-ethereum-improvement-proposal-eip-6fda3966d171

Global state



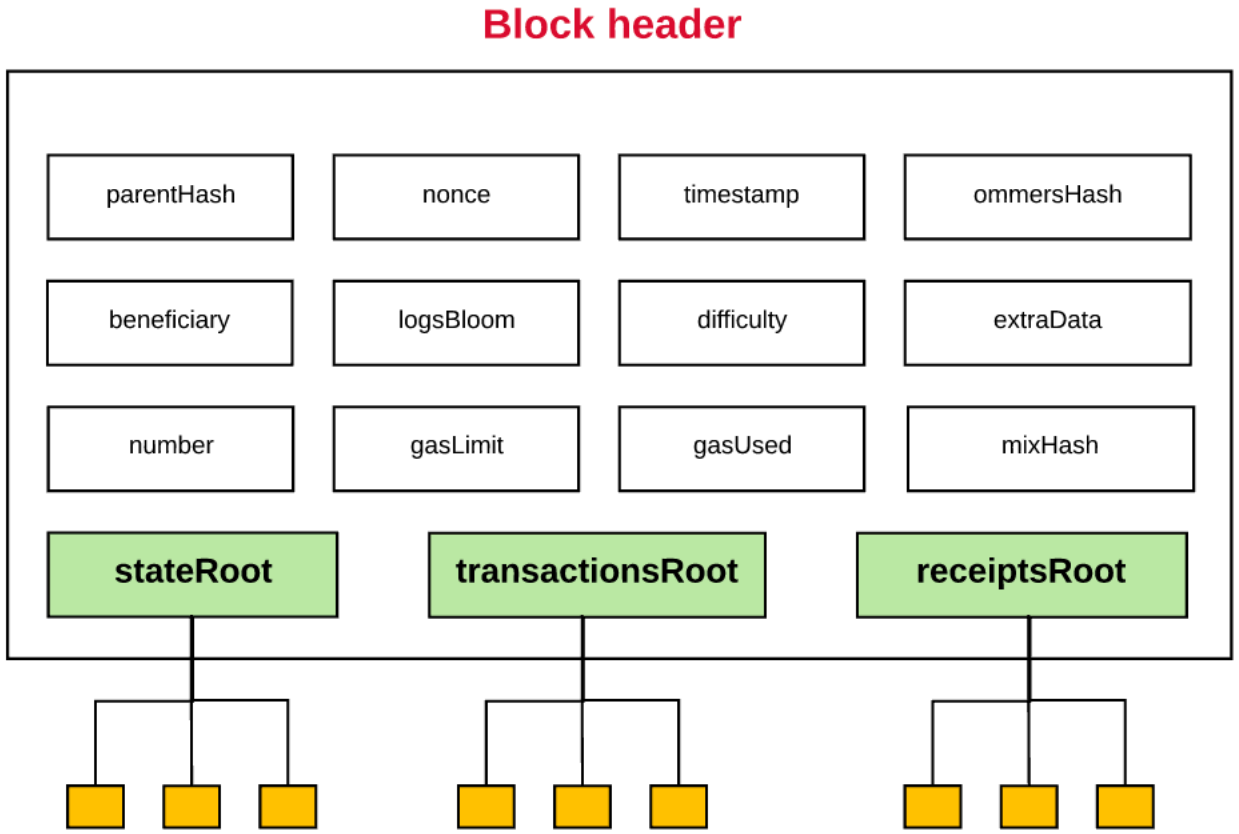
Block header



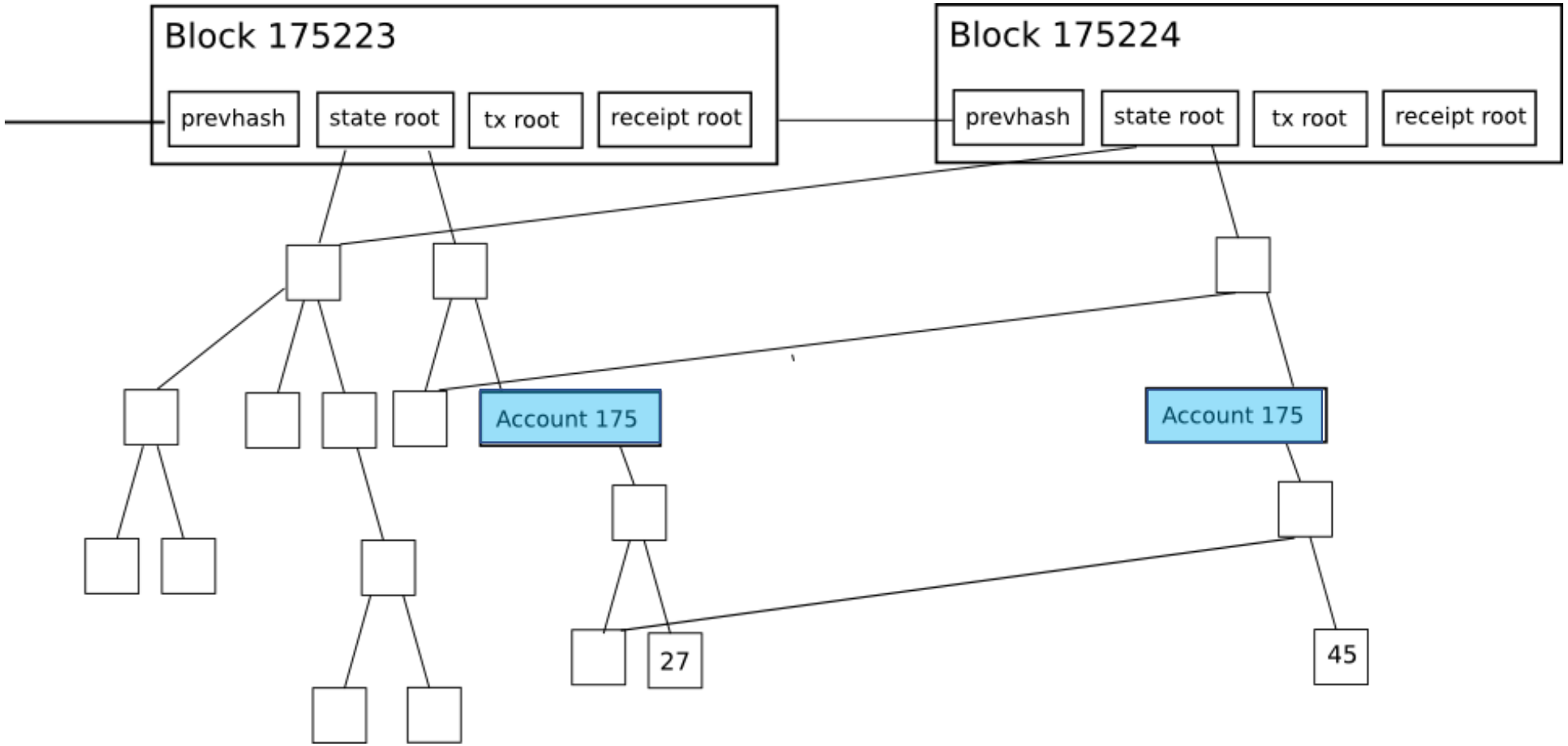
Block header

- state (**stateRoot**)
- transactions (**transactionsRoot**)
- receipts (**receiptsRoot**)

- Merkle Patricia tries

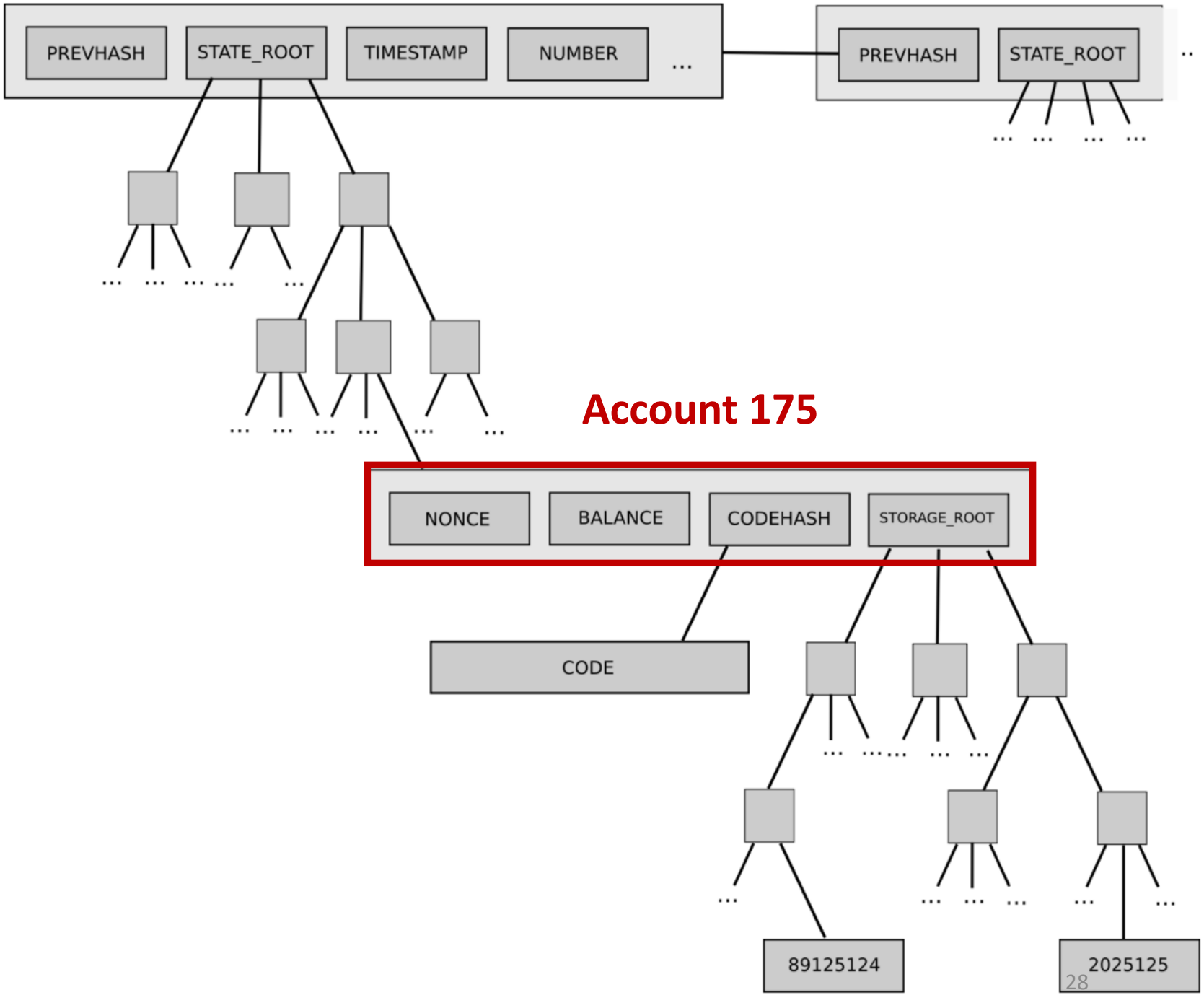


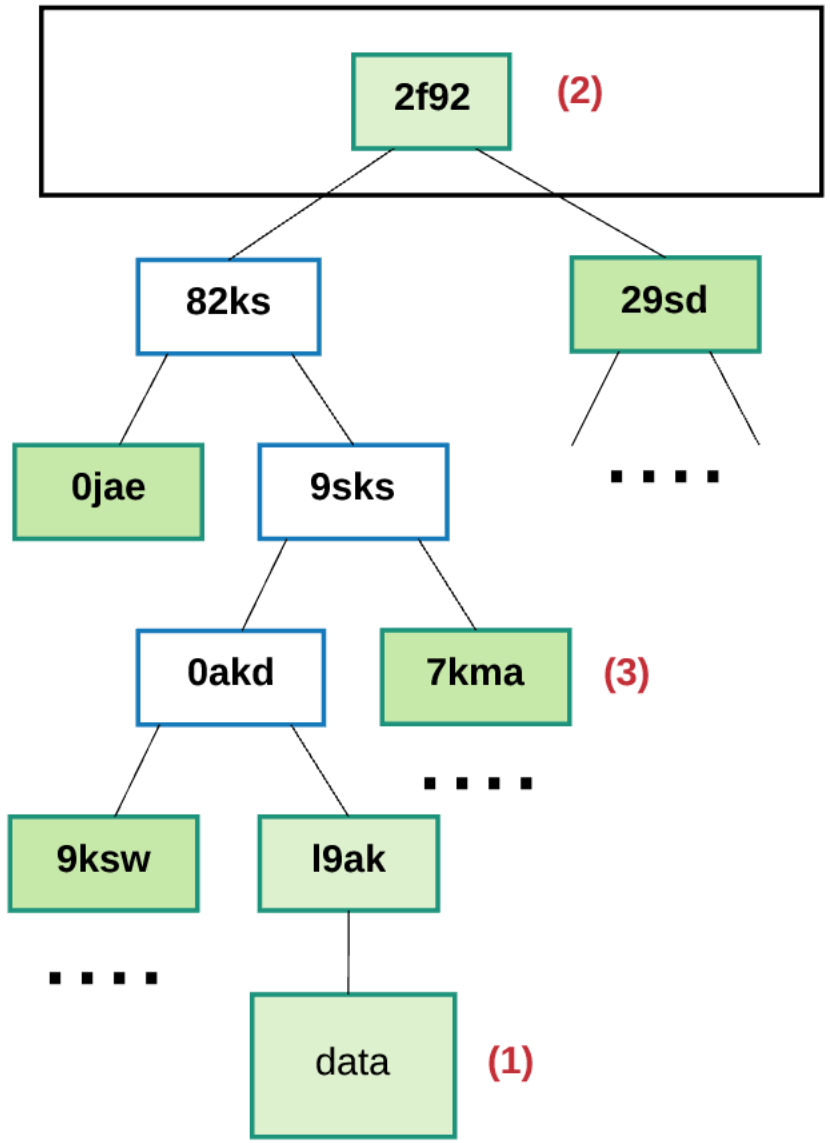
Global state

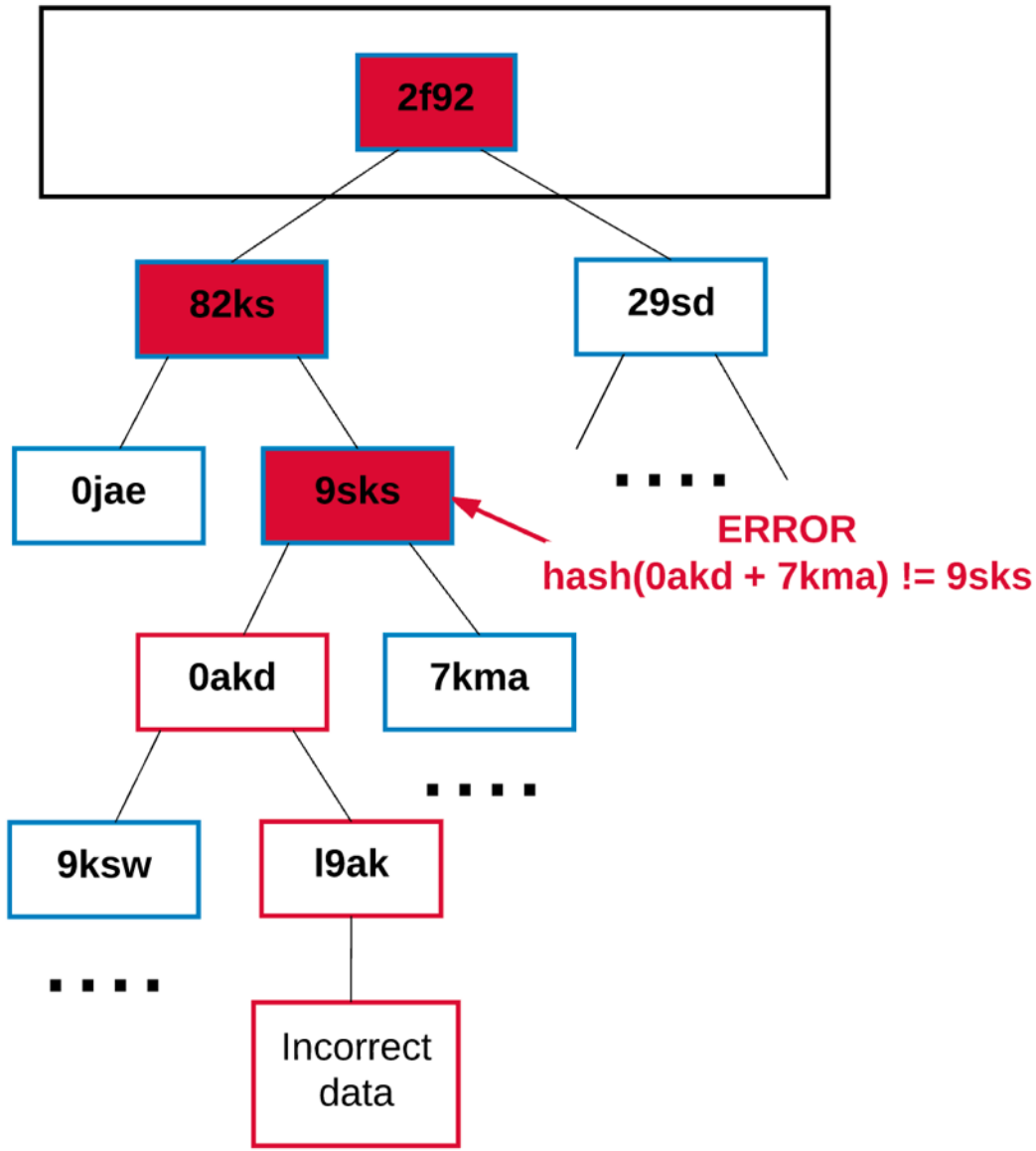


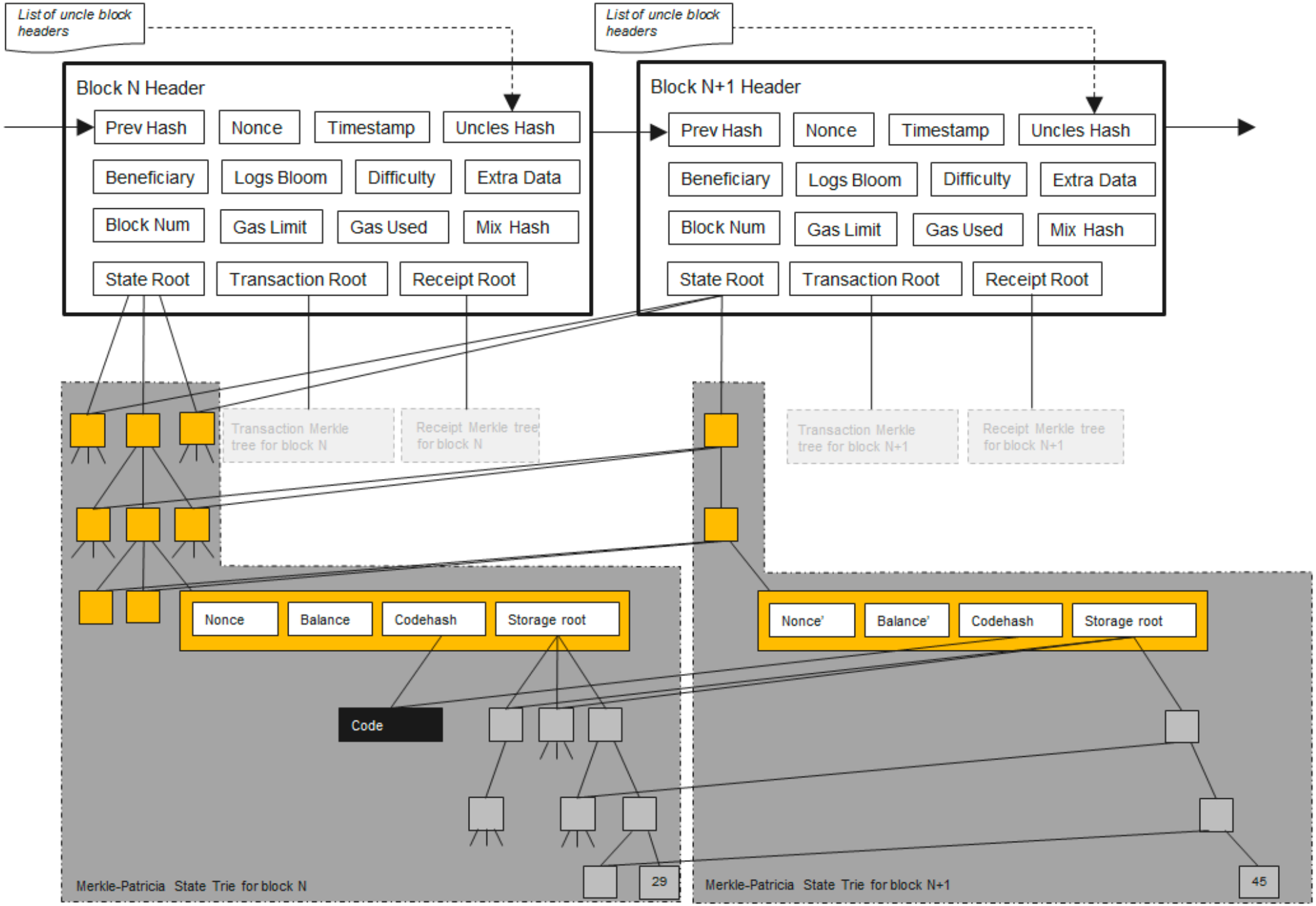
Account state

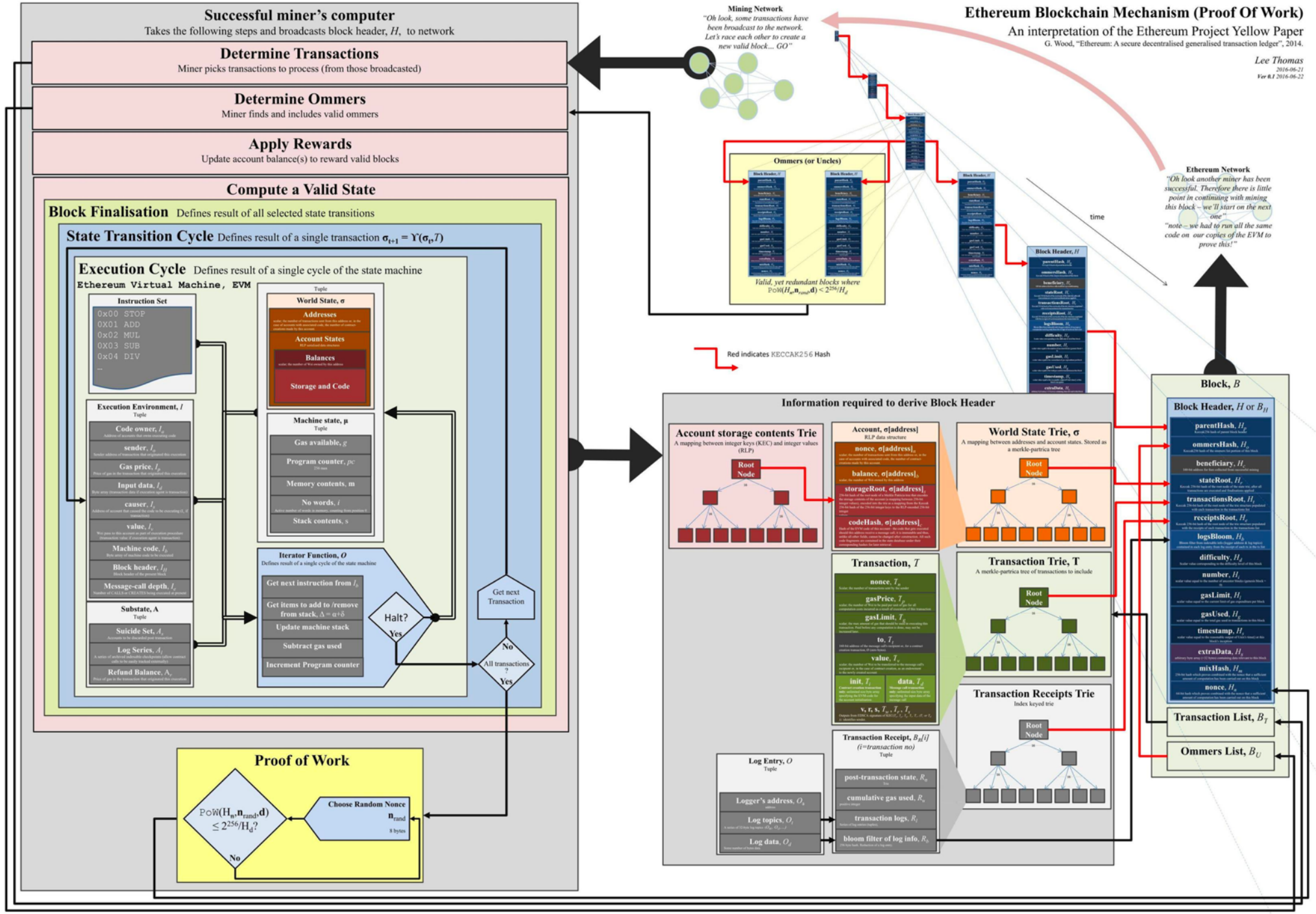
- nonce
- balance
- codeHash
- storageRoot




2f92
(2)
82ks
29sd
0jae
9sks
0akd
7kma
(3)
9ksw
l9ak
data
(1)


2f92
82ks
29sd
0jae
9sks
ERROR
hash(0akd + 7kma) != 9sks
0akd
7kma
9ksw
l9ak
Incorrect
data


List of uncle block headers
List of uncle block headers
Block N Header
Prev Hash
Nonce
Timestamp
Uncles Hash
Beneficiary
Logs Bloom
Difficulty
Extra Data
Block Num
Gas Limit
Gas Used
Mix Hash
State Root
Transaction Root
Receipt Root
Block N+1 Header
Prev Hash
Nonce
Timestamp
Uncles Hash
Beneficiary
Logs Bloom
Difficulty
Extra Data
Block Num
Gas Limit
Gas Used
Mix Hash
State Root
Transaction Root
Receipt Root
Transaction Merkle tree for block N
Receipt Merkle tree for block N
Transaction Merkle tree for block N+1
Receipt Merkle tree for block N+1
Nonce
Balance
Codehash
Storage root
Code
Nonce'
Balance'
Codehash
Storage root
Merkle-Patricia State Trie for block N
29
Merkle-Patricia State Trie for block N+1
45

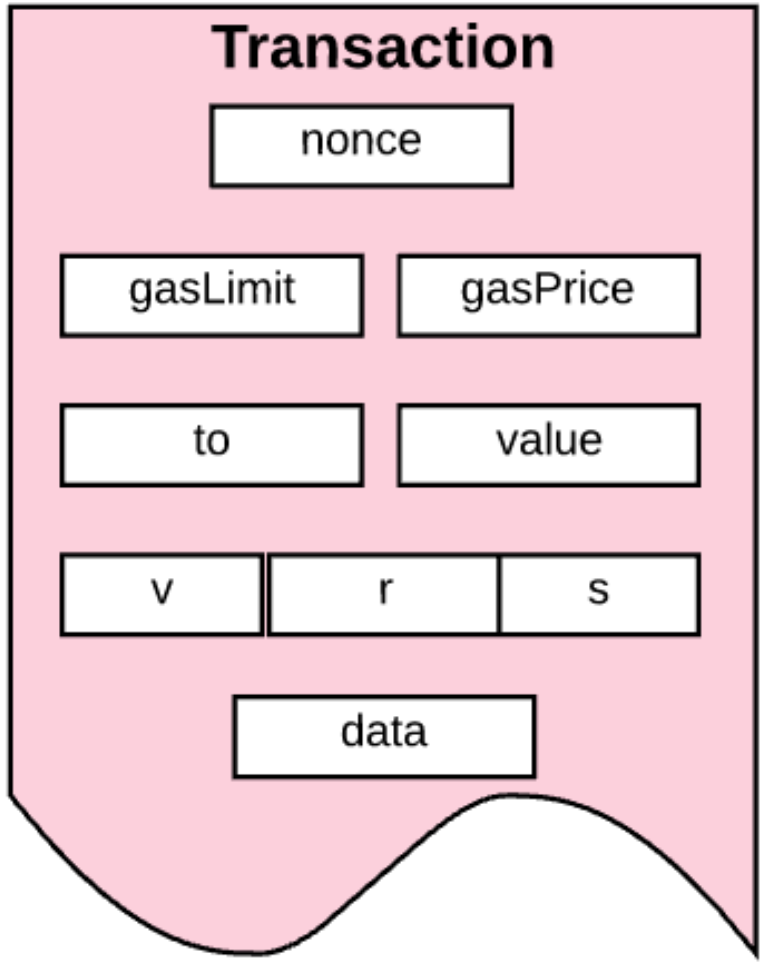

Successful miner's computer
Takes the following steps and broadcasts block header, H, to network
Determine Transactions
Miner picks transactions to process (from those broadcasted)
Determine Ommers
Miner finds and includes valid ommers
Apply Rewards
Update account balance(s) to reward valid blocks
Compute a Valid State
Block Finalisation Defines result of all selected state transitions
State Transition Cycle Defines result of a single transaction σt+1 = Υ(σt,T)
Execution Cycle Defines result of a single cycle of the state machine
Ethereum Virtual Machine, EVM
Instruction Set
0x00 STOP
0X01 ADD
0x02 MUL
0X03 SUB
0x04 DIV
Execution Environment, I
Tuple
Code owner, Ia
sender, Io
Gas price, Ip
Input data, Id
causer, Is
value, Iv
Machine code, Ib
Block header, IH
Message-call depth, Ie
Substate, A
Tuple
Suicide Set, As
Log Series, Aj
Refund Balance, Ar
Tuple
World State, σ
Addresses
Account States
Balances
Storage and Code
Machine state, μ
Tuple
Gas available, g
Program counter, pc
Memory contents, m
No words, i
Stack contents, s
Iterator Function, O
Defines result of a single cycle of the state machine
Get next instruction from Ib
Get items to add to /remove from stack, Δ = α+δ
Update machine stack
Subtract gas used
Increment Program counter
Halt?
Yes
Get next Transaction
No
All transactions ?
Yes
Proof of Work
PoW(Hn,nrand,d) ≤ 2256/Hd?
No
Choose Random Nonce
nrand
8 bytes
Mining Network
"Oh look, some transactions have been broadcast to the network. Let's race each other to create a new valid block… GO"
Ethereum Blockchain Mechanism (Proof Of Work)
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
2016-06-21
Ver 0.1 2016-06-22
Ommers (or Uncles)
Valid, yet redundant blocks where PoW(Hn,nrand,d) < 2256/Hd
time
Ethereum Network
"Oh look another miner has been successful. Therefore there is little point in continuing with mining this block – we'll start on the next one"
"note – we had to run all the same code on our copies of the EVM to prove this!"
Red indicates KECCAK256 Hash
Information required to derive Block Header
Account storage contents Trie
A mapping between integer keys (KEC) and integer values (RLP)
Root Node
Account, σ[address]
RLP data structure
nonce, σ[address]n
balance, σ[address]b
storageRoot, σ[address]s
codeHash, σ[address]c
World State Trie, σ
A mapping between addresses and account states. Stored as a merkle-partrica tree
Root Node
Transaction, T
nonce, Tn
gasPrice, Tp
gasLimit, Tg
to, Tt
value, Tv
init, Ti
data, Td
v, r, s, Tw, Tr, Ts
Transaction Trie, T
A merkle-partrica tree of transactions to include
Root Node
Transaction Receipts Trie
Index keyed trie
Root Node
Transaction Receipt, BR[i]
(i=transaction no)
Tuple
post-transaction state, Rσ
cumulative gas used, Ru
transaction logs, Rl
bloom filter of log info, Rb
Log Entry, O
Tuple
Logger's address, Oa
Log topics, Ot
Log data, Od
Block, B
Block Header, H or BH
parentHash, HP
ommersHash, HO
beneficiary, HC
stateRoot, HR
transactionsRoot, HT
receiptsRoot, HE
logsBloom, HB
difficulty, HD
number, Hi
gasLimit, Hl
gasUsed, Hg
timestamp, Hs
extraData, Hx
mixHash, Hm
nonce, Hn
Transaction List, BT
Ommers List, BU

Transactions

- 以太坊是一個基於交易的狀態機
- 在不同帳戶間發生交易，會讓狀態發生改變
- 交易是外部世界與以太坊內部狀態的橋樑

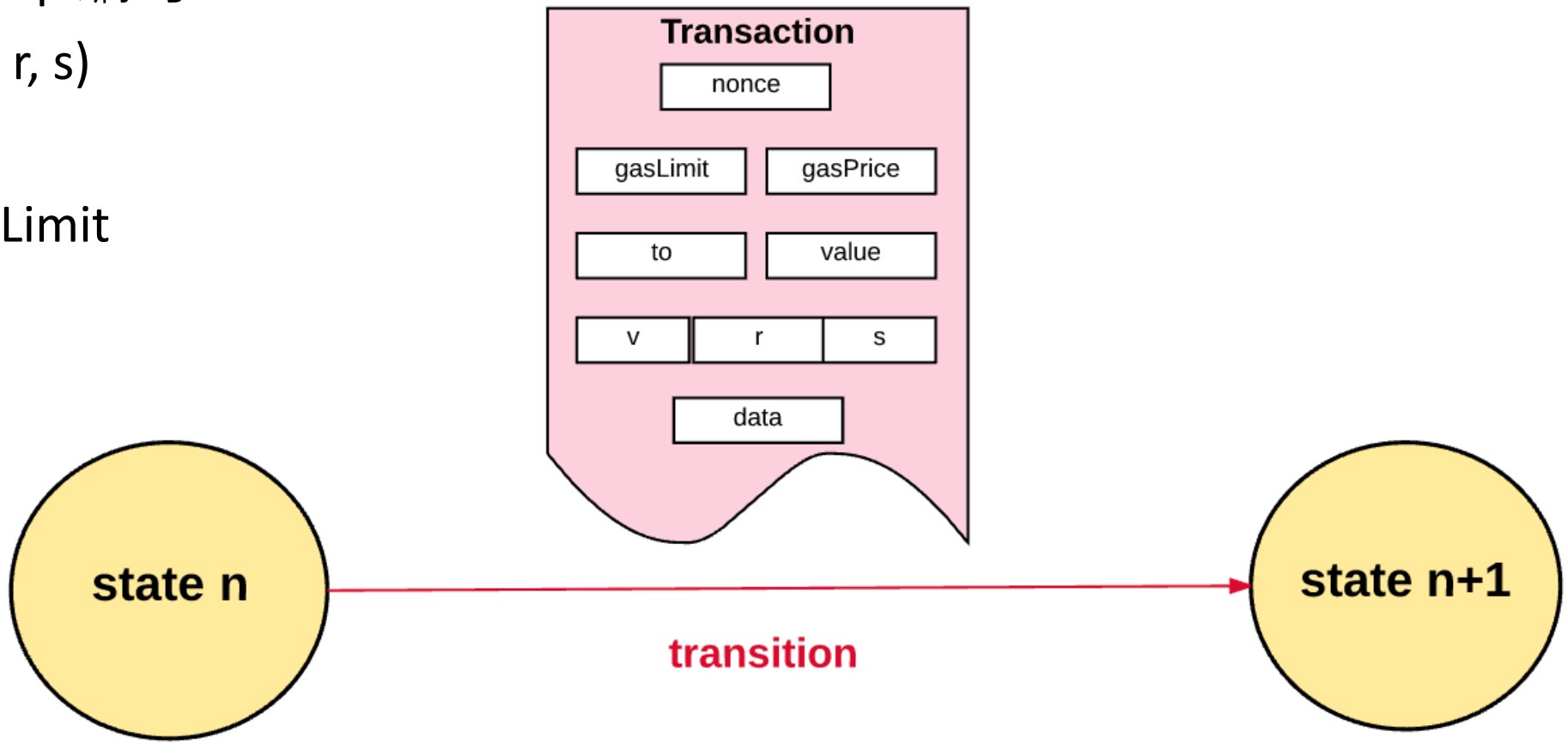
- 有兩種不同類型的交易
  - Message calls
  - Contract-creating transactions

- 一筆交易是由外部帳戶(EOA)簽名後送到區塊鏈



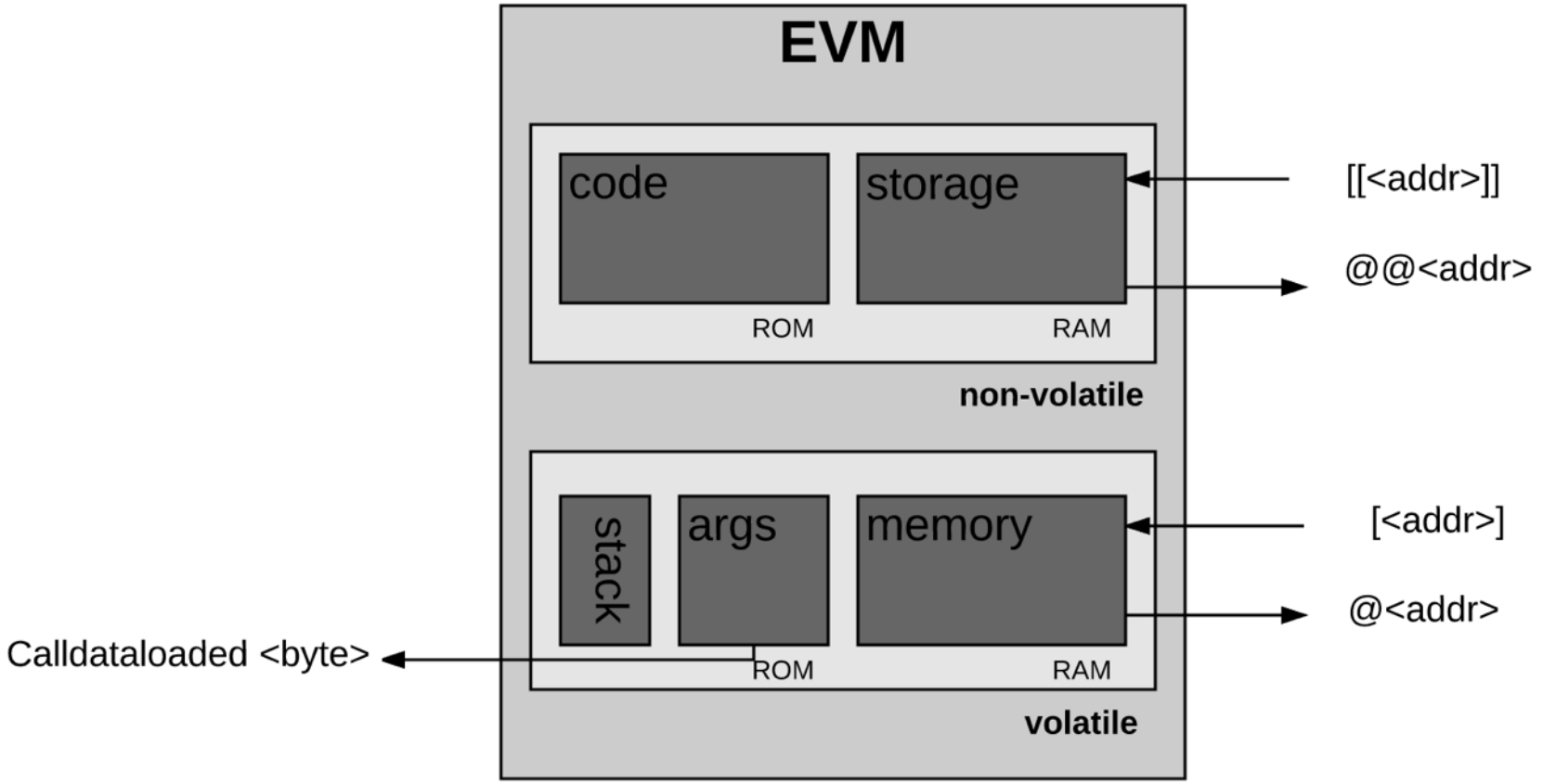
Transaction Execution

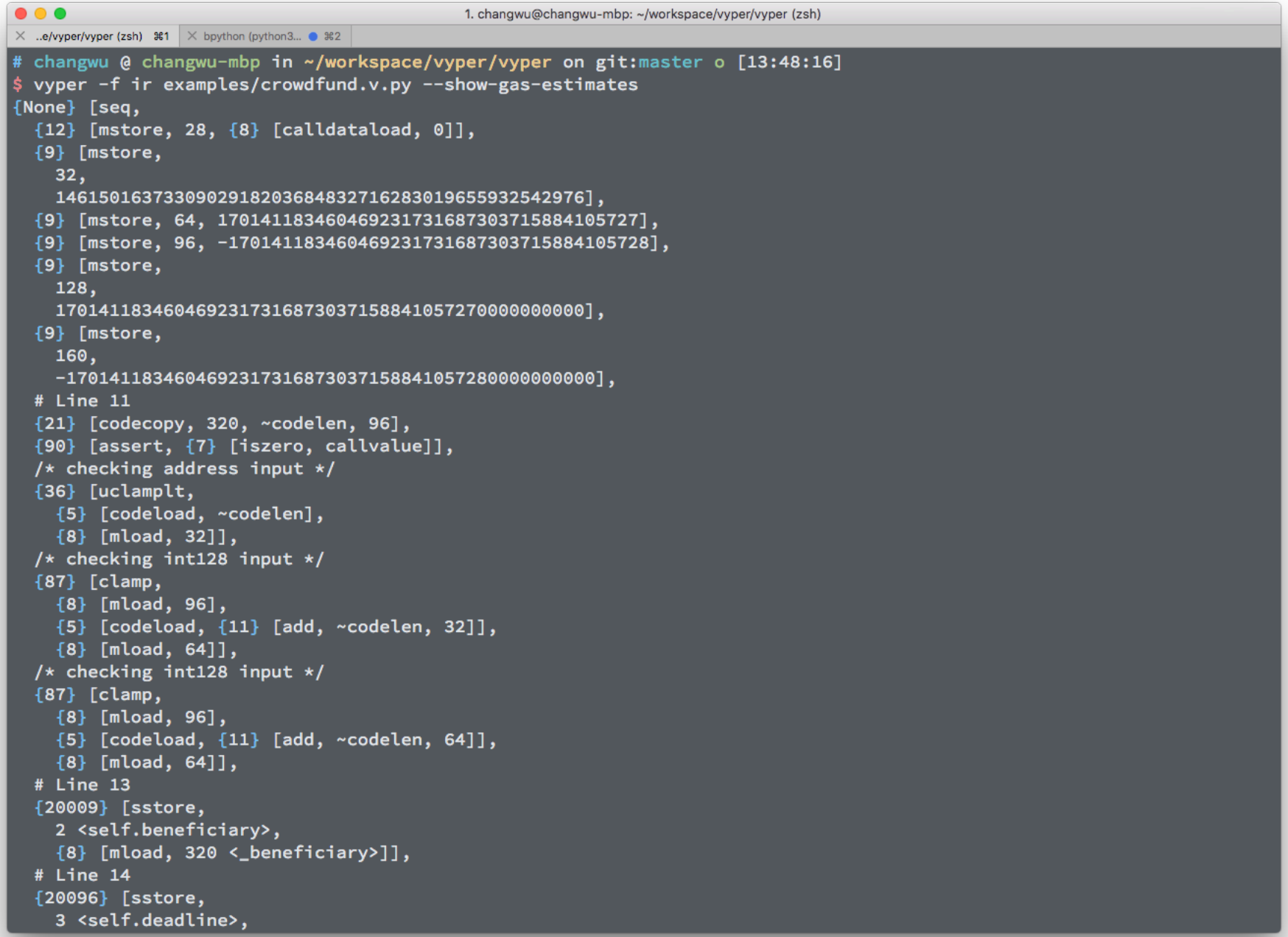
- 交易是正確的 rlp 編碼
- 有效的簽名 (v, r, s)
- 有效的 nonce
- gasUsed <= gasLimit



Execution model

- Contract
- Bytecode
- Transaction
- Message call

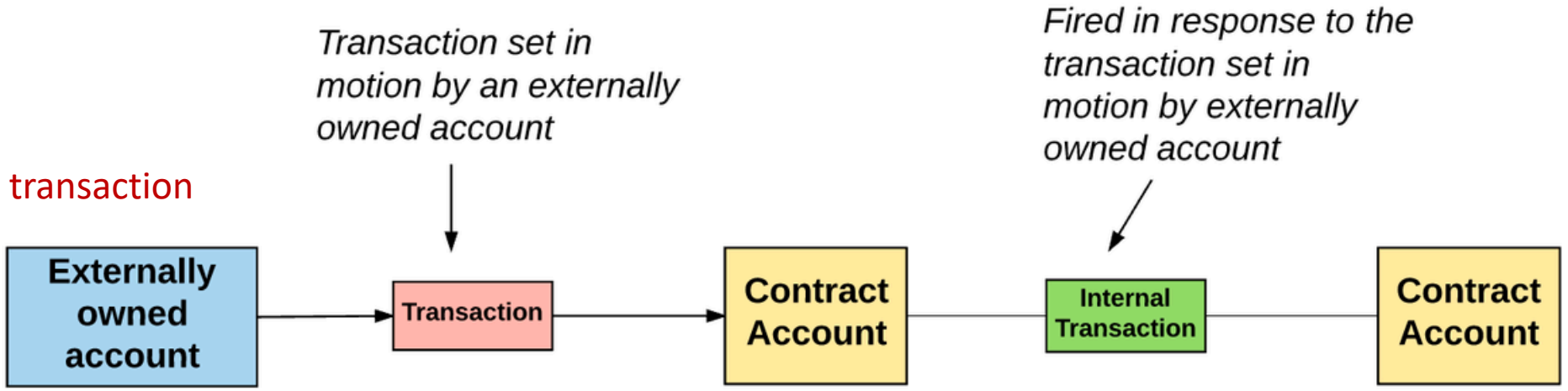



1. changwu@changwu-mbp: ~/workspace/vyper/vyper (zsh)
..e/vyper/vyper (zsh) ⌘1
bpython (python3... ⌘2
# changwu @ changwu-mbp in ~/workspace/vyper/vyper on git:master o [13:48:16]
$ vyper -f ir examples/crowdfund.v.py --show-gas-estimates
{None} [seq,
{12} [mstore, 28, {8} [calldataload, 0]],
{9} [mstore,
32,
1461501637330902918203684832716283019655932542976],
{9} [mstore, 64, 170141183460469231731687303715884105727],
{9} [mstore, 96, -170141183460469231731687303715884105728],
{9} [mstore,
128,
170141183460469231731687303715884105727000000000000],
{9} [mstore,
160,
-170141183460469231731687303715884105728000000000000],
Line 11
{21} [codecopy, 320, ~codelen, 96],
{90} [assert, {7} [iszero, callvalue]],
/* checking address input */
{36} [uclamplt,
{5} [codeload, ~codelen],
{8} [mload, 32]],
/* checking int128 input */
{87} [clamp,
{8} [mload, 96],
{5} [codeload, {11} [add, ~codelen, 32]],
{8} [mload, 64]],
/* checking int128 input */
{87} [clamp,
{8} [mload, 96],
{5} [codeload, {11} [add, ~codelen, 64]],
{8} [mload, 64]],
Line 13
{20009} [sstore,
2 <self.beneficiary>,
{8} [mload, 320 <_beneficiary>]],
Line 14
{20096} [sstore,
3 <self.deadline>,

訊息傳遞

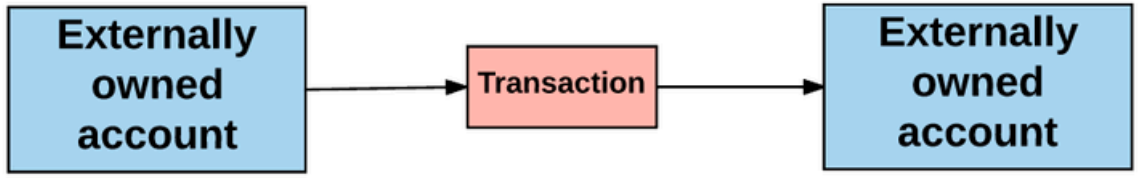
合約帳戶無法主動發起一筆交易，只有當外部帳戶或合約帳戶傳送一則訊息接收後，才會觸發合約帳戶執行程式碼或進行價值轉移

Contract transaction



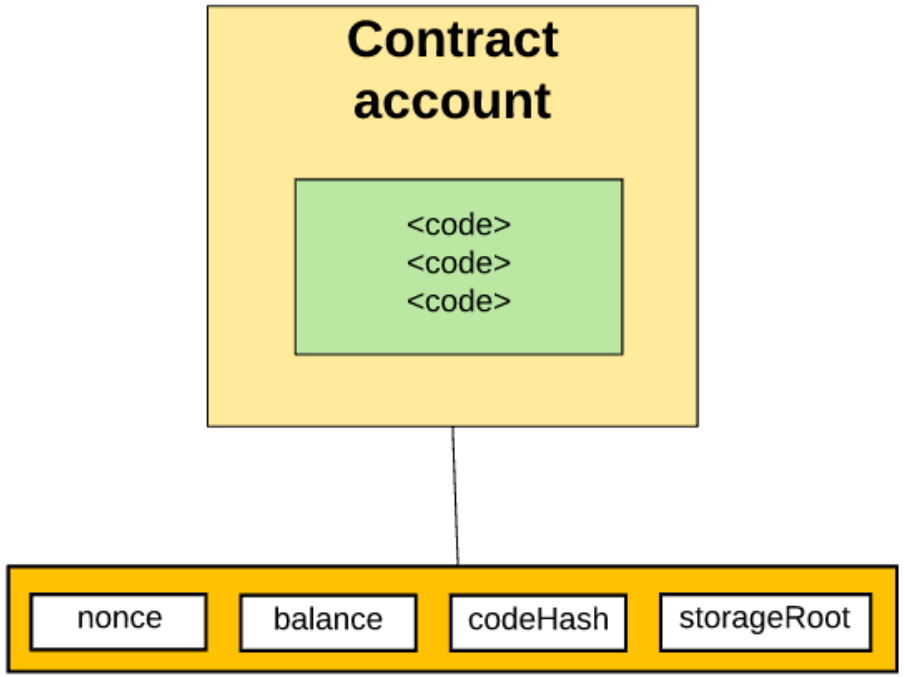
Standard transaction

兩個外部帳戶之間的訊息傳遞只是一個簡單的價值轉移

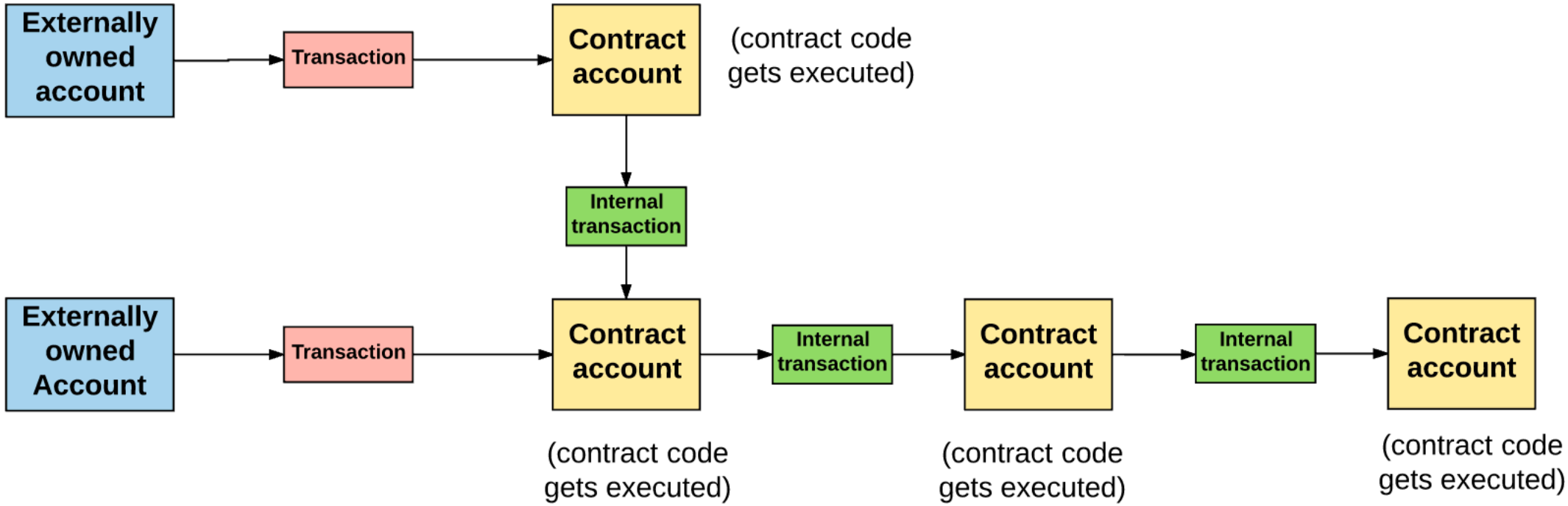


Contract creation

- Contract-creating transaction = Create a new contract account

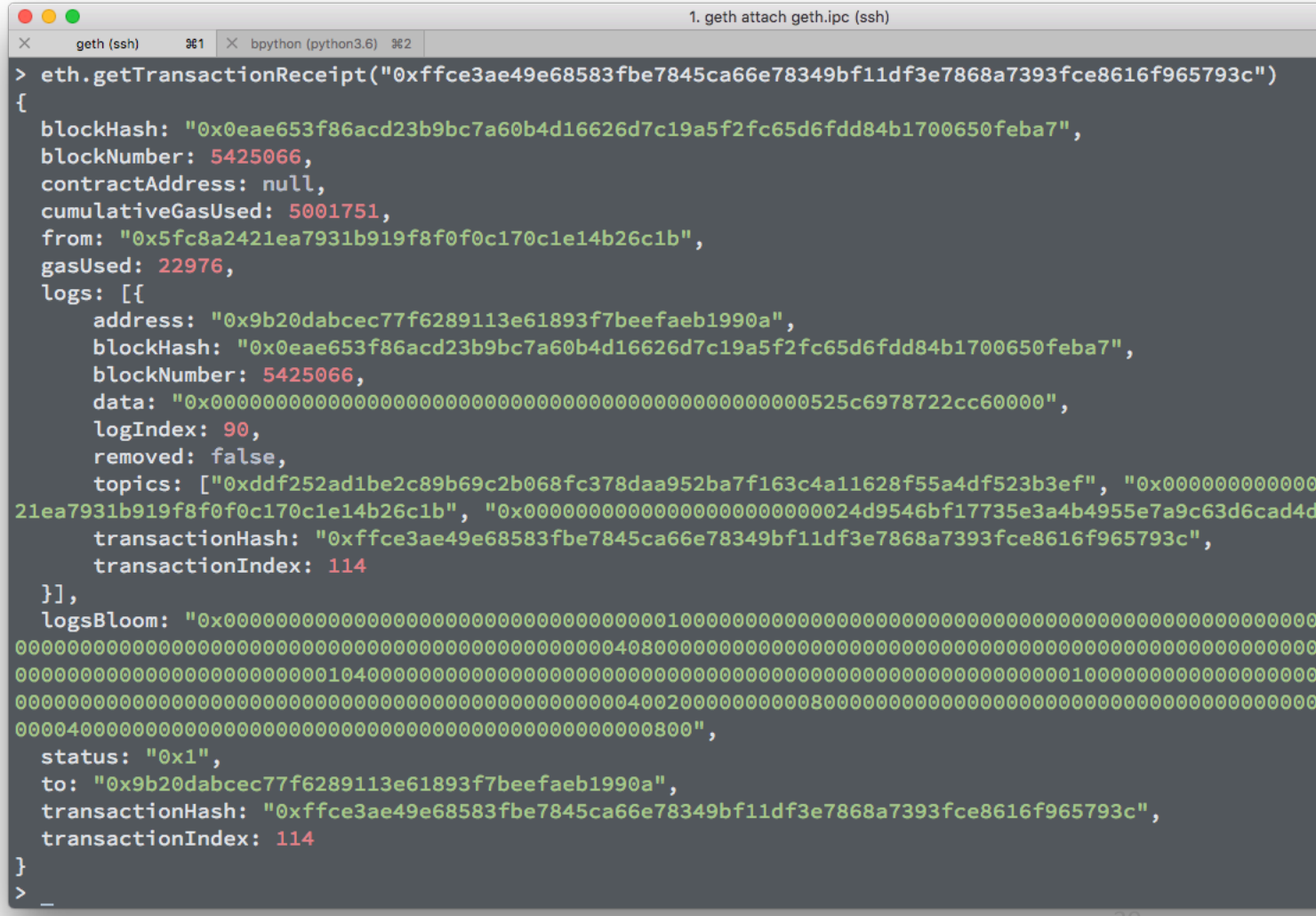


Internal transaction



Transaction receipt

- 交易執行後的收據
  - 區塊 Hash
  - 區塊號碼
  - 使用多少 gas
  - 執行交易後的 Log 訊息
  - 交易 hash
  - 交易在區塊內的 index
  - From
  - To



Logs

- 用來追蹤交易的狀態與訊息

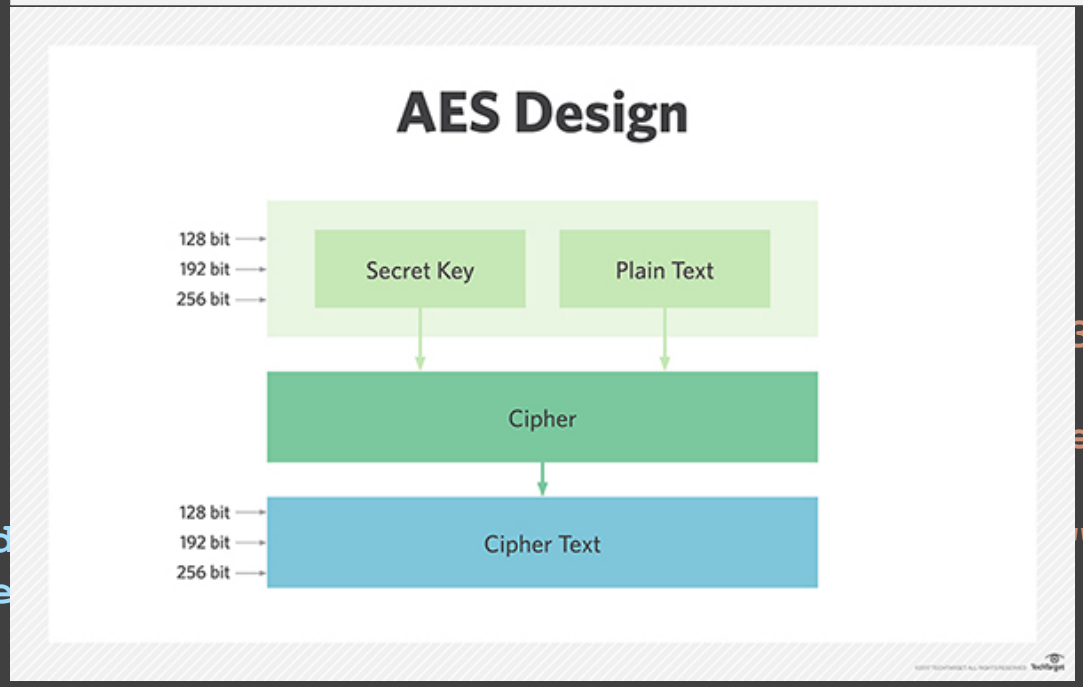
- Log 包含
  - 合約地址
  - 合約內部事件(event) 訂閱的主題 (topic)
  - 與 event 相關的 data

Keystore

keystore.json

cipher: AES 加密演算法
cipherparams: 是 AES 演算法需要的參數
ciphertext: ETH private key 加密後的密文

```
{
  "crypto" : {
      "cipher" : "aes-128-ctr",
      "cipherparams" : {
          "iv" : "83dbcc02d8ccb40e466191a123791e0e"
      },
      "ciphertext" : "d172bf743a674da9cdad04534d56926ef8358534d458fffccd4e6ad2fbde479c",
                                                                    3fe8984592145b591fc8fb5c6d43190334ba19"
                                                                    e95cfd7ff8faea1056c33131d846e3097"
  },
  "id                                                               ",
  "ve
}
```



keystore.json

AES(plaintext, enc_key) = ciphertext

kdf: 密鑰推導函數

這邊可以採取的 kdf 有兩種

1. scrypt
2. pbkdf2


```
{
  "crypto" : {
      "cipher" : "aes-128-ctr",
      "cipherparams" : {
          "iv" : "83dbcc02d8ccb40e466191a123791e0e"
      },
      "ciphertext" : "d172bf743a674da9cdad04534d56926ef8358534d458fffccd4e6ad2fbde479c",
      "kdf" : "scrypt",
      "kdfparams" : {
          "dklen" : 32,
          "n" : 262144,
          "r" : 1,
          "p" : 8,
          "salt" : "ab0c7876052600dd703518d6fc3fe8984592145b591fc8fb5c6d43190334ba19"
      },
      "mac" : "2103ac29920d71da29f15d75b4a16dbe95cfd7ff8faea1056c33131d846e3097"
  },
  "id" : "3198bc9c-6672-5ab3-d995-4942343ae5b6",
  "version" : 3
}
```

keystore.json

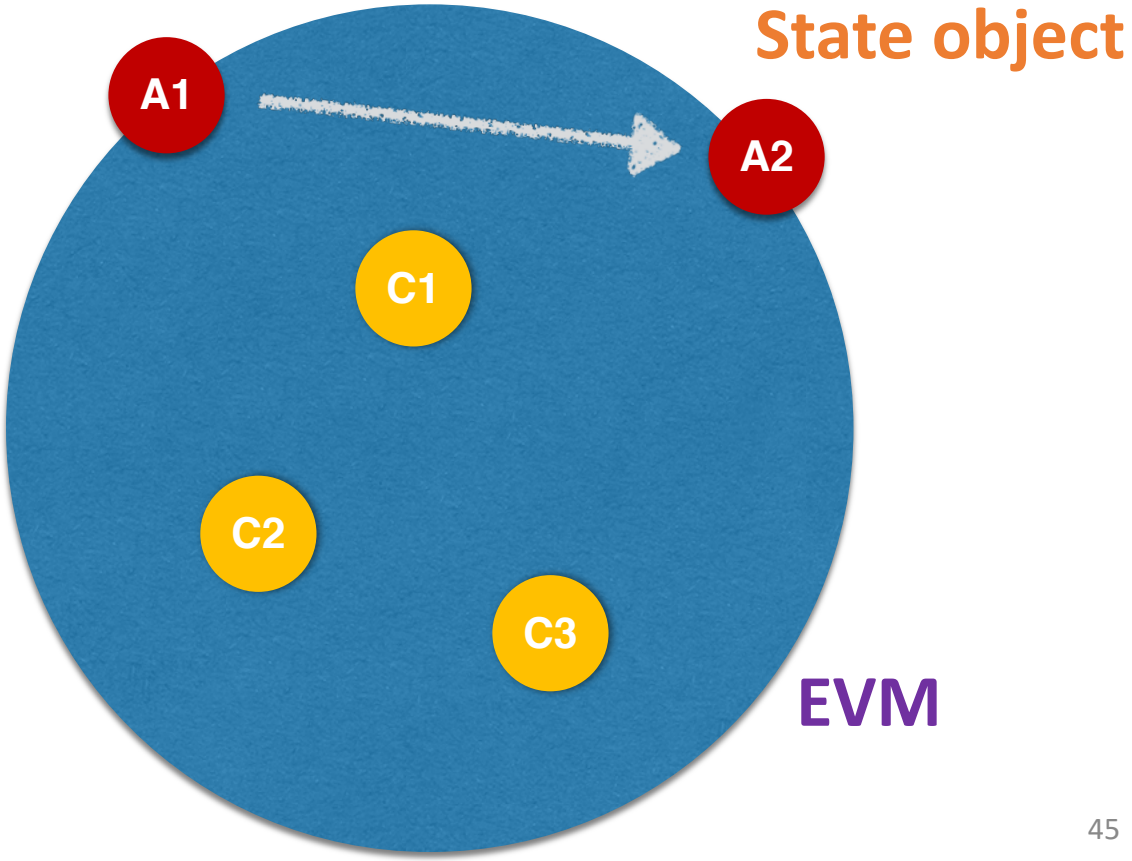
為了檢查使用者的 password 正確與否, 使用 mac 來驗證一致性

mac: sha3(derivedkey[16:32] + cyphertext)


```
{
  "crypto" : {
      "cipher" : "aes-128-ctr",
      "cipherparams" : {
          "iv" : "83dbcc02d8ccb40e466191a123791e0e"
      },
      "ciphertext" : "d172bf743a674da9cdad04534d56926ef8358534d458fffccd4e6ad2fbde479c",
      "kdf" : "scrypt",
      "kdfparams" : {
          "dklen" : 32,
          "n" : 262144,
          "r" : 1,
          "p" : 8,
          "salt" : "ab0c7876052600dd703518d6fc3fe8984592145b591fc8fb5c6d43190334ba19"
      },
      "mac" : "2103ac29920d71da29f15d75b4a16dbe95cfd7ff8faea1056c33131d846e3097"
  },
  "id" : "3198bc9c-6672-5ab3-d995-4942343ae5b6",
  "version" : 3
}
```

Ethereum

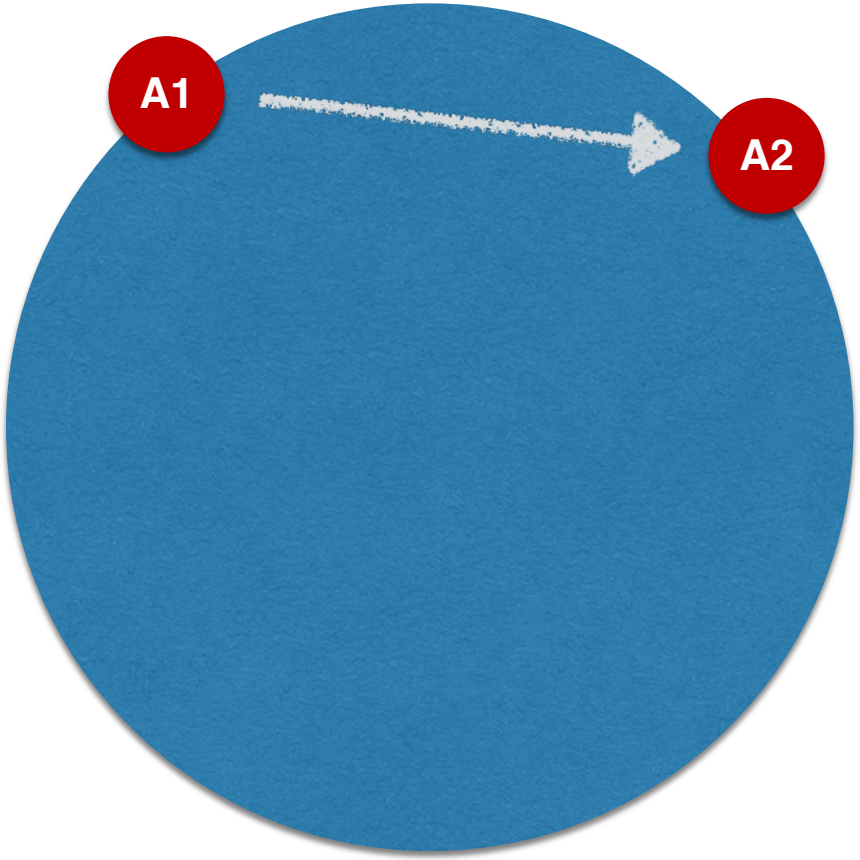
- 外部帳戶 (Externally owned account, EOA)
- 合約帳戶 (Contract account)
  - Nonce
  - Balance
  - codeHash
  - storageRoot



Ethereum

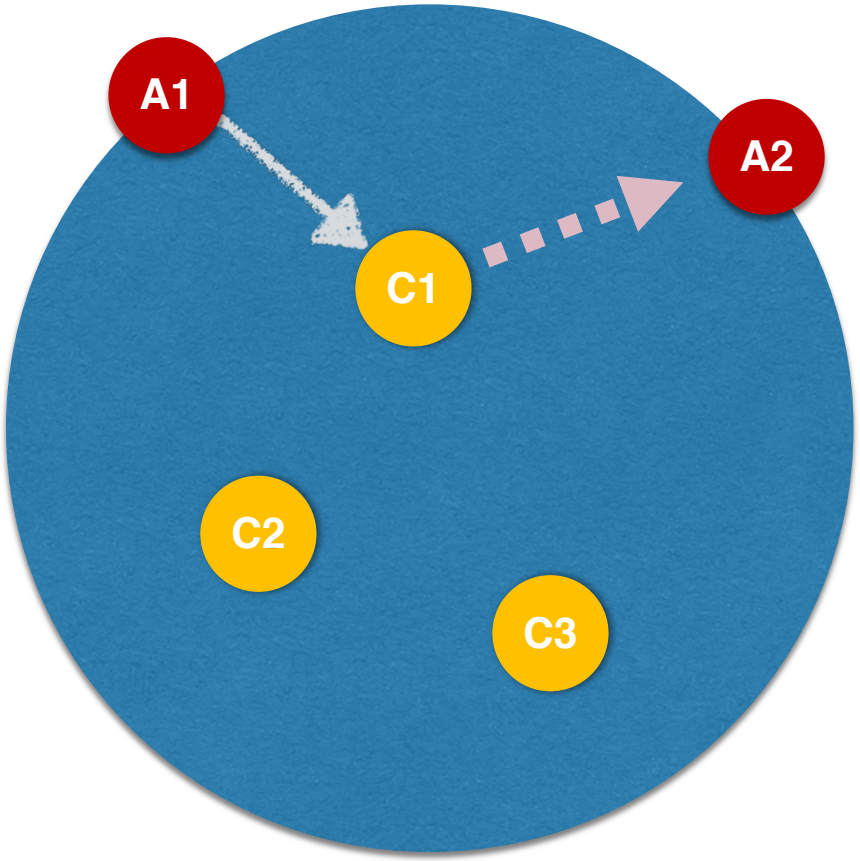
address: '0x556657e4691afe833b9c6978056706fbe095f4ad'
address: '0x556662d50c8789885fb5dc2263c332134fdf95c8'

- 外部帳戶 (Externally owned account, EOA)
- 帳戶之間轉帳
  - Nonce
  - Balance
  - codeHash
  - storageRoot



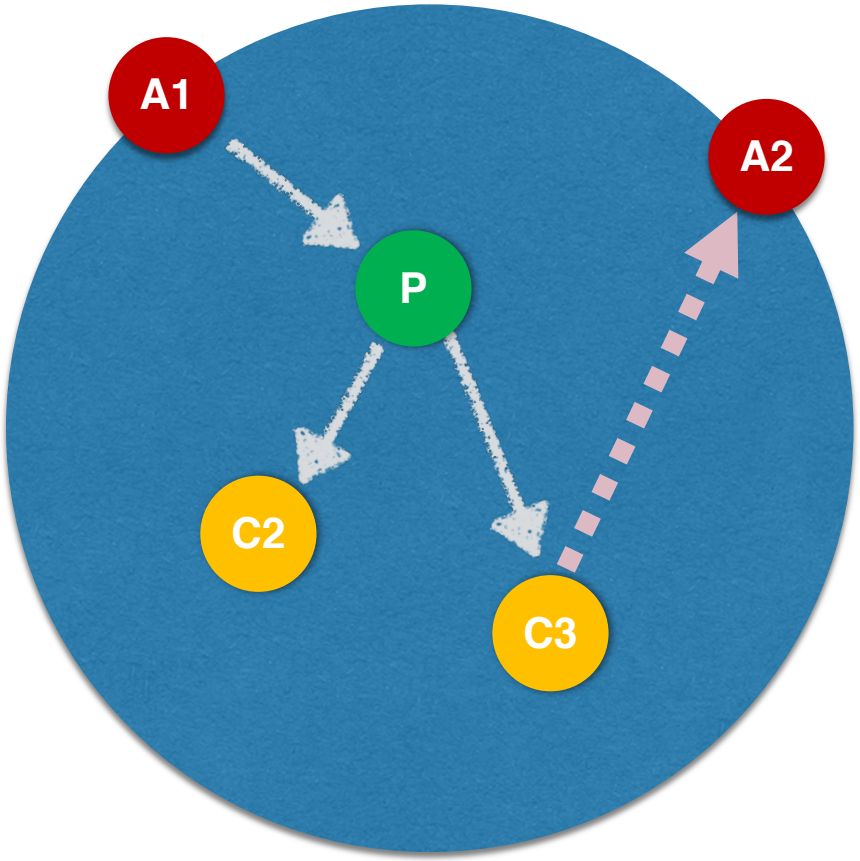
Ethereum

- 合約被呼叫後的反應
- 合約帳戶 (Contract account)
  - Nonce
  - Balance
  - codeHash
  - storageRoot



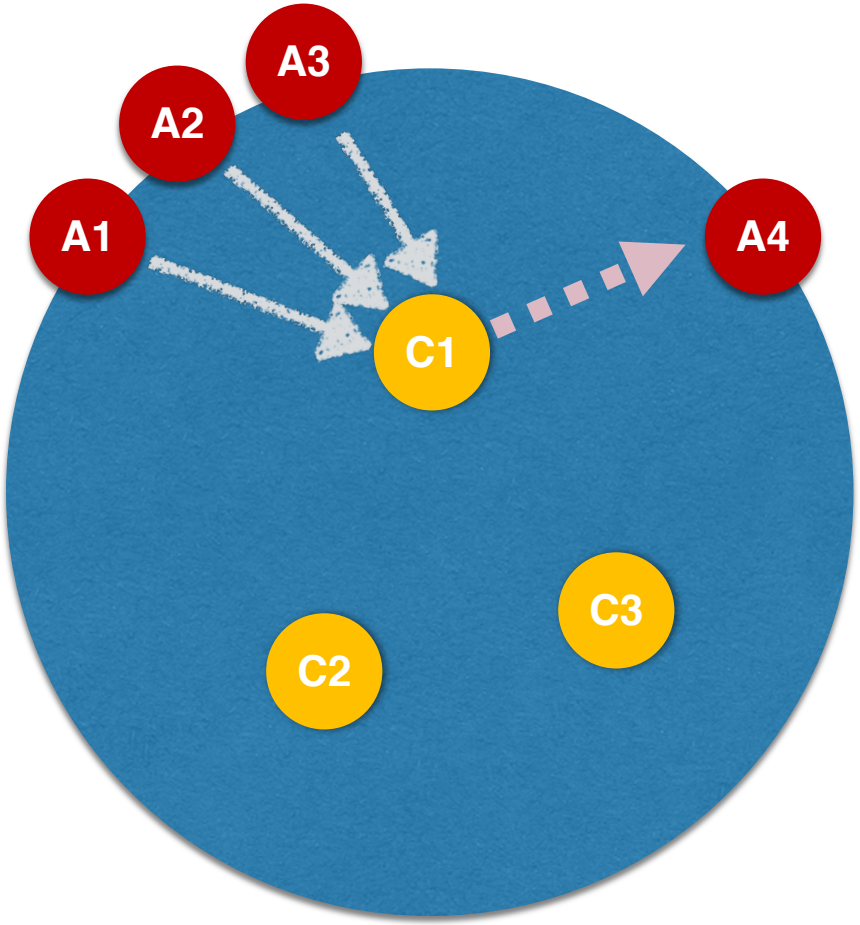
Ethereum

- 連鎖反應 (Proxy, Controller)
- 替換合約



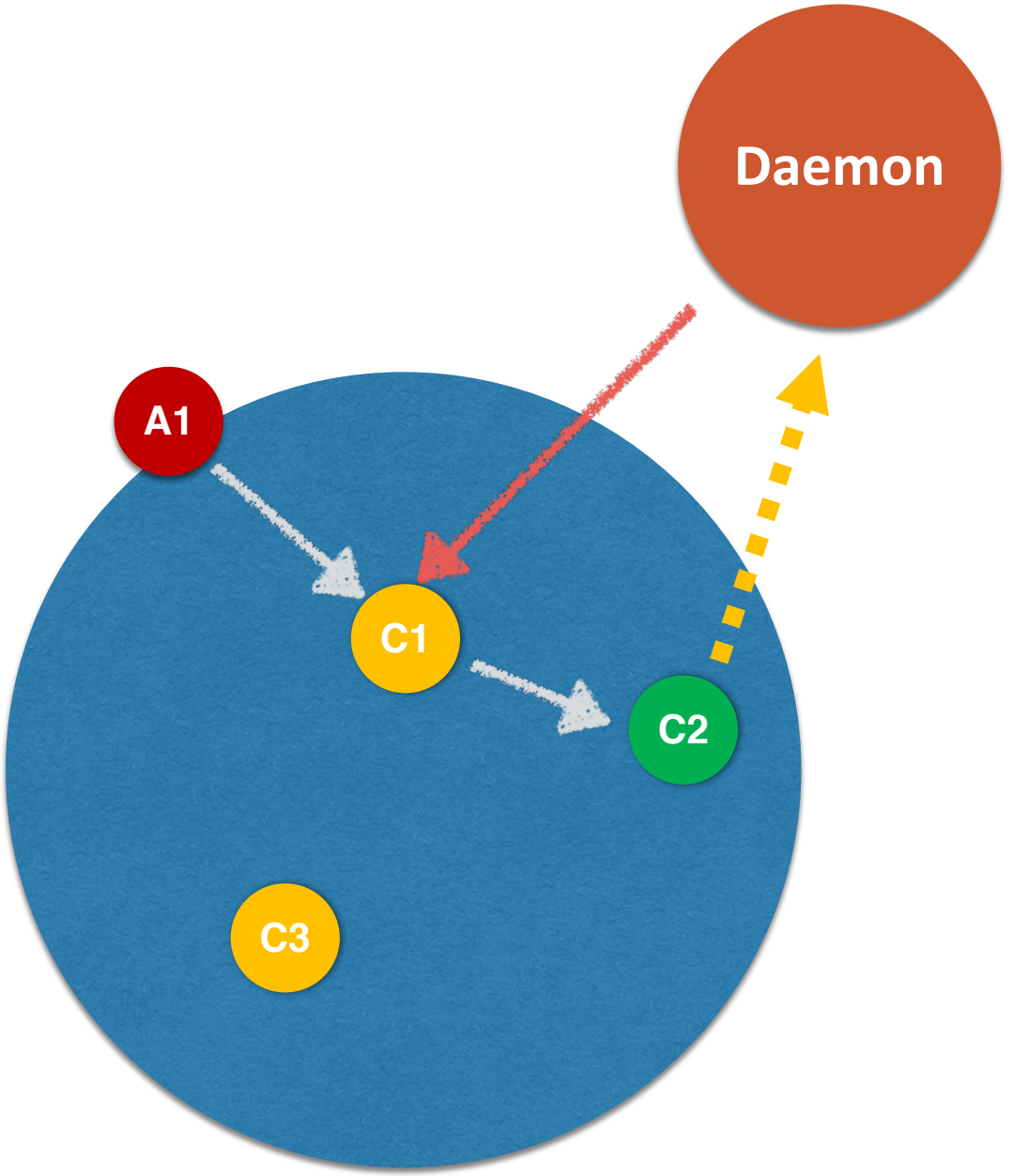
Ethereum

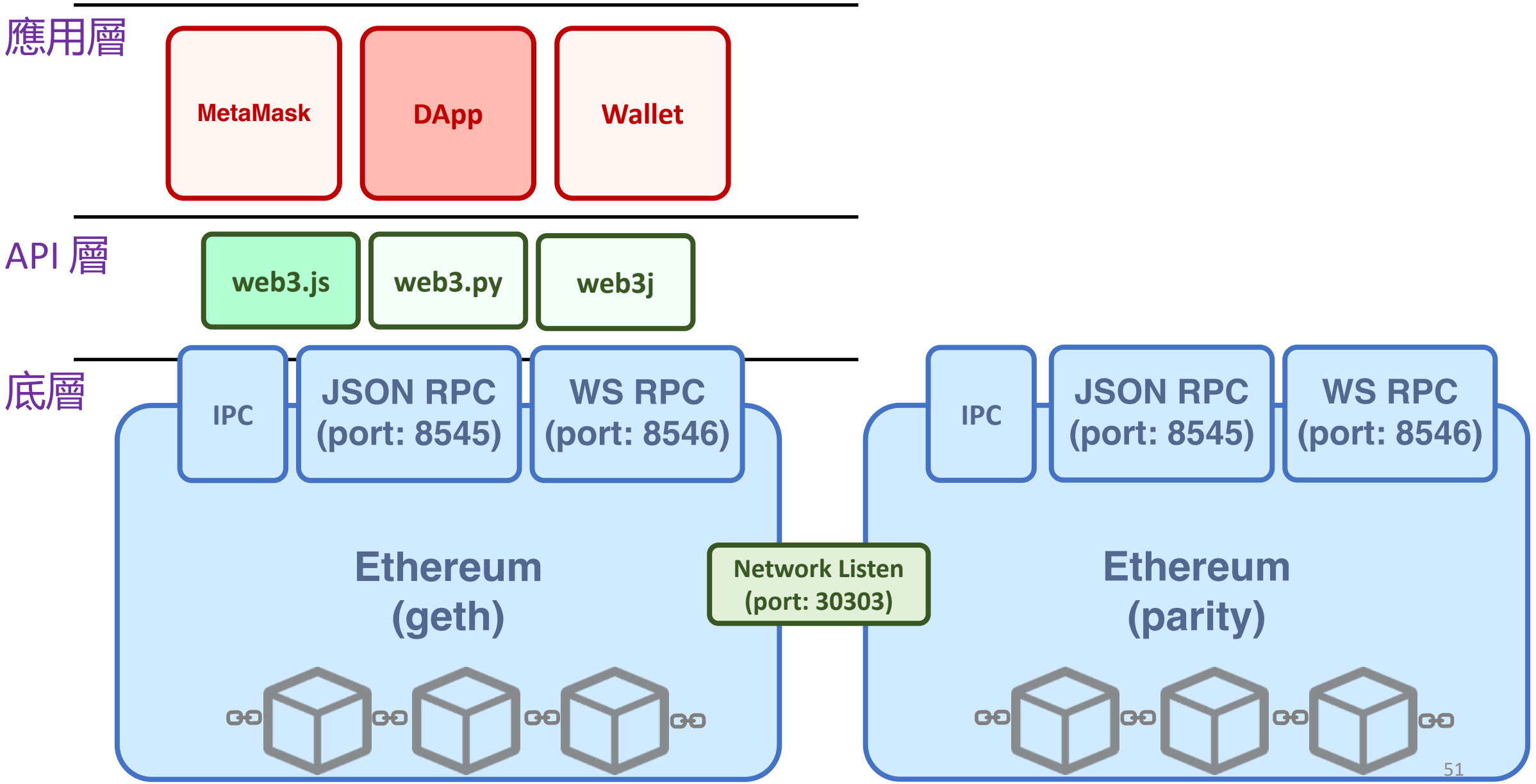
- 多簽 (multi-sig) 透過 Proxy 合約

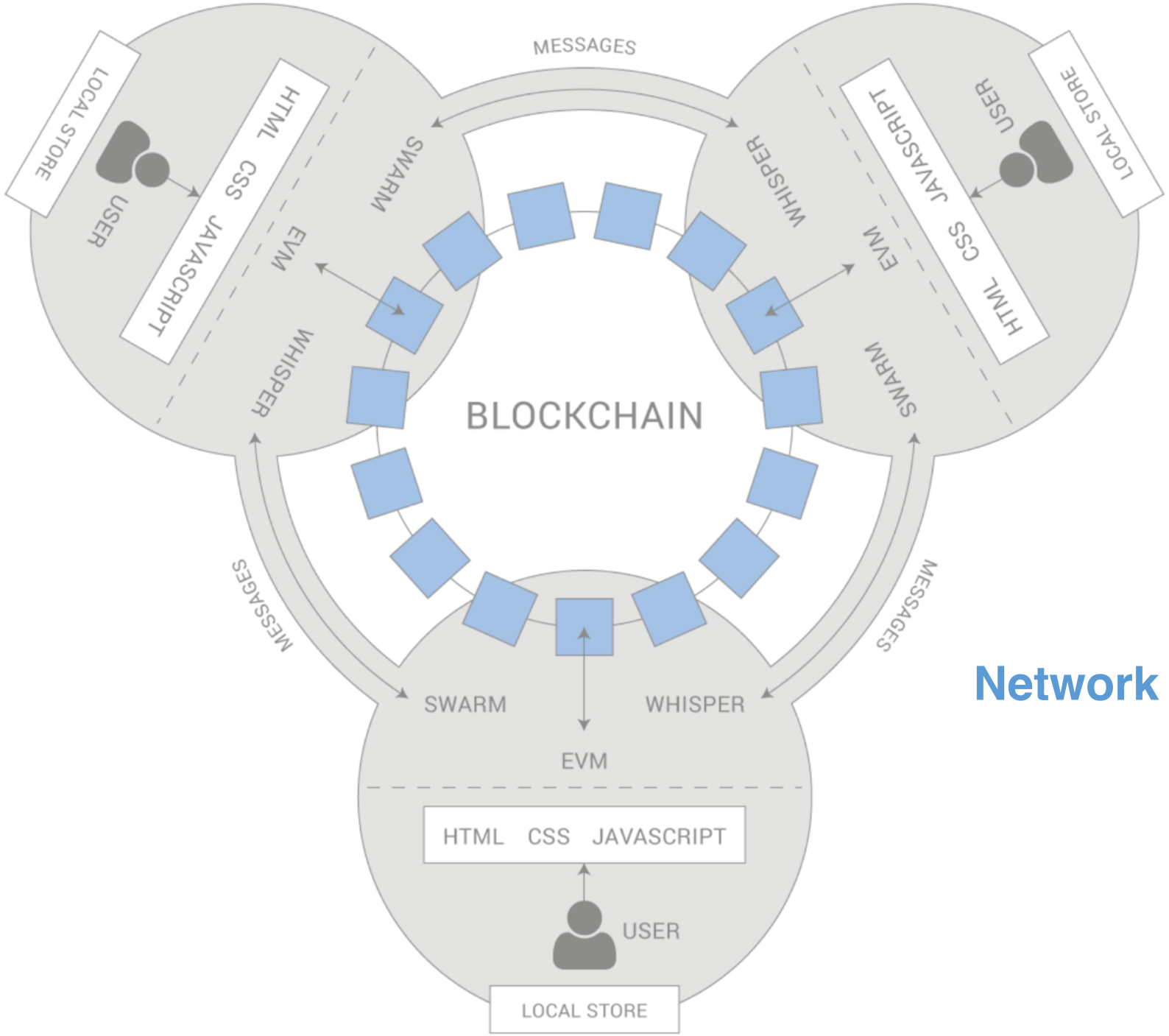


Ethereum

- Oracle
- Feed Data & Proof

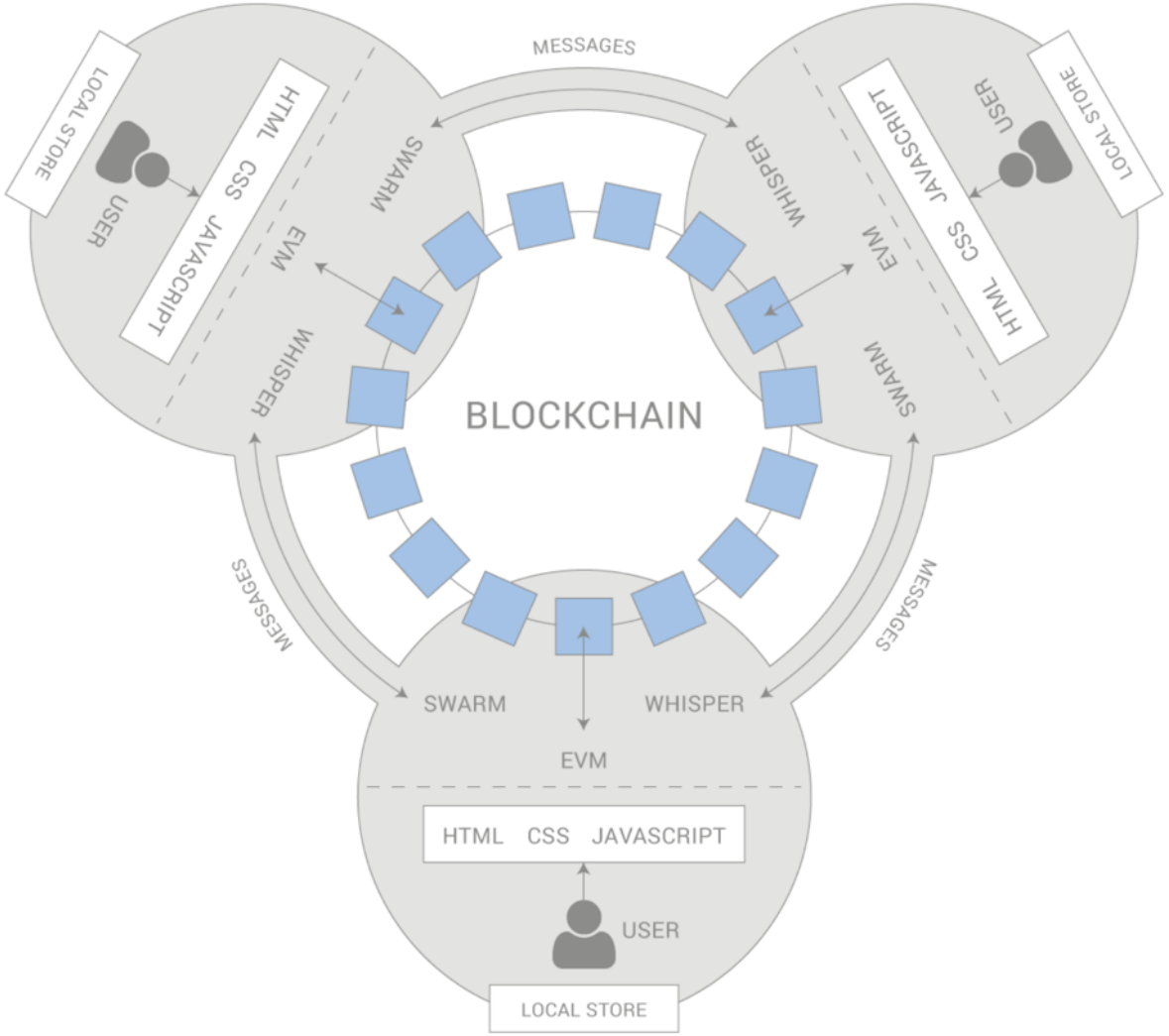



應用層
MetaMask
DApp
Wallet
API 層
web3.js
web3.py
web3j
底層
IPC
JSON RPC
(port: 8545)
WS RPC
(port: 8546)
Ethereum
(geth)
Network Listen
(port: 30303)
IPC
JSON RPC
(port: 8545)
WS RPC
(port: 8546)
Ethereum
(parity)

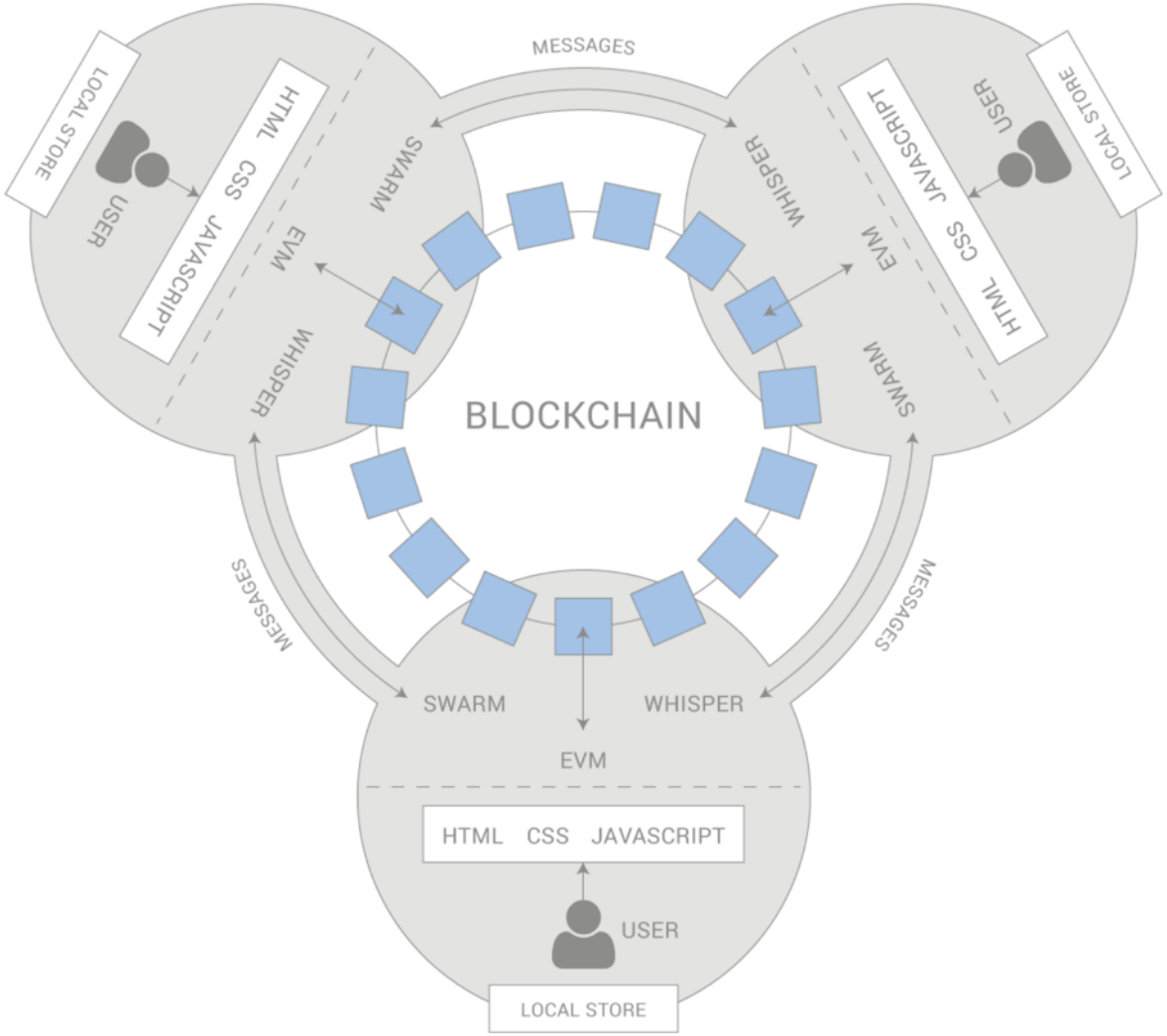

MESSAGES
LOCAL STORE
USER
HTML CSS JAVASCRIPT
EVM
SWARM
WHISPER
BLOCKCHAIN
WHISPER
SWARM
EVM
HTML CSS JAVASCRIPT
USER
LOCAL STORE
MESSAGES
MESSAGES
SWARM
WHISPER
EVM
HTML CSS JAVASCRIPT
USER
LOCAL STORE

Network ID

Mainnet

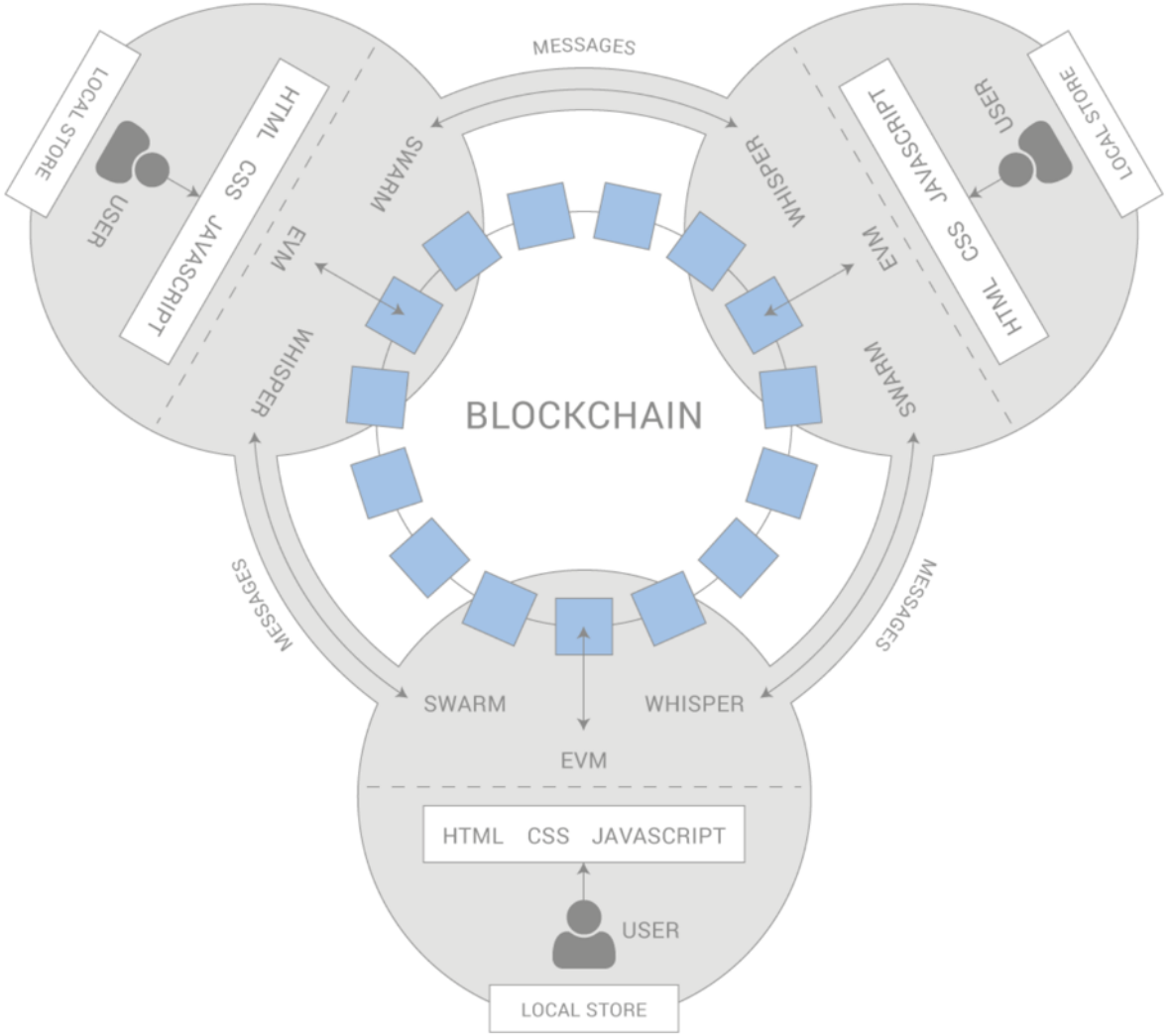


Testnet

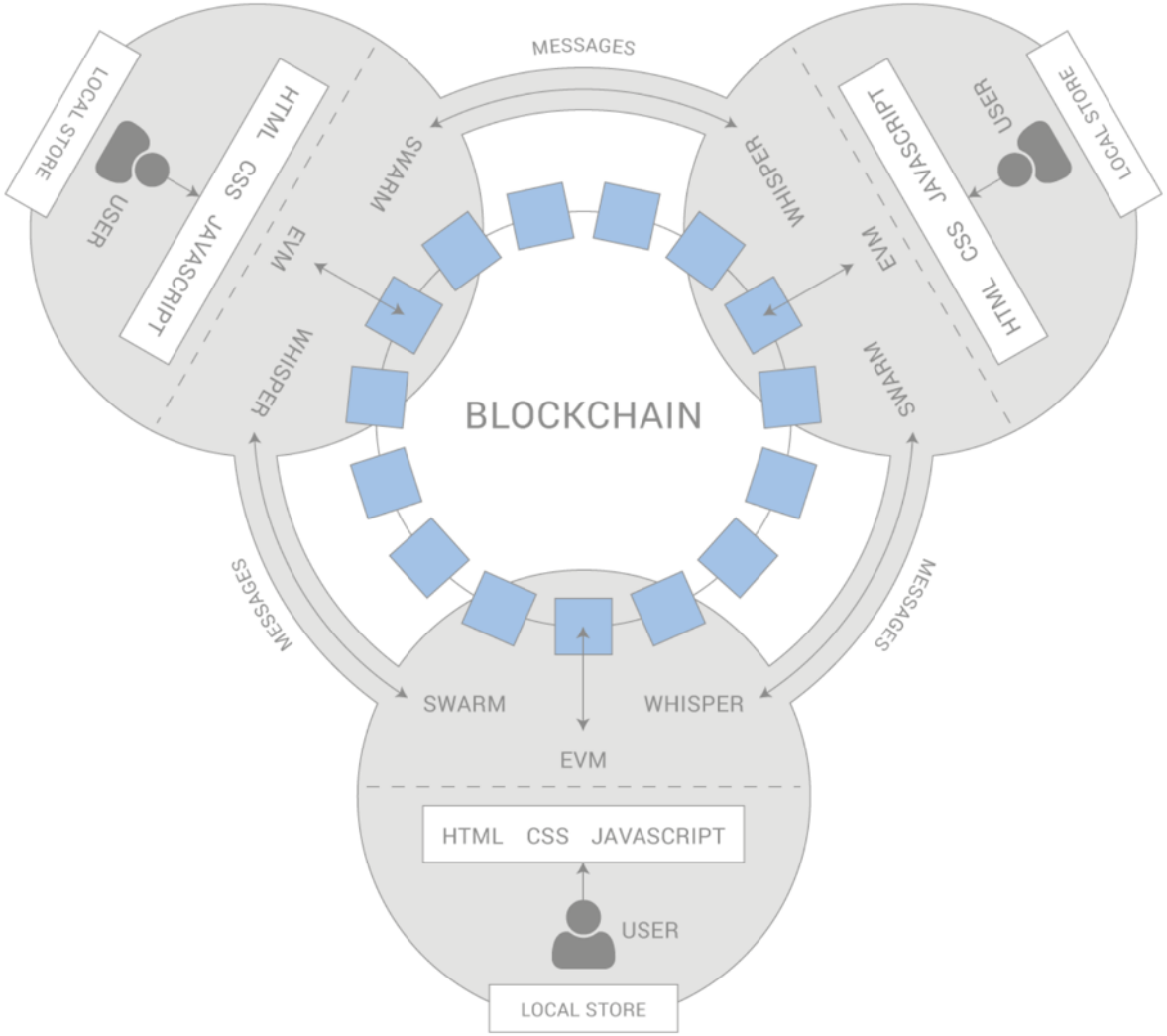


公網上所使用的鏈

Dev chain

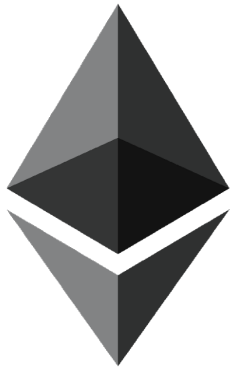


Private chain



開發上所使用的鏈，方便開發者測試

Full node

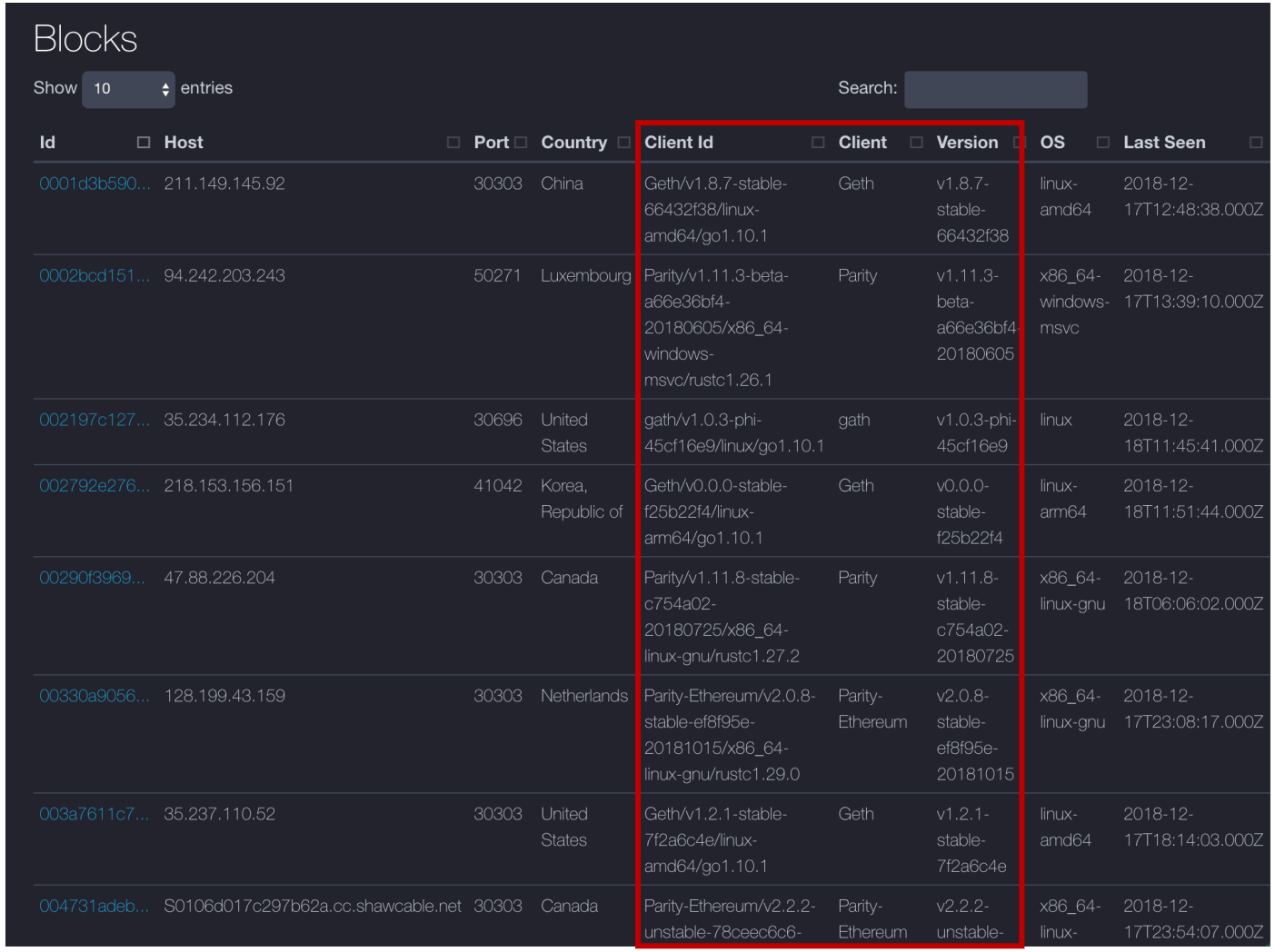


Ethereum protocol implementation

- Ethereum Client
  - Full node
  - Light client
- geth (go-ethereum)
  - https://ethereum.github.io/go-ethereum/
- parity (Rust)
  - https://www.parity.io/

parity

Ethereum nodes



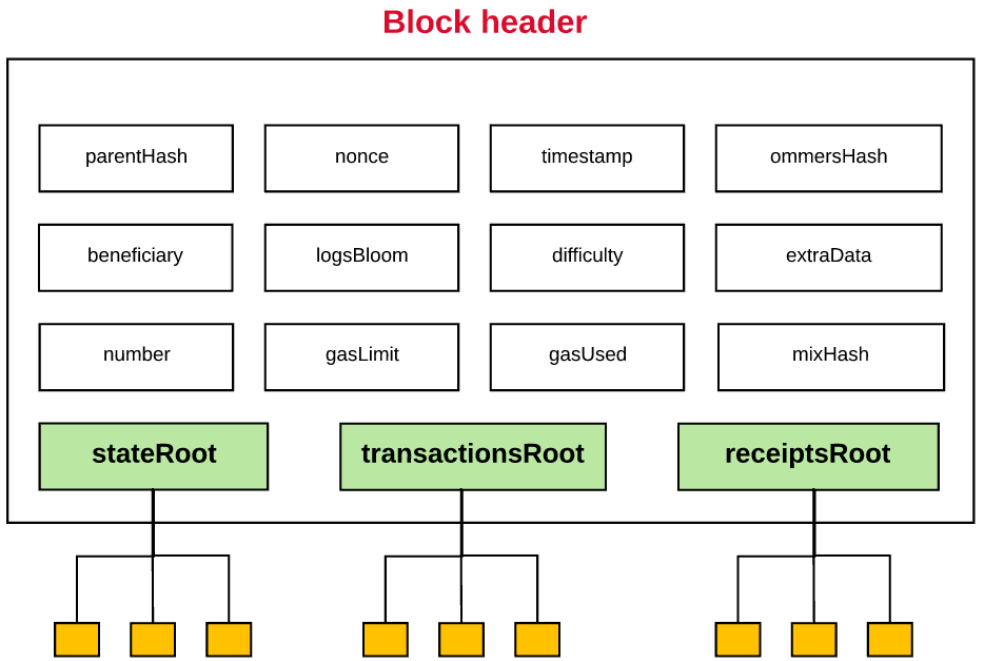
Blocks

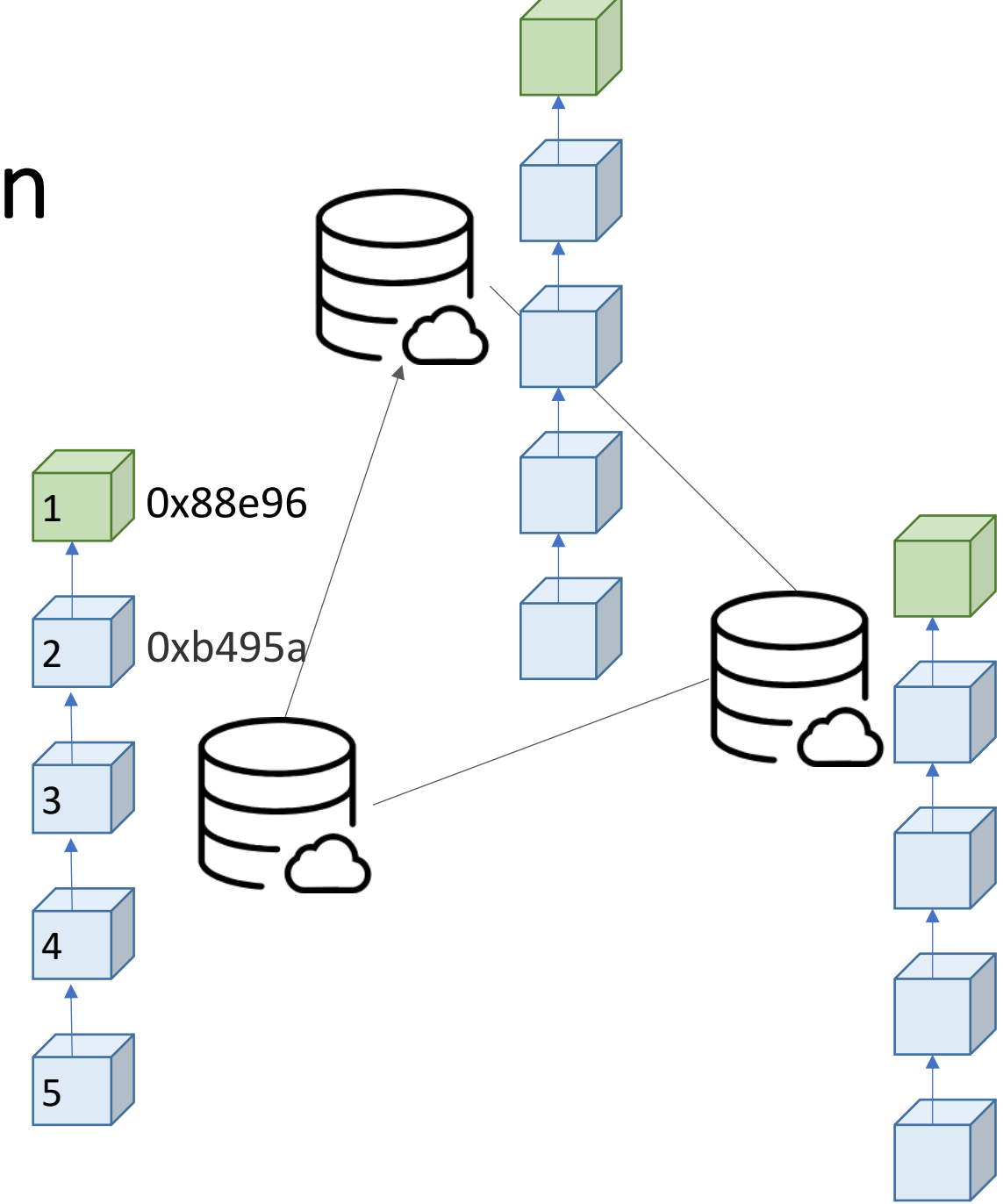
Show 10 entries Search:

Id	Host	Port	Country	Client Id	Client	Version	OS	Last Seen
0001d3b590...	211.149.145.92	30303	China	Geth/v1.8.7-stable-66432f38/linux-amd64/go1.10.1	Geth	v1.8.7-stable-66432f38	linux-amd64	2018-12-17T12:48:38.000Z
0002bcd151...	94.242.203.243	50271	Luxembourg	Parity/v1.11.3-beta-a66e36bf4-20180605/x86_64-windows-msvc/rustc1.26.1	Parity	v1.11.3-beta-a66e36bf4-20180605	x86_64-windows-msvc	2018-12-17T13:39:10.000Z
002197c127...	35.234.112.176	30696	United States	gath/v1.0.3-phi-45cf16e9/linux/go1.10.1	gath	v1.0.3-phi-45cf16e9	linux	2018-12-18T11:45:41.000Z
002792e276...	218.153.156.151	41042	Korea, Republic of	Geth/v0.0.0-stable-f25b22f4/linux-arm64/go1.10.1	Geth	v0.0.0-stable-f25b22f4	linux-arm64	2018-12-18T11:51:44.000Z
00290f3969...	47.88.226.204	30303	Canada	Parity/v1.11.8-stable-c754a02-20180725/x86_64-linux-gnu/rustc1.27.2	Parity	v1.11.8-stable-c754a02-20180725	x86_64-linux-gnu	2018-12-18T06:06:02.000Z
00330a9056...	128.199.43.159	30303	Netherlands	Parity-Ethereum/v2.0.8-stable-ef8f95e-20181015/x86_64-linux-gnu/rustc1.29.0	Parity-Ethereum	v2.0.8-stable-ef8f95e-20181015	x86_64-linux-gnu	2018-12-17T23:08:17.000Z
003a7611c7...	35.237.110.52	30303	United States	Geth/v1.2.1-stable-7f2a6c4e/linux-amd64/go1.10.1	Geth	v1.2.1-stable-7f2a6c4e	linux-amd64	2018-12-17T18:14:03.000Z
004731adeb...	S0106d017c297b62a.cc.shawcable.net	30303	Canada	Parity-Ethereum/v2.2.2-unstable-78ceec6c6-	Parity-Ethereum	v2.2.2-unstable-	x86_64-linux-	2018-12-17T23:54:07.000Z

https://www.ethernodes.org/network/1/nodes

Node synchronization



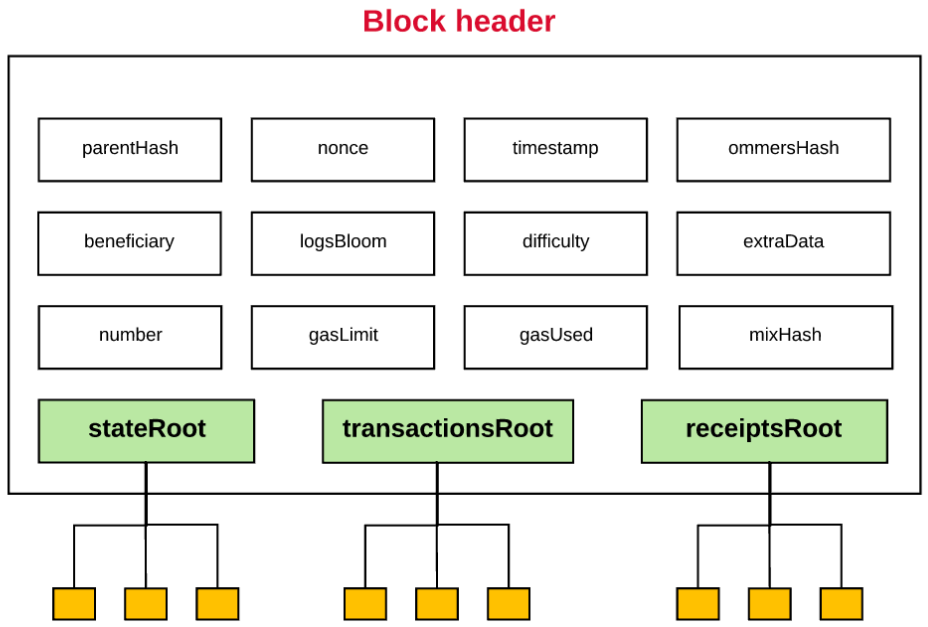


Block header

1. geth attach http://127.0.0.1:8545 (geth)

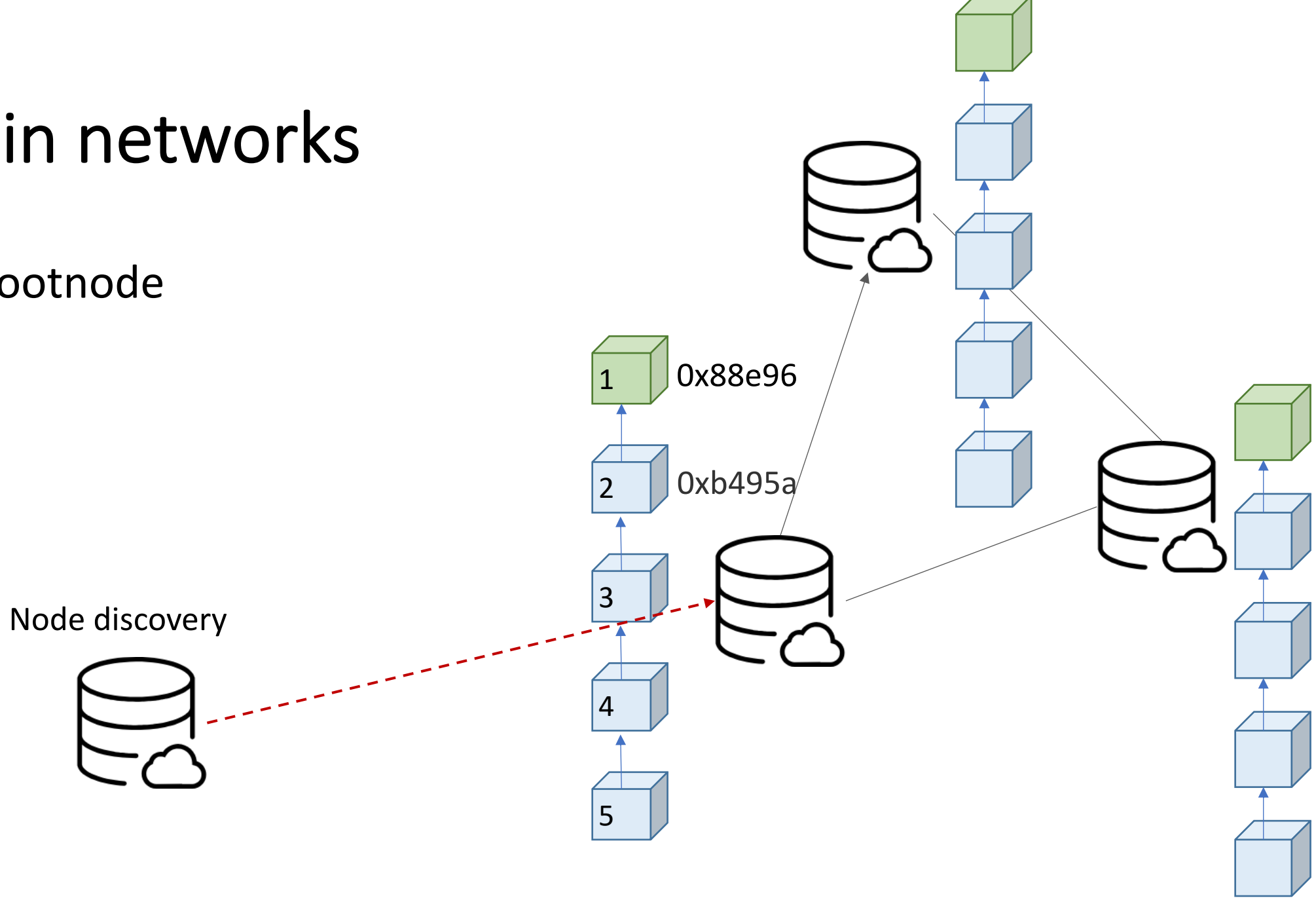
geth (geth) ⌘1 | geth (geth) ⌘2 | ~ (zsh) ⌘3 | ~ (zsh) ⌘4


```
{
  difficulty: 2,
  extraData: "0xd783010810846765746886676f312e3131866461727769 6e0000000000000000bf481b42df532442d9cfeab2828863df0b4efb
0544e169db7f03069828f72801365711137eeb9a7c6a423ae60fee5f0b41182003eff46259d0a6f0f1f7e7e69500",
  gasLimit: 6289319,
  gasUsed: 0,
  hash: "0x8cd93ff67645d3a1b96952ca3f55296a8ebebf66c4f63eb5daf7e7bc0492459c",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000",
  miner: "0x0000000000000000000000000000000000000000",
  mixHash: "0x0000000000000000000000000000000000000000000000000000000000000000",
  nonce: "0x0000000000000000",
  number: 1,
  parentHash: "0xd8e5944bddc81104674d8446c44402e90bbd75930f6917460b22eeff04b1263f",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 606,
  stateRoot: "0xeedbe7f7c84bec5eb68c548b47d7d905516e19117f080ffa3ed44b8def3acd4c",
  timestamp: 1545135093,
  totalDifficulty: 3,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
> _
```

Join networks

- Bootnode



https://github.com/ethereum/go-ethereum/wiki/Connecting-to-the-network

Bootnode list

- Hardcode
- DNS

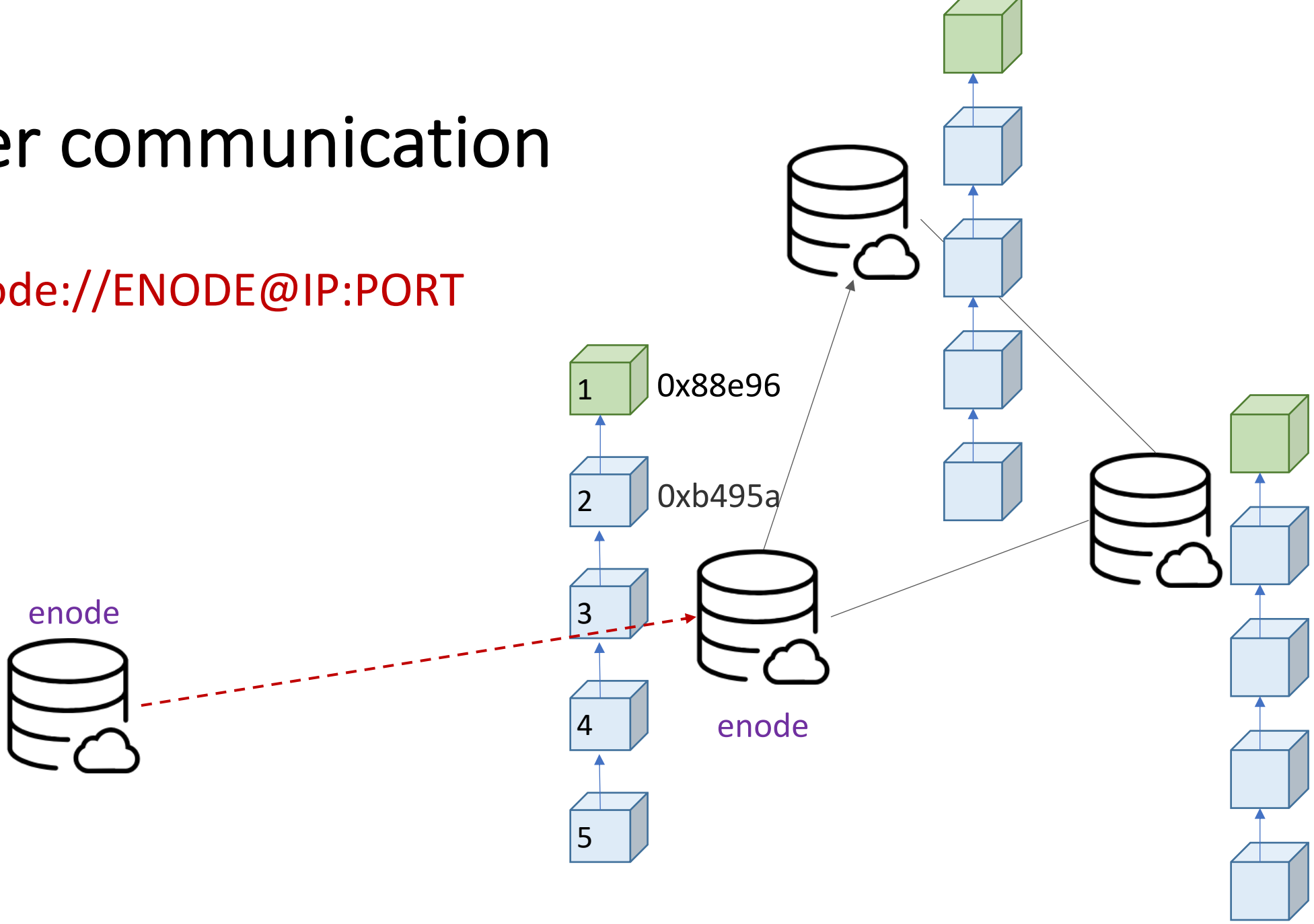
```
var MainnetBootnodes = []string{
    // Ethereum Foundation Go Bootnodes
    "enode://a979fb575495b8d6db44f750317d0f4622bf4c2aa3365d6af7c284339968eef29b69ad0dce72a4d8db5ebb4968de0e3bec910127f134779fbcb0cb6d3331163c@52.16.188.185:30303", // IE
    "enode://3f1d12044546b76342d59d4a05532c14b85aa669704bfe1f864fe079415aa2c02d743e03218e57a33fb94523adb54032871a6c51b2cc5514cb7c7e35b3ed0a99@13.93.211.84:30303", // US-WEST
    "enode://78de8a0916848093c73790ead81d1928bec737d565119932b98c6b100d944b7a95e94f847f689fc723399d2e31129d182f7ef3863f2b4c820abbf3ab2722344d@191.235.84.50:30303", // BR
    "enode://158f8aab45f6d19c6cbf4a089c2670541a8da11978a2f90dbf6a502a4a3bab80d288afdbeb7ec0ef6d92de563767f3b1ea9e8e334ca711e9f8e2df5a0385e8e6@13.75.154.138:30303", // AU
    "enode://1118980bf48b0a3640bdba04e0fe78b1add18e1cd99bf22d53daac1fd9972ad650df52176e7c7d89d1114cfef2bc23a2959aa54998a46afcf7d91809f0855082@52.74.57.123:30303", // SG

    // Ethereum Foundation C++ Bootnodes
    "enode://979b7fa28feeb35a4741660a16076f1943202cb72b6af70d327f053e248bab9ba81760f39d0701ef1d8f89cc1fbd2cacba0710a12cd5314d5e0c9021aa3637f9@5.1.83.226:30303", // DE
}
```


https://github.com/ethereum/go-ethereum/blob/master/params/bootnodes.go

Peer communication

- enode://ENODE@IP:PORT



Bootnode command

```
$ cd go-ethereum
$ make all
$ cd build/bin


// bootnode command
$ bootnode -nodekeyhex [privkey] -writeaddress

// example
$ bootnode -nodekeyhex af713077fa1244509d9872b989f7776e428a0a6afd4a77f83deabd15d80ec869 -writeaddress
f6449e7acaef2a59c2bb269260a89f6782c58976abf94ea3f01f67e6693b795cb2ec1287678f3ea615004f5993e78b0fe7e8739ffe2ac3
2a7fbdb5cbf62f81ef
```

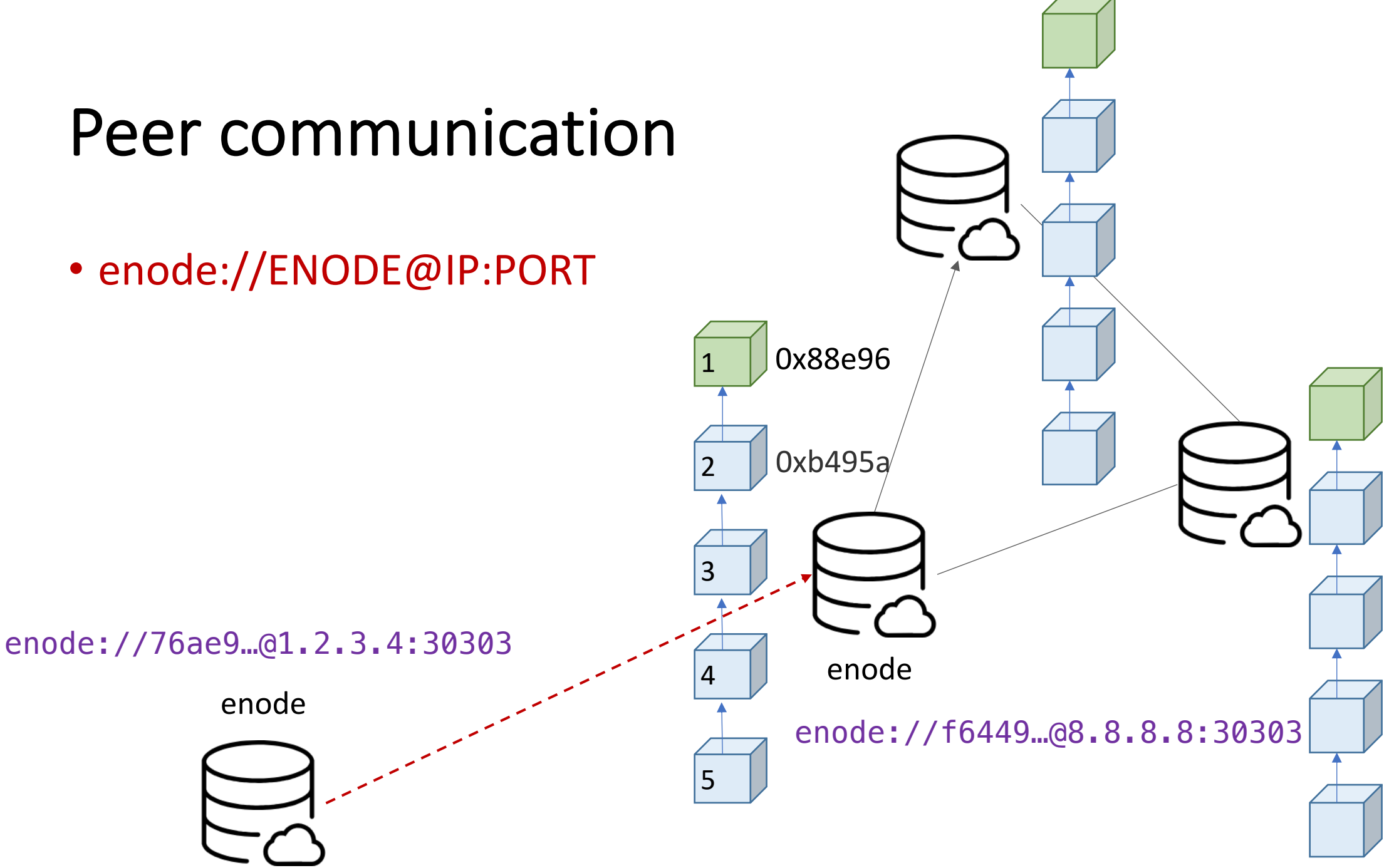
enode://**ENODE**@IP:PORT

```
enode://f6449e7acaef2a59c2bb269260a89f6782c58976abf94ea3f01f67e6693b795cb2ec1287678f
3ea615004f5993e78b0fe7e8739ffe2ac32a7fbdb5cbf62f81ef@8.8.8.8:30303
```


https://ethereum.stackexchange.com/questions/28970/how-to-produce-enode-from-node-key/28978

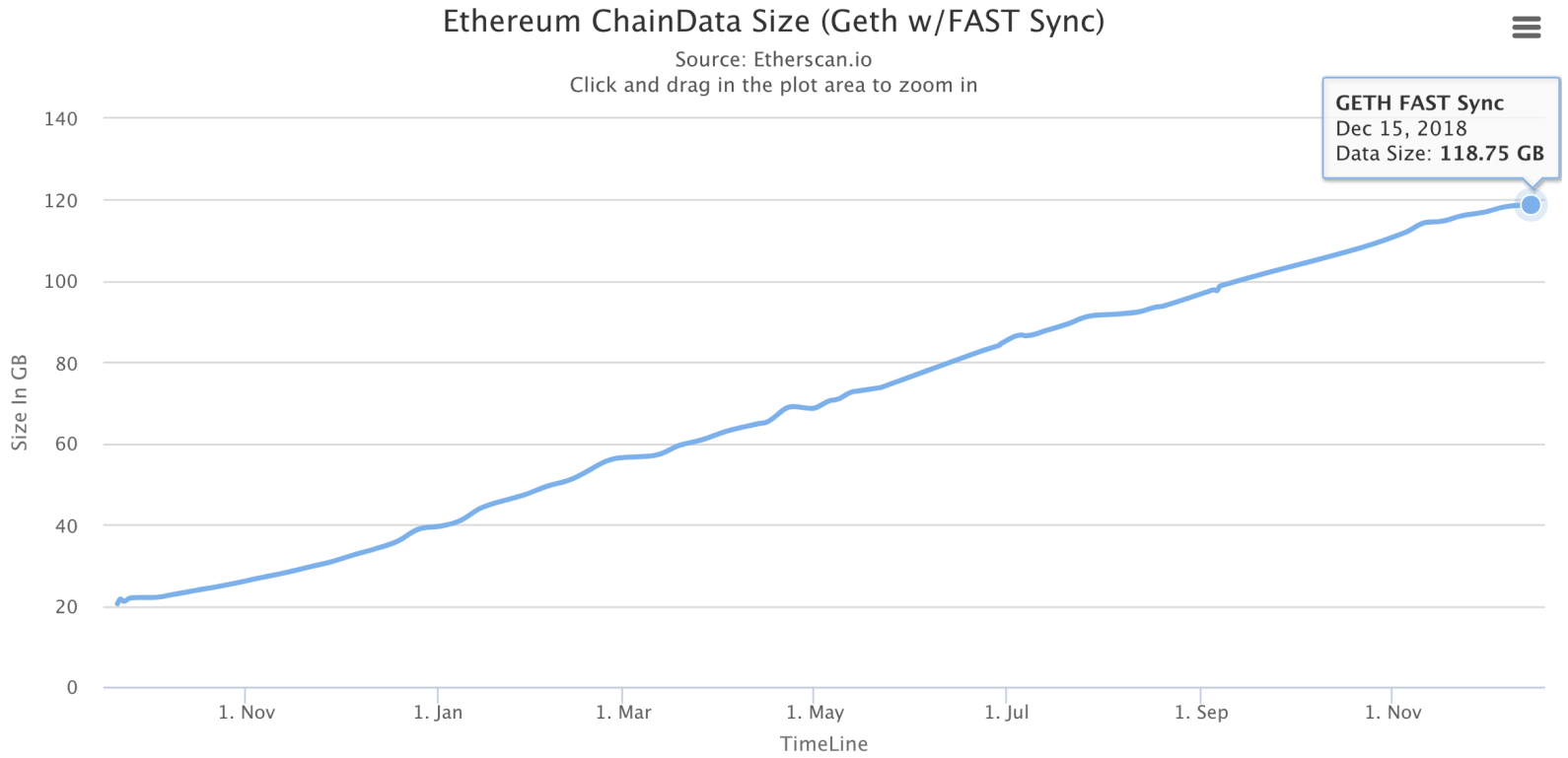
Peer communication

- enode://ENODE@IP:PORT

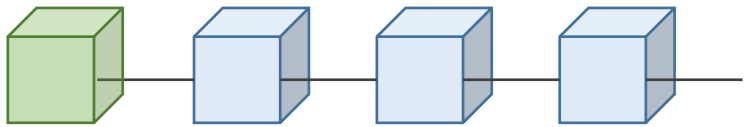


geth modes

- Disk: **SSD**
- --syncmode
  - **fast**
  - full
  - light
- --gcmode
  - **full**
  - archive



Get block headers and block bodies until current block - 64(*)



https://ethereum.stackexchange.com/questions/11297/what-is-geths-light-sync-and-why-is-it-so-fast

geth

```
$ geth --syncmode "full" --gcmode "archive" --rpc --rpcapi
"db,eth,net,web3,personal" --cache=1024 --rpcport 8545 --rpcaddr 0.0.0.0 -
-rpccorsdomain "*"

$ geth --testnet --rpc --rpcapi "db,eth,net,web3,personal" --cache=1024 --
rpcport 8545 --rpcaddr 0.0.0.0 --rpccorsdomain "*"
```

- Full + archive node
  - History
  - Present state
  - All historical states

Chain types

- mainnet
- rospten
- rinkeby
  - (Proof-of-Authority network)

- 0: Olympic, Ethereum public pre-release testnet
- 1: Frontier, Homestead, Metropolis, the Ethereum public main network
- 1: Classic, the (un)forked public Ethereum Classic main network, *chain ID 61*
- 1: Expanse, an alternative Ethereum implementation, *chain ID 2*
- 2: Morden, the public Ethereum testnet, now Ethereum Classic testnet
- 3: Ropsten, the public cross-client Ethereum testnet
- 4: Rinkeby, the public Geth PoA testnet

https://ethereum.stackexchange.com/questions/17051/how-to-select-a-network-id-or-is-there-a-list-of-network-ids

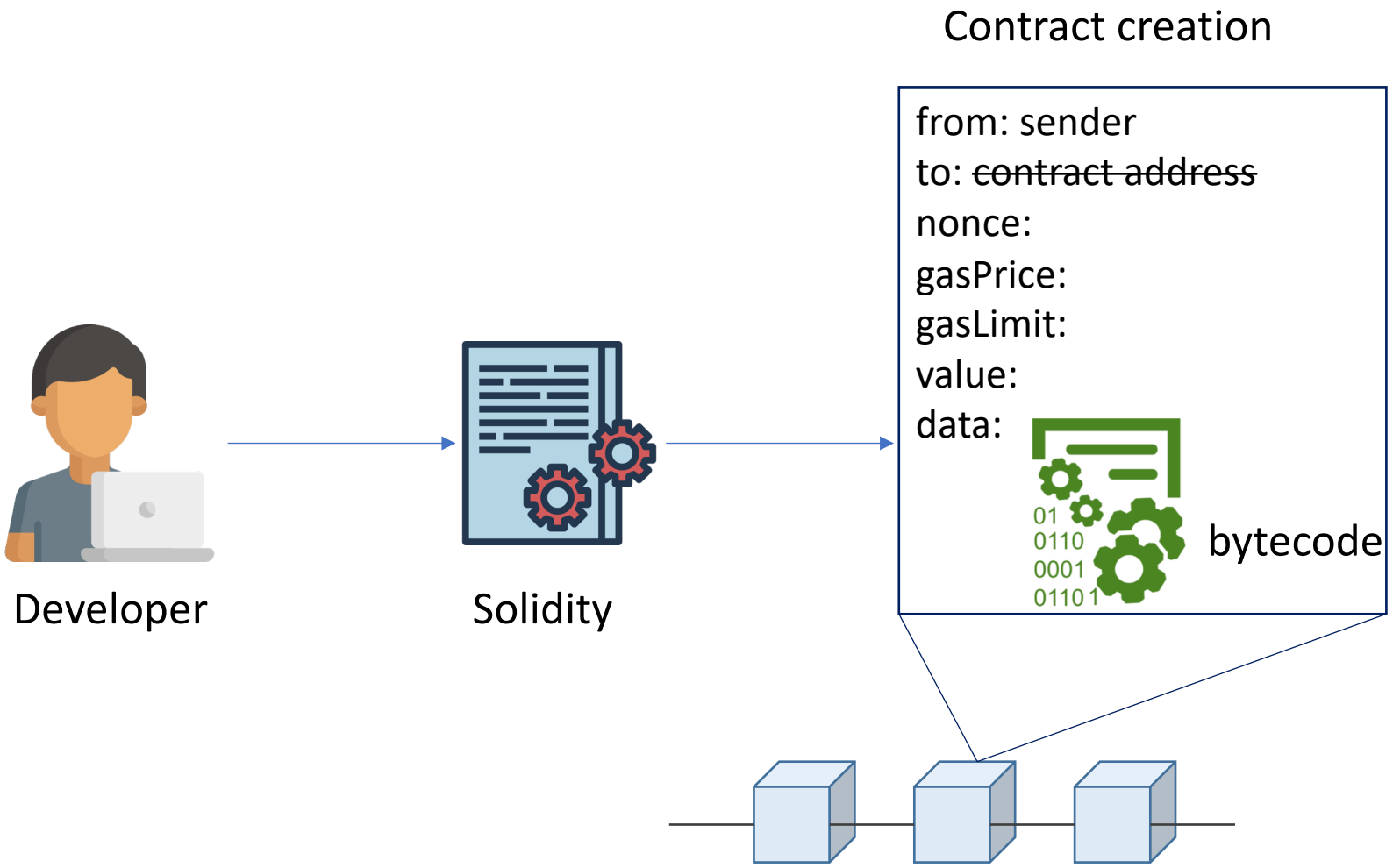
Comparison of the different TestNets

- **Ropsten**
  - PoW
  - Supported by geth and parity
  - Best reproduces the current production environment
  - Chaindata size 15 GB - Apr 2018
- **Kovan**
  - PoA (Immune to spam attacks)
  - Supported by parity only
  - Chaindata size 13 GB - Apr 2018
- **Rinkeby**
  - PoA (Immune to spam attacks)
  - Supported by geth only
  - Chaindata size 6 GB - Apr 2018
- **Sokol**
  - PoA (Immune to spam attacks)
  - Supported by parity only
  - Chaindata size 5gb - Jun 2018

testnet 資料會定期清空
千萬不要用於 production

https://ethereum.stackexchange.com/questions/27048/comparison-of-the-different-testnets

Contract development

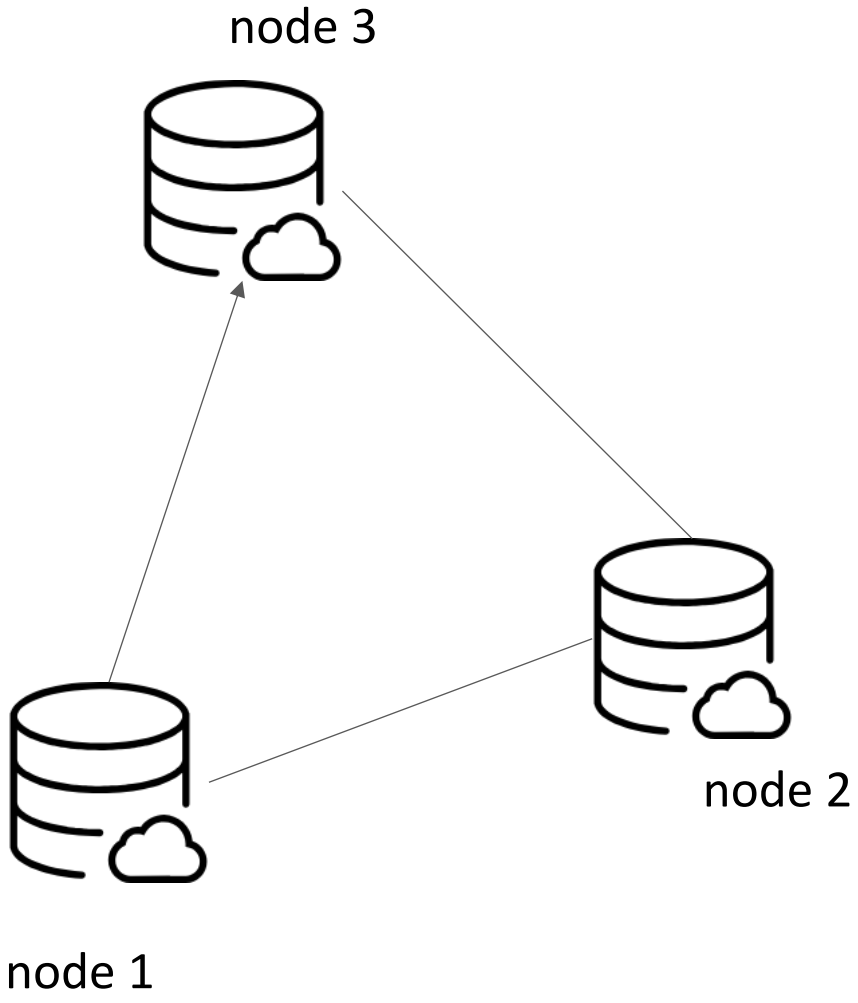


1. Solidity version
2. Contract vulnerability
3. Enough funds
4. Reasonable gas price
5. Node management
6. Contract maintenance
7. Data migration
8. Data confidentiality
9. Data privacy
10. External event
11. Optimized gas usage
12. Test

Private net

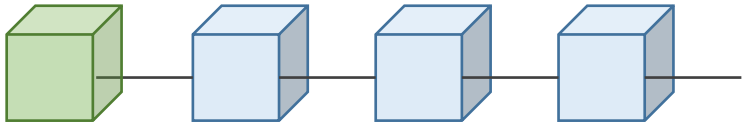
以太坊私有链

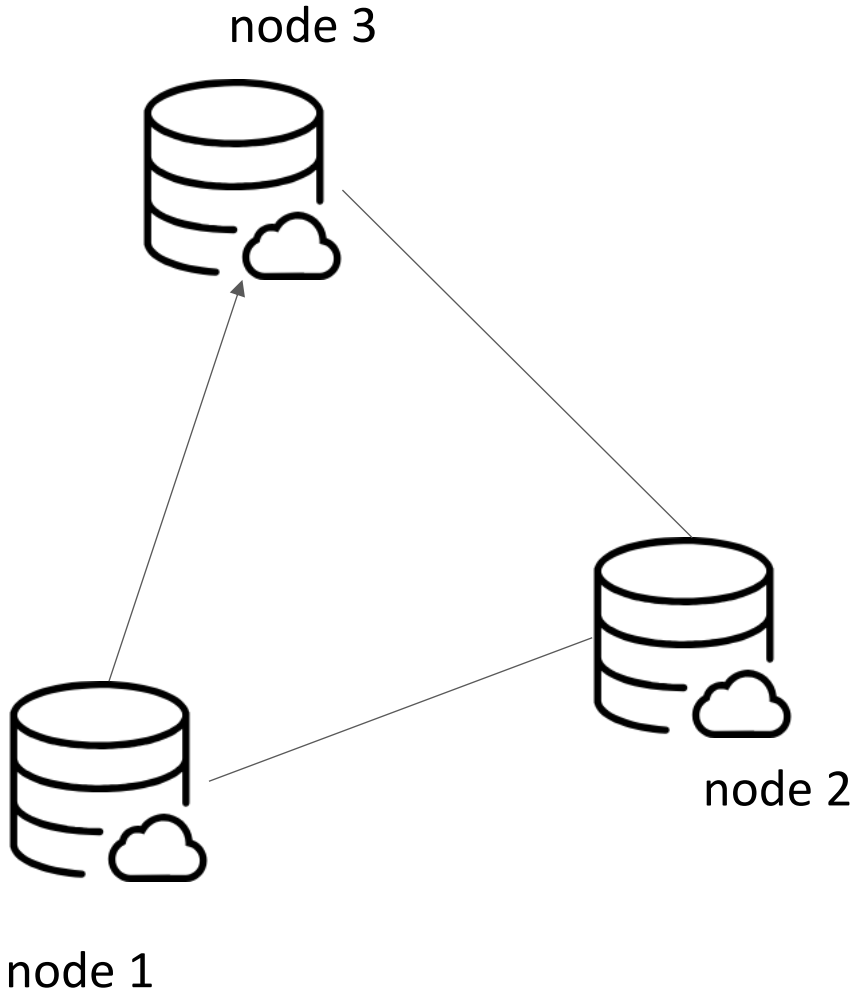
- 客製化的私有鏈
- 實驗
- 挖礦難度較易且可設定



環境設定

- 三台機器
- 創世區塊(genesis block)設定相同
- geth client

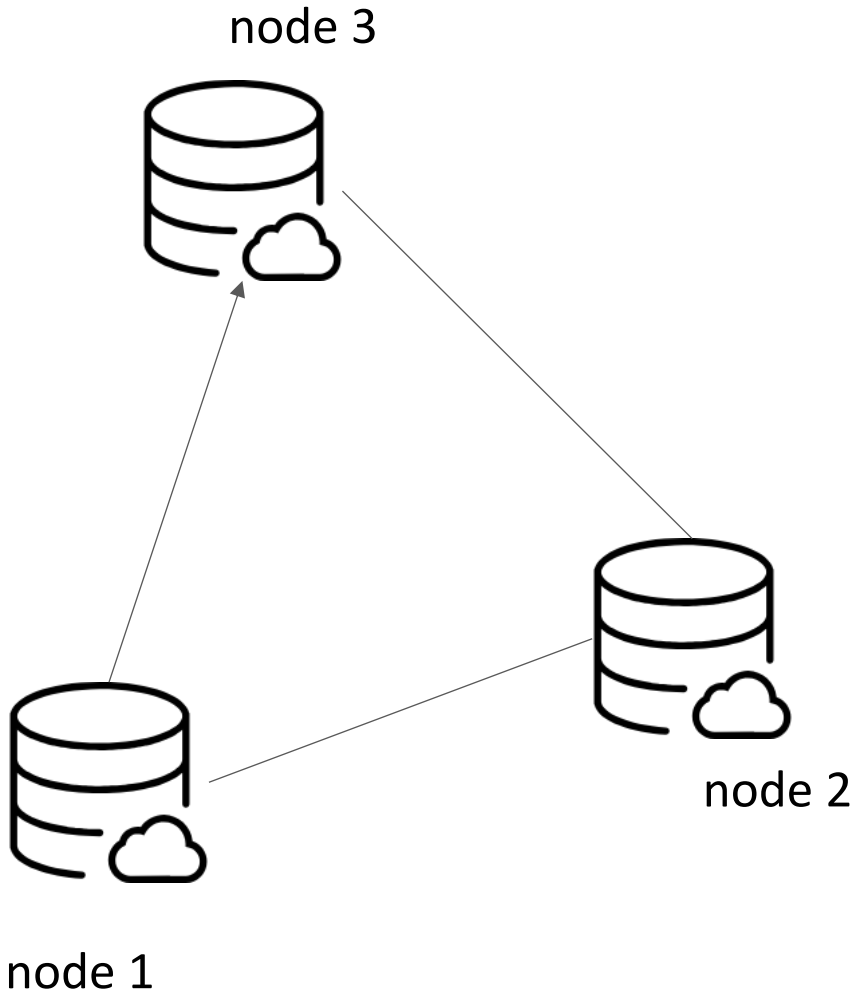




檔案目錄

```
.
├── genesis.json
├── node1
└── node2

2 directories, 1 file
```



genesis.json

```
{
    "alloc": {
        "0xb36fc7a57bbbc5ba62a51427acb6ad37bc8be4a0": {
            "balance": "1000000000000000000000"
        },
        "0x1635e5014dbcd3df2f21744eaa2a868ee69d25f2": {
            "balance": "1000000000000000000000000"
        }
    },
    "config": {
        "chainID": 5566,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
    "nonce": "0x0000000000000000",
    "difficulty": "0x0400",
    "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "coinbase": "0x0000000000000000000000000000000000000000",
    "timestamp": "0x00",
    "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "extraData": "0x6e636375",
    "gasLimit": "0x7a121d"
}
```

初始化 node1

```
$ geth --datadir node1 init genesis.json
```

```
$ geth --datadir node1 init genesis.json
INFO [12-18|21:37:55.025] Maximum peer count                       ETH=25 LES=0 total=25
INFO [12-18|21:37:55.035] Allocated cache and file handles         database=/Users/changwu/nccu-course/nodes/node1/get
h/chaindata cache=16 handles=16
INFO [12-18|21:37:55.041] Writing custom genesis block
INFO [12-18|21:37:55.042] Persisted trie from memory database      nodes=3 size=414.00B time=384.218µs gcnodes=0 gcsiz
e=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [12-18|21:37:55.043] Successfully wrote genesis state         database=chaindata
            hash=d7b205…1e165a
INFO [12-18|21:37:55.043] Allocated cache and file handles         database=/Users/changwu/nccu-course/nodes/node1/get
h/lightchaindata cache=16 handles=16
INFO [12-18|21:37:55.047] Writing custom genesis block
INFO [12-18|21:37:55.047] Persisted trie from memory database      nodes=3 size=414.00B time=104.212µs gcnodes=0 gcsiz
e=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [12-18|21:37:55.048] Successfully wrote genesis state         database=lightchaindata
                hash=d7b205…1e165a
```

初始化 node2

```
$ geth --datadir node2 init genesis.json
```

```
$ geth --datadir node2 init genesis.json
INFO [12-18|21:39:41.963] Maximum peer count                       ETH=25 LES=0 total=25
INFO [12-18|21:39:41.974] Allocated cache and file handles         database=/Users/changwu/nccu-course/nodes/node2/get
h/chaindata cache=16 handles=16
INFO [12-18|21:39:41.977] Writing custom genesis block
INFO [12-18|21:39:41.978] Persisted trie from memory database      nodes=3 size=414.00B time=94.999µs gcnodes=0 gcsize
=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [12-18|21:39:41.978] Successfully wrote genesis state         database=chaindata
            hash=d7b205…1e165a
INFO [12-18|21:39:41.978] Allocated cache and file handles         database=/Users/changwu/nccu-course/nodes/node2/get
h/lightchaindata cache=16 handles=16
INFO [12-18|21:39:41.982] Writing custom genesis block
INFO [12-18|21:39:41.983] Persisted trie from memory database      nodes=3 size=414.00B time=100.576µs gcnodes=0 gcsiz
e=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [12-18|21:39:41.983] Successfully wrote genesis state         database=lightchaindata
                hash=d7b205…1e165a
```

建立帳號 node1

```
$ geth --datadir "node1" account new
```

```
$ geth --datadir "node1" account new

INFO [12-18|21:48:39.434] Maximum peer count                       ETH=25 LES=0 total=25
Your new account is locked with a password. Please give a password. Do not forget this password.
Passphrase:
Repeat passphrase:
Address: {3eaff2c7bd8bb391fc843fdc7eb55d02c4a5aa35}
```

建立帳號 node2

```
$ geth --datadir "node2" account new
```

```
$ geth --datadir "node2" account new
INFO [12-18|21:48:48.789] Maximum peer count                       ETH=25 LES=0 total=25
Your new account is locked with a password. Please give a password. Do not forget this password.
Passphrase:
Repeat passphrase:

Address: {55d07fd4da7b887817331c219219cc951301dfe1}
```

啟動 geth (node1)

```
$ geth --identity "node1" --datadir "node1" --rpc --rpcport 7788 --
rpccorsdomain "*" --port 30301 --nodiscover --rpcapi
"db,eth,net,web3,personal" --networkid 1234 console
```

identity	給予當前節點命名
port	P2P 通訊 port
rpc	啟用 rpc
rpcapi	允許 client 使用的 rpc module
networkid	網路 ID
console	命令列模式

啟動 geth (node2)

```
$ geth --identity "node2" --datadir "node2" --rpc --rpcport 5566 --
rpccorsdomain "*" --port 30302 --nodiscover --rpcapi
"db,eth,net,web3,personal" --networkid 1234 console
```

identity	給予當前節點命名
port	P2P 通訊 port
rpc	啟用 rpc
rpcapi	允許 client 使用的 rpc module
networkid	網路 ID
console	命令列模式

查看節點帳號

```
(node1)
> eth.coinbase
"0x3eaff2c7bd8bb391fc843fdc7eb55d02c4a5aa35”

(node2)
> eth.coinbase
"0x55d07fd4da7b887817331c219219cc951301dfe1"
```

查看連接的節點

```
> admin.peers
[]
```

因為目前沒有任何連接數，所以為空

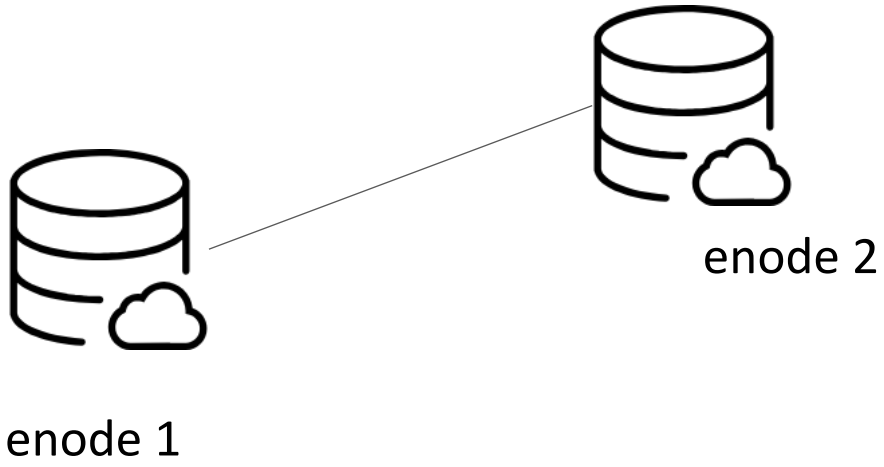
取得 node2 的 enode ID

```
(node2)
> admin.nodeInfo.enode
"enode://c7b938c499881ac1d0dfd29cb37a0c67f93e6811b6a0b0f0093fd9d81e5fd08ed
56670f689ea79e9af22f7c17da8e8323234dc26b23099e9fe2ad1b556af116a@[::]:30302
?discport=0"
```

```
enode://c7b938c499881ac1d0dfd29cb37a0c67f93e6811b6a0b0f0093fd9d81e5
fd08ed56670f689ea79e9af22f7c17da8e8323234dc26b23099e9fe2ad1b556af11
6a@127.0.0.1:30302
```

Connect node 1 with node 2

```
>
admin.addPeer("enode://c7b938c499881ac1d0dfd29cb37a0c67f93e6811b6a0b0f0093
fd9d81e5fd08ed56670f689ea79e9af22f7c17da8e8323234dc26b23099e9fe2ad1b556af1
16a@127.0.0.1:30302")
true
```



查看連接的節點

```
(node1)
> admin.peers
[{
    caps: ["eth/63"],
    id:
"c7b938c499881ac1d0dfd29cb37a0c67f93e6811b6a0b0f0093fd9d81e5fd08ed56670f689ea79e9af22f7c17da8e832
3234dc26b23099e9fe2ad1b556af116a",
    name: "Geth/node2/v1.8.16-unstable-62e94895/darwin-amd64/go1.11",
    network: {
        inbound: false,
        localAddress: "127.0.0.1:51246",
        remoteAddress: "127.0.0.1:30302",
        static: true,
        trusted: false
    },
    protocols: {
        eth: {
            difficulty: 17179869184,
            head: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
            version: 63
        }
    }
}]
>
```

查看區塊數目

```
(node1)
> eth.blockNumber
0
```

查看帳戶餘額

```
(node1)
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")
0
```

```
(node1)
> web3.fromWei(eth.getBalance("0xb36fc7a57bbbc5ba62a51427acb6ad37bc8be4a0"), "ether")
10000
```

啟動挖礦

```
> miner.start()
INFO [12-18|22:43:41.865] Updated mining threads threads=4
INFO [12-18|22:43:41.865] Transaction pool price threshold updated price=1000000000
null
> INFO [12-18|22:43:41.866] Commit new mining work number=1 sealhash=27e1aa…6f5a0d
uncles=0 txs=0 gas=0 fees=0 elapsed=677.102µs
INFO [12-18|22:43:42.453] Successfully sealed new block number=1 sealhash=27e1aa…6f5a0d
hash=f36304…0090fc elapsed=587.048ms
INFO [12-18|22:43:42.453] 🔨 mined potential block number=1 hash=f36304…0090fc
INFO [12-18|22:43:42.453] Commit new mining work number=2 sealhash=46eed2…58fc64 uncles=0
txs=0 gas=0 fees=0 elapsed=242.039µs
INFO [12-18|22:43:43.557] Successfully sealed new block number=2 sealhash=46eed2…58fc64
hash=8b2f4f…f39249 elapsed=1.104s
```

attach to node1

```
$ geth attach http://127.0.0.1:5566
Welcome to the Geth JavaScript console!

instance: Geth/node2/v1.8.16-unstable-62e94895/darwin-amd64/go1.11
coinbase: 0x55d07fd4da7b887817331c219219cc951301dfe1
at block: 38 (Tue, 18 Dec 2018 22:44:25 CST)
modules: eth:1.0 net:1.0 personal:1.0 rpc:1.0 web3:1.0

>
```

attach to node2

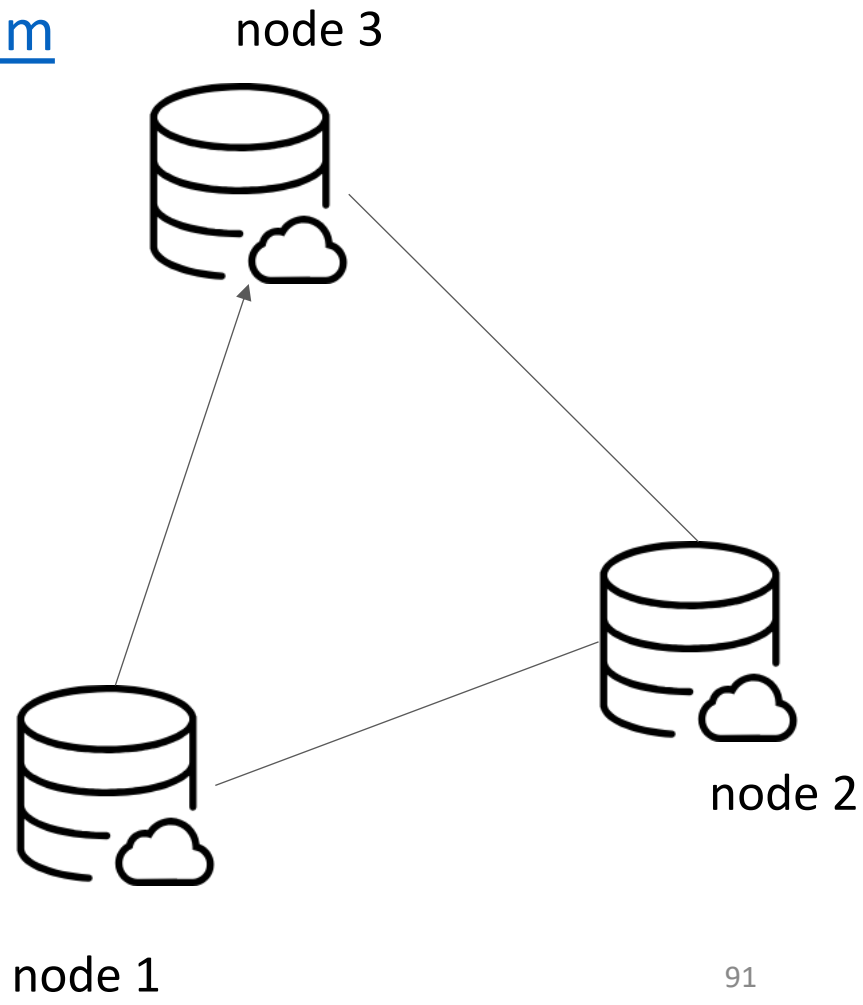
```
$ geth attach http://127.0.0.1:7788
Welcome to the Geth JavaScript console!

instance: Geth/node1/v1.8.16-unstable-62e94895/darwin-amd64/go1.11
coinbase: 0x3eaff2c7bd8bb391fc843fdc7eb55d02c4a5aa35
at block: 29 (Tue, 18 Dec 2018 22:44:15 CST)
modules: eth:1.0 net:1.0 personal:1.0 rpc:1.0 web3:1.0

>
```

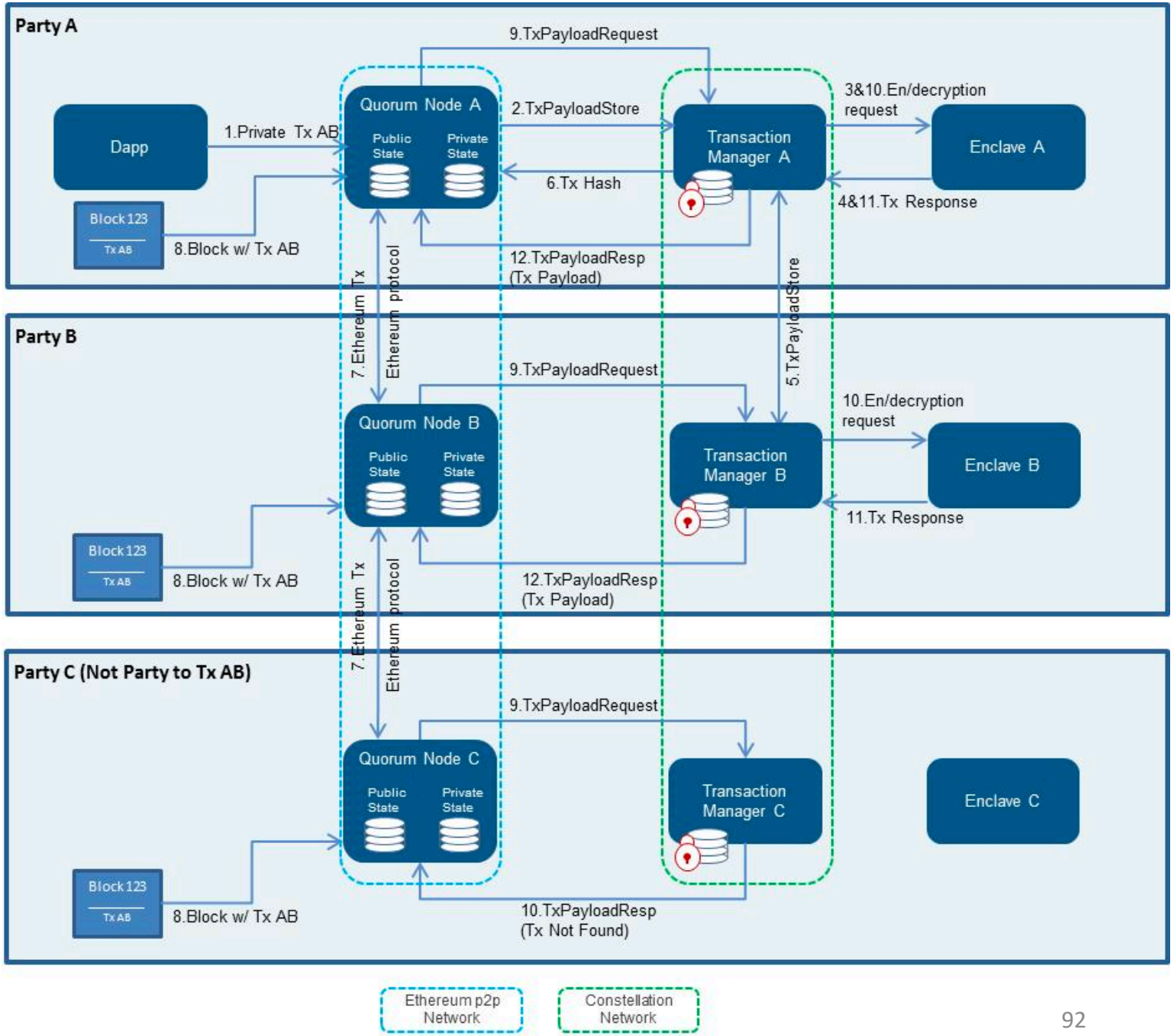
Clique

- https://ethfans.org/posts/Clique-Consensus-Algorithm
- 不同共識演算法，非 PoW
- 輪流出塊
- 認證節點 (block proposer)
- 非認證節點 (only sync block)



quorum

- 企業私有鏈



https://github.com/jpmorganchase/quorum

References

- https://feelncut.com/2018/03/05/102.html
- https://arvanaghi.com/blog/how-to-set-up-a-private-ethereum-blockchain-using-geth/
- https://arvanaghi.com/blog/explaining-the-genesis-block-in-ethereum/

devcon4 workshop

- http://www.cryptokube.io/
- Slide
  - http://www.cryptokube.io/Devcon4-ArchitectingWithEthereum.pdf
- Exercise
  - https://github.com/CryptoKube-io/devcon4-workshop#exercises

Blockchain as a Service

- Run your own Ethereum node
  - https://kaleido.io/
  - https://quiknode.io/

- Decentralized P2P network
  - https://dappnode.io/

Book

- https://github.com/ethereumbook/ethereumbook

Trie nodes

- **NULL** (represented as the empty string)
- **Branch** A 17-item node [v0 ... v15, vt]
- **Leaf** A 2-item node [encodedPath, value]
- **Extension** A 2-item node [encodedPath, key]

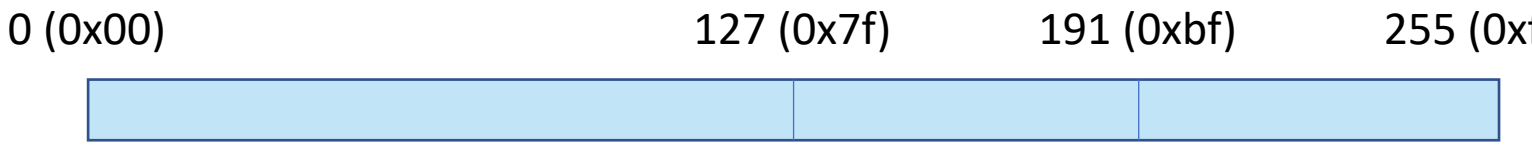
https://github.com/ethereum/wiki/wiki/RLP

RLP

- RLP ("recursive length prefix") encoding is the main encoding method used to serialize objects in Ethereum, and is used everywhere
  - Blocks
  - Transactions
  - Account state data
  - Wire protocol messages
- Encoding/decoding

https://github.com/ethereum/wiki/wiki/RLP

RLP



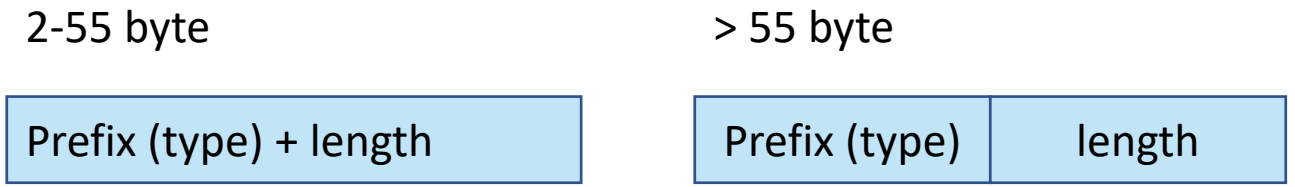
- [0x00, 0x7f]: byte
- [0x80, 0xbf]: string
- [0xc0, 0xff]: list

- The string "dog" = [0x83, 'd', 'o', 'g']
- The list ["cat", "dog"] = [0xc8, 0x83, 'c', 'a', 't', 0x83, 'd', 'o', 'g']
- The empty list = [0xc0]
- The integer 0 = [0x80]
- The empty string ('null') = [0x80]
- The byte('') = [0x80]

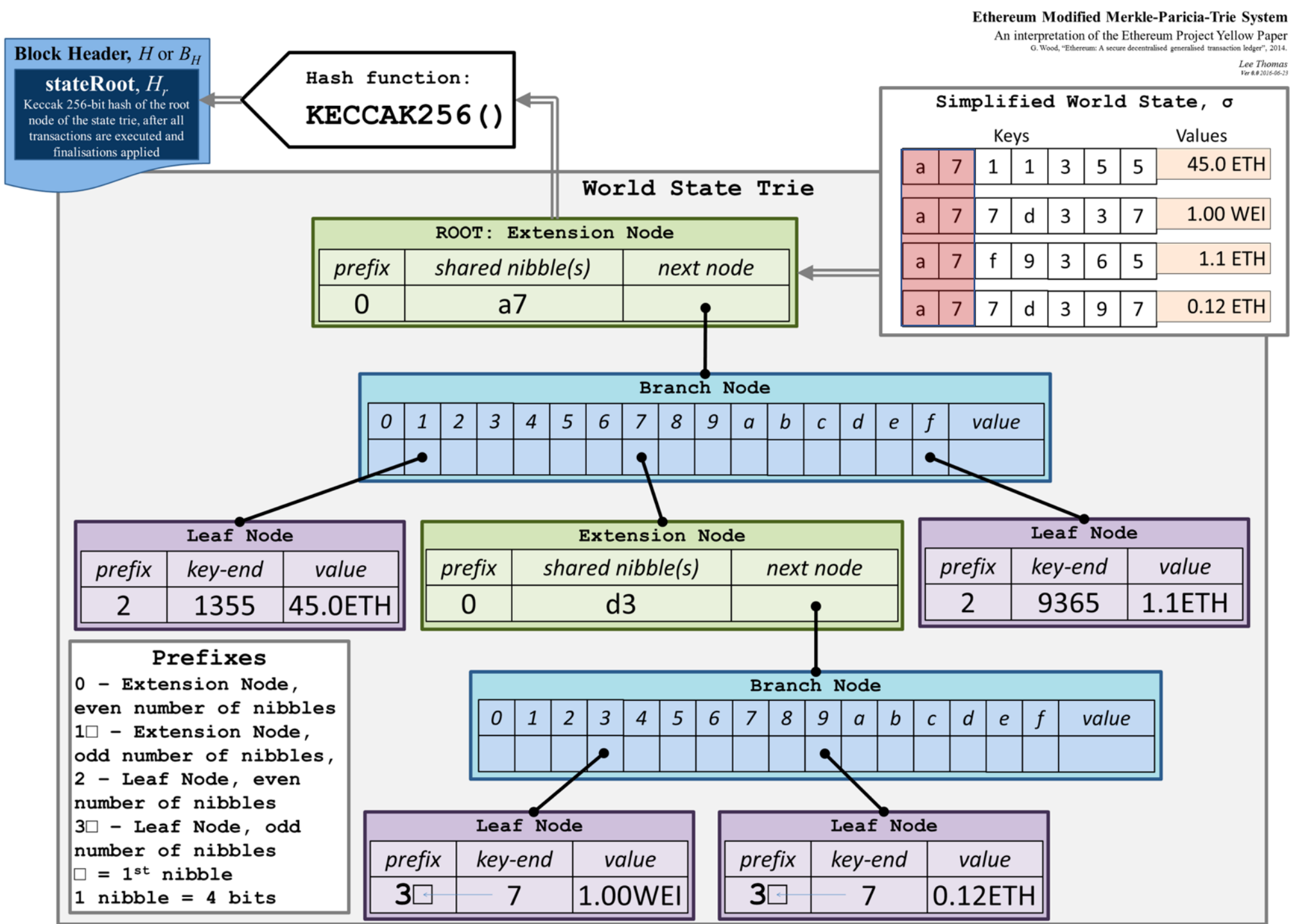
https://github.com/ethereum/wiki/wiki/RLP

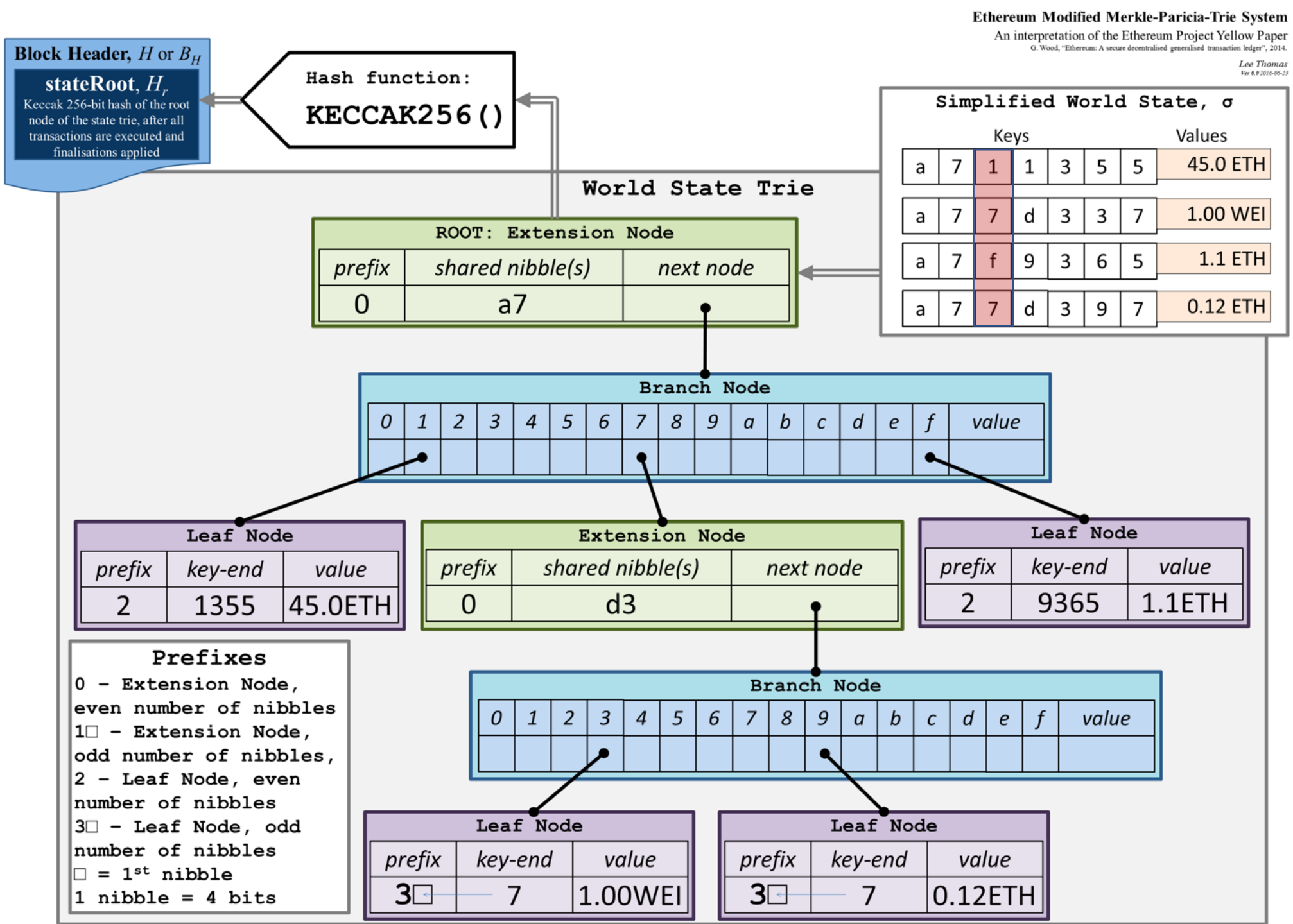
RLP

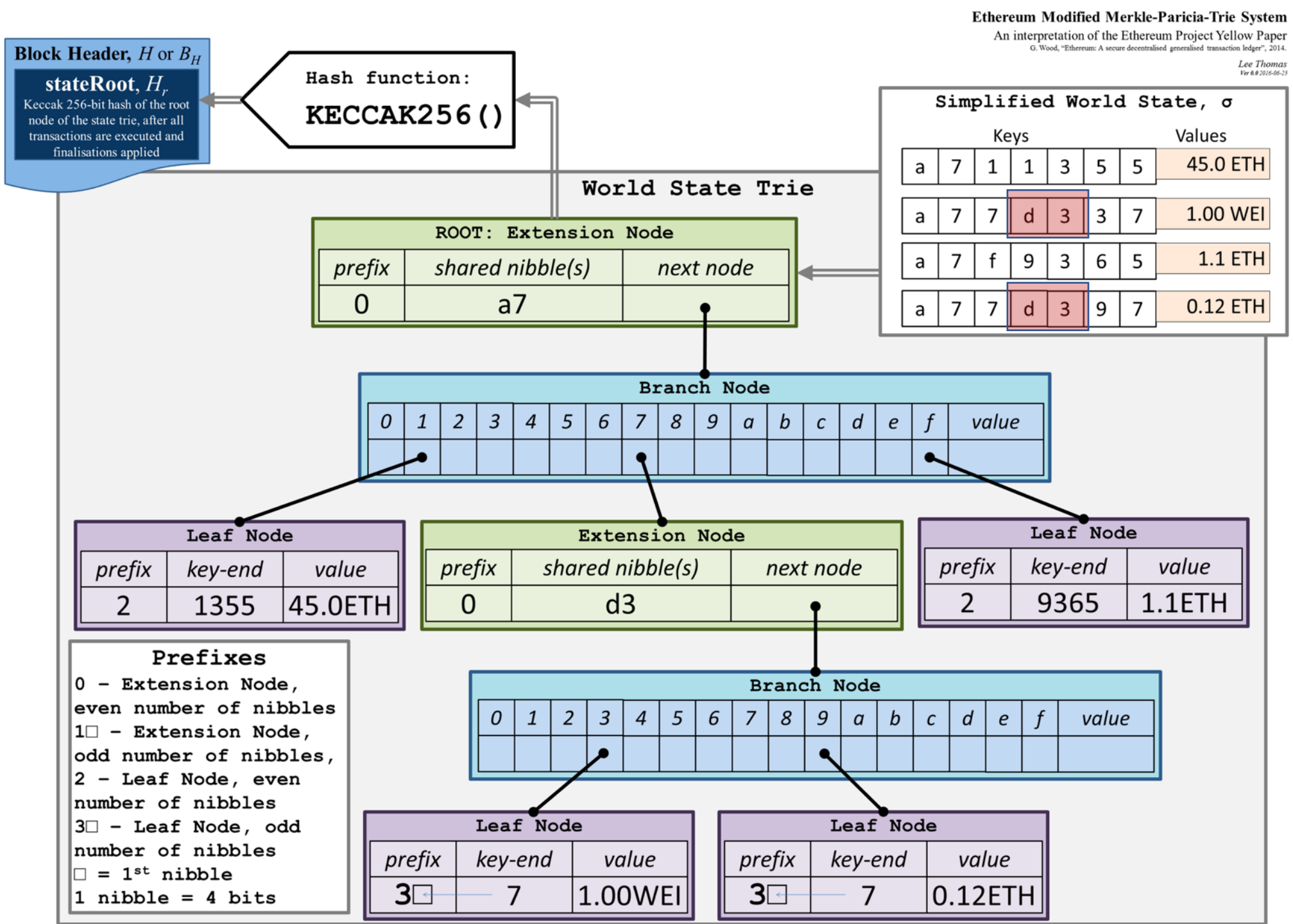
- [0x00, 0x7f]: byte
- [0x80, 0xbf]: string
- [0xc0, 0xff]: list

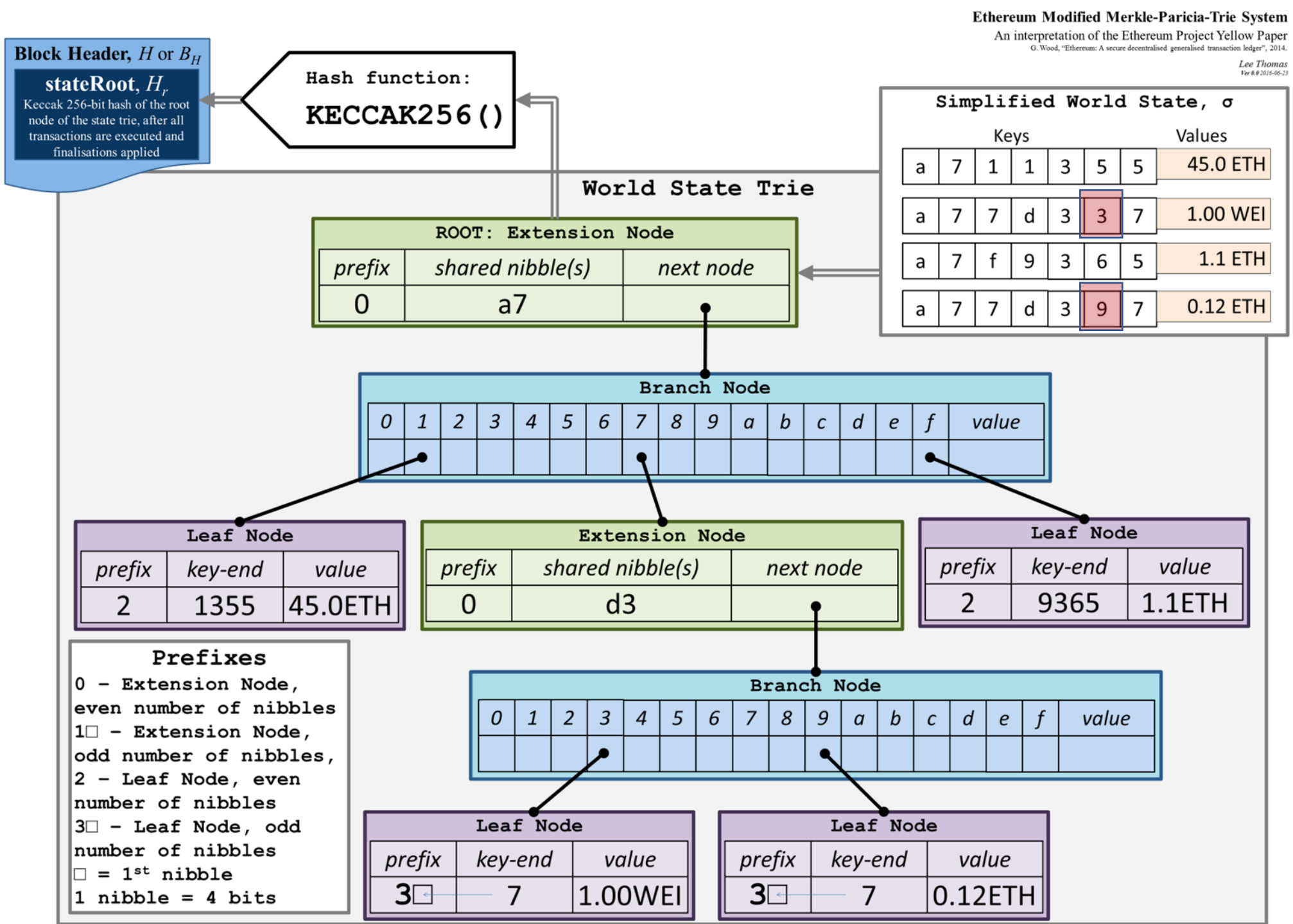


- The string “taiwan” 0x86 = 0x80 + 0x06
  - [0x86, 0x74, 0x61, 0x69, 0x77, 0x61, 0x6e]
- The list ['nccu', 'taiwan’] 0xcc = 0xc0 + 0x0c
  - [0xcc, 0x84, 0x6e, 0x63, 0x63, 0x75, 0x86, 0x74, 0x61, 0x69, 0x77, 0x61, 0x6e]
- The string with 256 “a”, “aaaaaaaa….aaaaa”
  - [0xb9, 0x01, 0x00, 0x61, …, 0x61] 0xb9 = 0xb7 + 0x02


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 45.0 ETH
a 7 7 d 3 3 7 1.00 WEI
a 7 f 9 3 6 5 1.1 ETH
a 7 7 d 3 9 7 0.12 ETH
World State Trie
ROOT: Extension Node
prefix
shared nibble(s)
next node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
2
1355
45.0ETH
Extension Node
prefix
shared nibble(s)
next node
0
d3
Leaf Node
prefix
key-end
value
2
9365
1.1ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
3□
7
1.00WEI
Leaf Node
prefix
key-end
value
3□
7
0.12ETH


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 45.0 ETH
a 7 7 d 3 3 7 1.00 WEI
a 7 f 9 3 6 5 1.1 ETH
a 7 7 d 3 9 7 0.12 ETH
World State Trie
ROOT: Extension Node
prefix
shared nibble(s)
next node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
2
1355
45.0ETH
Extension Node
prefix
shared nibble(s)
next node
0
d3
Leaf Node
prefix
key-end
value
2
9365
1.1ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
3□
7
1.00WEI
Leaf Node
prefix
key-end
value
3□
7
0.12ETH


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 45.0 ETH
a 7 7 d 3 3 7 1.00 WEI
a 7 f 9 3 6 5 1.1 ETH
a 7 7 d 3 9 7 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 45.0 ETH
a 7 7 d 3 3 7 1.00 WEI
a 7 f 9 3 6 5 1.1 ETH
a 7 7 d 3 9 7 0.12 ETH
World State Trie
ROOT: Extension Node
prefix
shared nibble(s)
next node
0
a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
2
1355
45.0ETH
Extension Node
prefix
shared nibble(s)
next node
0
d3
Leaf Node
prefix
key-end
value
2
9365
1.1ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix
key-end
value
3□
7
1.00WEI
Leaf Node
prefix
key-end
value
3□
7
0.12ETH

Smart contract

Greeter contract

```
contract mortal {
   /* Define variable owner of the type address*/
   address owner;

   /* this function is executed at initialization and sets the owner of the contract */
   function mortal() { owner = msg.sender; }

   /* Function to recover the funds on the contract */
   function kill() { if (msg.sender == owner) suicide(owner); }
}

contract greeter is mortal {
   /* define variable greeting of the type string */
   string greeting;
   /* this runs when the contract is executed */
   function greeter(string _greeting) public {
       greeting = _greeting;
   }

   /* main function */
   function greet() constant returns (string) {
      return greeting;
   }
}
```

https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.24+commit.e67f0147.js

Remix IDE

https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.24+commit.e67f0147.js

Web3 deploy code

```
var _greeting = "NCCU";
var greeterContract =
web3.eth.contract([{"constant":false,"inputs":[],"name":"kill","outputs":[],"payable":false,"st
ateMutability":"nonpayable","type":"function"},{"constant":true,"inputs":[],"name":"greet","out
puts":[{"name":"","type":"string"}],"payable":false,"stateMutability":"view","type":"function"}
,{"inputs":[{"name":"_greeting","type":"string"}],"payable":false,"stateMutability":"nonpayable
","type":"constructor"}]);
var greeter = greeterContract.new(
    _greeting,
    {
        from: web3.eth.accounts[0],
        data: '---------[BYTECODE]---------',
        gas: '4700000'
    }, function (e, contract){
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' +
contract.transactionHash);
    }
})
```

Deploy and get mined

1. geth attach /var/folders/x8/gs3w34_n4wx6cmrp35b_q4r40000gn/T/geth.ipc (geth)

geth (geth) ⌘1 | geth (geth) ⌘2

```
820380516001836020036101000a031916815260200191505b509250505060405180910390f35b60008090549061010000a900473ffffffffffffffffff
ffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffffffffffff163373ffffffffffffffffffffffffffffffffffffffff16141561
0187576000809054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16ff5
b565b606060018054600181600116156101000203166002900480601f016020809104026020016040519081016040528092919081815260200182805460
60018160011615610100020316600290048015610221578060 1f106101f657610100808354040283529160200191610221565b820191906000526020600
0020905b81548152906001019060200180831161020457829003601f168201915b505050505090505090560a165627a7a723058205aa75599fe31344a
4b01e86e9fa09a234ec5fd56ea5d31c48ba38b138cc4ab340029',
......      gas: '4700000'
......    }, function (e, contract){
......     console.log(e, contract);
......     if (typeof contract.address !== 'undefined') {
.........          console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' + contract.transacti
onHash);
.........     }
......  })
null [object Object]
undefined
> null [object Object]
Contract mined! address: 0xfbd8e2427c9dcfe4df275d24615d850e8942900d transactionHash: 0xa570a60431fad2e1b0f8be357764e92afb
5dfcb6a471133e2885fbbd09b6c04a

> _
```

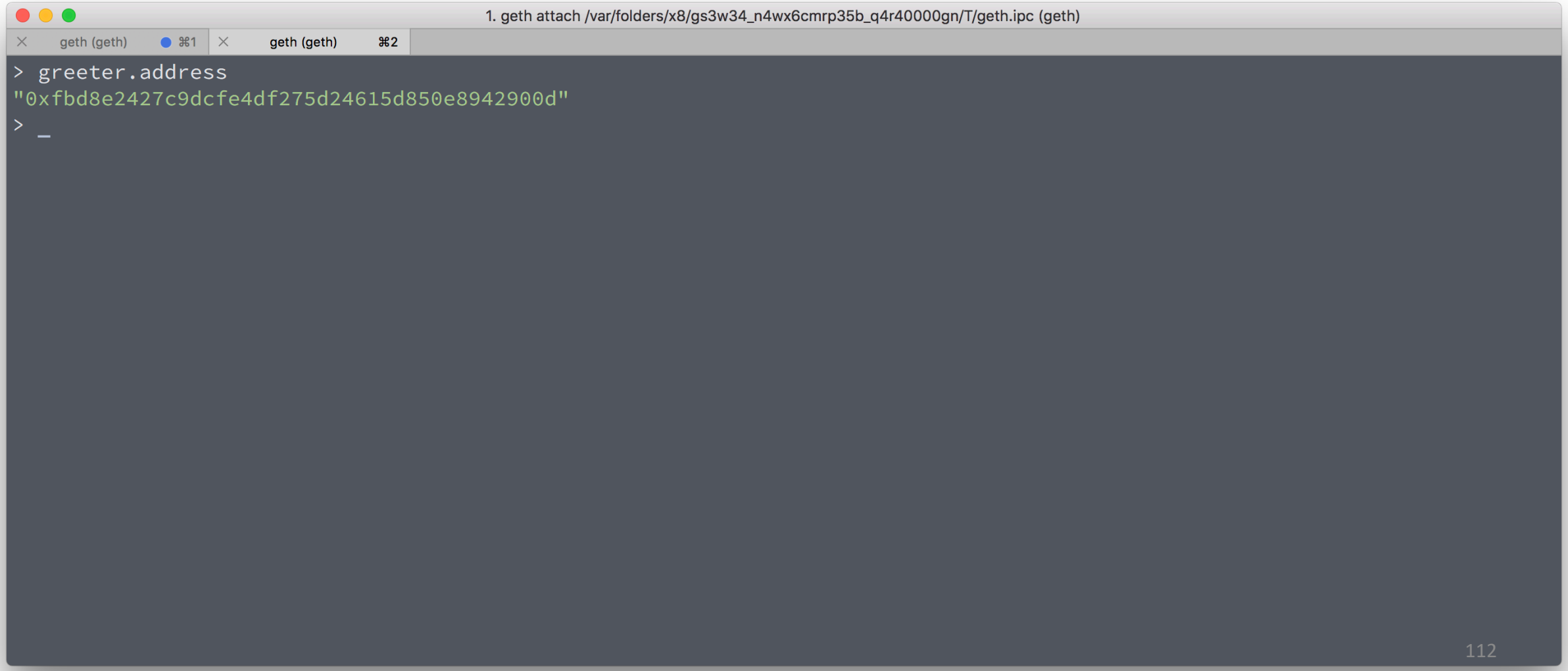
Contract ABI

1. geth attach /var/folders/x8/gs3w34_n4wx6cmrp35b_q4r40000gn/T/geth.ipc (geth)

geth (geth) ⌘1 | geth (geth) ⌘2

```
> greeter.abi
[{
    constant: false,
    inputs: [],
    name: "kill",
    outputs: [],
    payable: false,
    stateMutability: "nonpayable",
    type: "function"
}, {
    constant: true,
    inputs: [],
    name: "greet",
    outputs: [{
        name: "",
        type: "string"
    }],
    payable: false,
    stateMutability: "view",
    type: "function"
}, {
    inputs: [{
```

Contract address



Greeter contract

1. 有幾個合約函數?
2. 那個要花 gas 那個不用?

```
contract mortal {
    /* Define variable owner of the type address*/
    address owner;

    /* this function is executed at initialization and sets the owner of the contract */
    function mortal() { owner = msg.sender; }

    /* Function to recover the funds on the contract */
    function kill() { if (msg.sender == owner) suicide(owner); }
}

contract greeter is mortal {
    /* define variable greeting of the type string */
    string greeting;
    /* this runs when the contract is executed */
    function greeter(string _greeting) public {
        greeting = _greeting;
    }

    /* main function */
    function greet() constant returns (string) {
        return greeting;
    }
}
```

Greeter contract

```
contract mortal {
    /* Define variable owner of the type address*/
    address owner;

    /* this function is executed at initialization and sets the owner of the contract */
    function mortal() { owner = msg.sender; }

    /* Function to recover the funds on the contract */
    function kill() { if (msg.sender == owner) suicide(owner); }
}

contract greeter is mortal {
    /* define variable greeting of the type string */
    string greeting;
    /* this runs when the contract is executed */
    function greeter(string _greeting) public {
        greeting = _greeting;
    }

    /* main function */
    function greet() constant returns (string) {
        return greeting;
    }
}
```

Call contract function

1. geth attach /var/folders/x8/gs3w34_n4wx6cmrp35b_q4r40000gn/T/geth.ipc (geth)

geth (geth) ⌘1 | geth (geth) ⌘2

```
> greeter.greet()
"NCCU"
> greeter.kill.sendTransaction({from: eth.coinbase})
"0xe9719c3b7c92070129c0576d7735229185 40d79cf180157f0aa2bf5d5df26b81"
> greeter.greet()
Error: new BigNumber() not a base 16 number:
    at L (bignumber.js:3:2876)
    at bignumber.js:3:8435
    at a (bignumber.js:3:389)
    at web3.js:1110:23
    at web3.js:1634:20
    at web3.js:826:16
    at map (<native code>)
    at web3.js:825:12
    at web3.js:4080:18

> _
```

Contract life cycle

1. 編譯合約

Compile contract to **bytecode**

2. 部署合約

```
sendTransaction({
        from: eth.coinbase,
        data: bytecode,
        gas: 470000
})
```

3. 呼叫合約函數

```
sendTransaction({
        from: eth.coinbase,
        to: contarct_addr,
        data: encode(function),
        gas: 50000
})
```

4. 摧毀合約

```
sendTransaction({
        from: eth.coinbase,
        to: contarct_addr,
        data: encode(kill),
        gas: 50000
})
```

Contract life cycle

1. 編譯合約

Compile contract to **bytecode**

2. 部署合約

```
sendTransaction({
        from: eth.coinbase,
        data: bytecode,
        gas: 470000
})
```

Contract-creating TX

3. 呼叫合約函數

```
sendTransaction({
        from: eth.coinbase,
        to: contarct_addr,
        data: encode(function),
        gas: 50000
})
```

Message call

4. 摧毀合約

```
sendTransaction({
        from: eth.coinbase,
        to: contarct_addr,
        data: encode(kill),
        gas: 50000
})
```

Message call

Contract life cycle

1. 編譯合約

Compile contract to **bytecode**

2. 部署合約

```
sendTransaction({
        from: eth.coinbase,
        data: bytecode,
        gas: 470000
})
```

3. 呼叫合約函數

```
sendTransaction({
        from: eth.coinbase,
        to: contarct_addr,
        data: encode(function),
        gas: 50000
})
```

4. 摧毀合約

```
sendTransaction({
        from: eth.coinbase,
        to: contarct_addr,
        data: encode(kill),
        gas: 50000
})
```

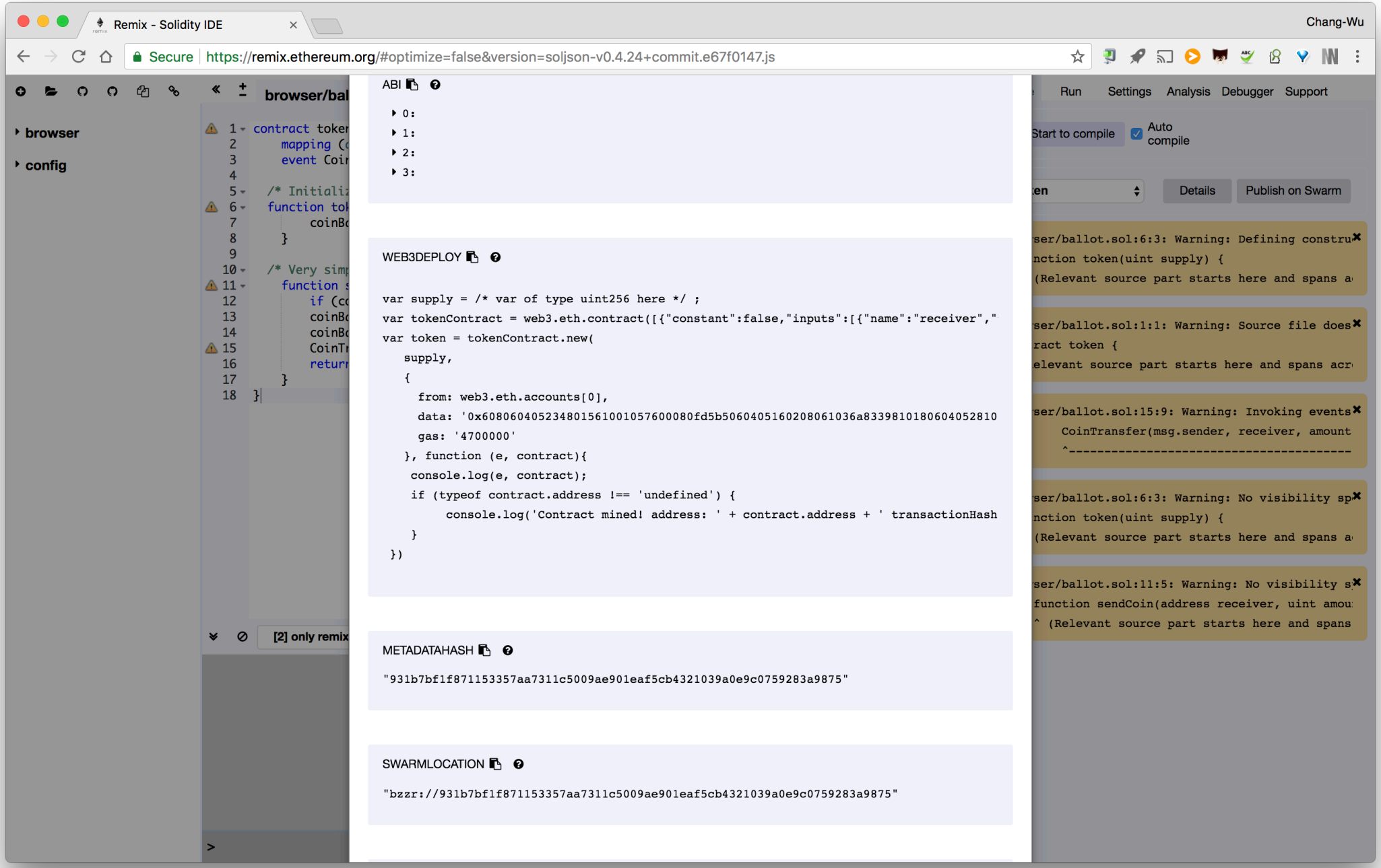
1. 怎麼知道跟那個合約互動?
2. 怎麼知道合約有哪些函數可以呼叫?
3. 誰可以跟合約互動?
4. 交易的執行都會花費 gas 嗎?
5. 如何估算交易執行要送的 gas?

Coin contract

```
contract token {
    mapping (address => uint) public coinBalanceOf;
    event CoinTransfer(address sender, address receiver, uint amount);

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function token(uint supply) {
        coinBalanceOf[msg.sender] = supply;
    }

    /* Very simple trade function */
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
        if (coinBalanceOf[msg.sender] < amount) return false;
        coinBalanceOf[msg.sender] -= amount;
        coinBalanceOf[receiver] += amount;
        CoinTransfer(msg.sender, receiver, amount);
        return true;
    }
}
```


Remix - Solidity IDE
Chang-Wu
Secure https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.24+commit.e67f0147.js
browser
config
Run Settings Analysis Debugger Support
Start to compile
Auto compile
Details
Publish on Swarm
ABI
0:
1:
2:
3:
WEB3DEPLOY
var supply = /* var of type uint256 here */ ;
var tokenContract = web3.eth.contract([{"constant":false,"inputs":[{"name":"receiver","
var token = tokenContract.new(
supply,
{
from: web3.eth.accounts[0],
data: '0x6080604052348015610010576000080fd5b506040516020806103a833981018060405281
gas: '4700000'
}, function (e, contract){
console.log(e, contract);
if (typeof contract.address !== 'undefined') {
console.log('Contract mined! address: ' + contract.address + ' transactionHash
}
})
METADATAHASH
"931b7bf1f871153357aa7311c5009ae901eaf5cb4321039a0e9c0759283a9875"
SWARMLOCATION
"bzzr://931b7bf1f871153357aa7311c5009ae901eaf5cb4321039a0e9c0759283a9875"

Web3 deploy code

```
var supply = 10000;
var tokenContract =
web3.eth.contract([{"constant":false,"inputs":[{"name":"receiver","type":"address"},{"name":"amount","ty
pe":"uint256"}],"name":"sendCoin","outputs":[{"name":"sufficient","type":"bool"}],"payable":false,"state
Mutability":"nonpayable","type":"function"},{"constant":true,"inputs":[{"name":"","type":"address"}],"na
me":"coinBalanceOf","outputs":[{"name":"","type":"uint256"}],"payable":false,"stateMutability":"view","t
ype":"function"},{"inputs":[{"name":"supply","type":"uint256"}],"payable":false,"stateMutability":"nonpa
yable","type":"constructor"},{"anonymous":false,"inputs":[{"indexed":false,"name":"sender","type":"addre
ss"},{"indexed":false,"name":"receiver","type":"address"},{"indexed":false,"name":"amount","type":"uint2
56"}],"name":"CoinTransfer","type":"event"}]);
var token = tokenContract.new(
    supply,
    {
        from: web3.eth.accounts[0],
        data: '---------[BYTECODE]---------',
        gas: '4700000'
    }, function (e, contract){
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' +
contract.transactionHash);
    }
})
```

bytecode

1. geth attach /var/folders/x8/gs3w34_n4wx6cmrp35b_q4r40000gn/T/geth.ipc (geth)

geth (geth) ⌘1 | geth (geth) ⌘2

```
> eth.getCode(token.address)
"0x6080604052600436106100c576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff16806390b98a
11146100515780633bbd39ac0146100b6575b600080fd5b34801561005d57600080fd5b5061009c600480360381019080803573ffffffffffffffffffff
ffffffffffffffffffff1690602001909291908035906020019092919050505061010d565b604051808215151515815260200191505060405180910390
90f35b3480156100c257600080fd5b506100f7600480360381019080803573ffffffffffffffffffffffffffffffffffffffff169060200190929190505
050506102a0565b604051808281526020019150506040518091039035b6000816000803373ffffffffffffffffffffffffffffffffffffffff1673ff
ffffffffffffffffffffffffffffffffffff16815260200190815260200160002054101561015e576000905061029a565b816000803373ffffffffff
ffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002060008282540392505081905550816000808573ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002060008282540192505081905550 7f16cdf1707799c6655baac6e210f52b94b7cec08adcaf9ede7dfe8649da926146338484604051808473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200183 73ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200182815260200193505050506040518091039a160019050505b92915050565b60006020528060005260406000206000915090505481560 0a165627a7a72305820931b7bf1f871153357aa7311c5009ae901eaf5cb4321039a0e9c0759283a98750029"
> token.address
"0x6966caf57755ef380aea6b3fa6885ed48f00c1ed"
> _
```

Coin contract function

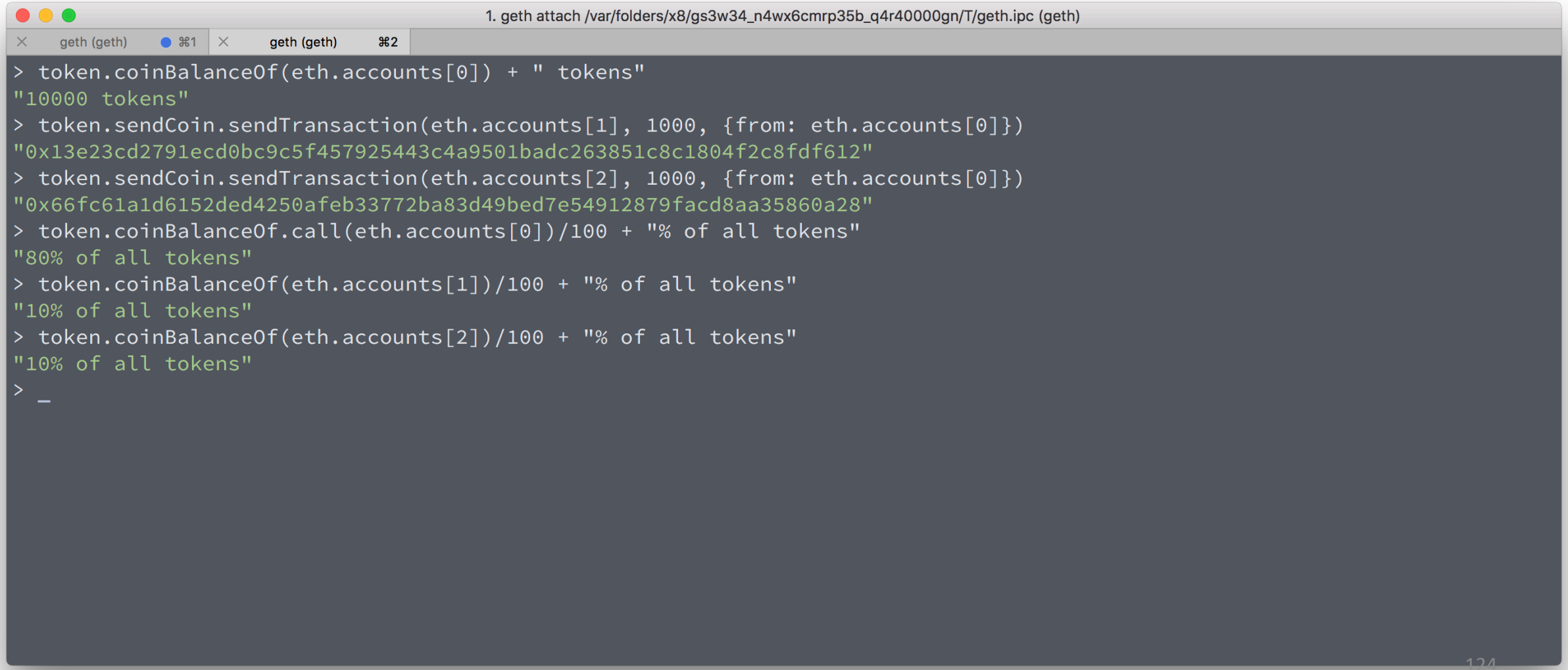
```
contract token {
    mapping (address => uint) public coinBalanceOf;
    event CoinTransfer(address sender, address receiver, uint amount);

    /* Initializes contract with initial supply tokens to the creator of the contract */
    function token(uint supply) {
        coinBalanceOf[msg.sender] = supply;
    }

    /* Very simple trade function */
    function sendCoin(address receiver, uint amount) returns(bool sufficient) {
        if (coinBalanceOf[msg.sender] < amount) return false;
        coinBalanceOf[msg.sender] -= amount;
        coinBalanceOf[receiver] += amount;
        CoinTransfer(msg.sender, receiver, amount);
        return true;
    }
}
```

https://github.com/ethereum/go-ethereum/wiki/Contract-Tutorial

sendTransaction



Logs

- 用來追蹤交易的狀態與訊息

- Log 包含
  - 合約地址
  - 合約內部事件(event) 訂閱的主題 (topic)
  - 與 event 相關的 data