Ethereum protocol

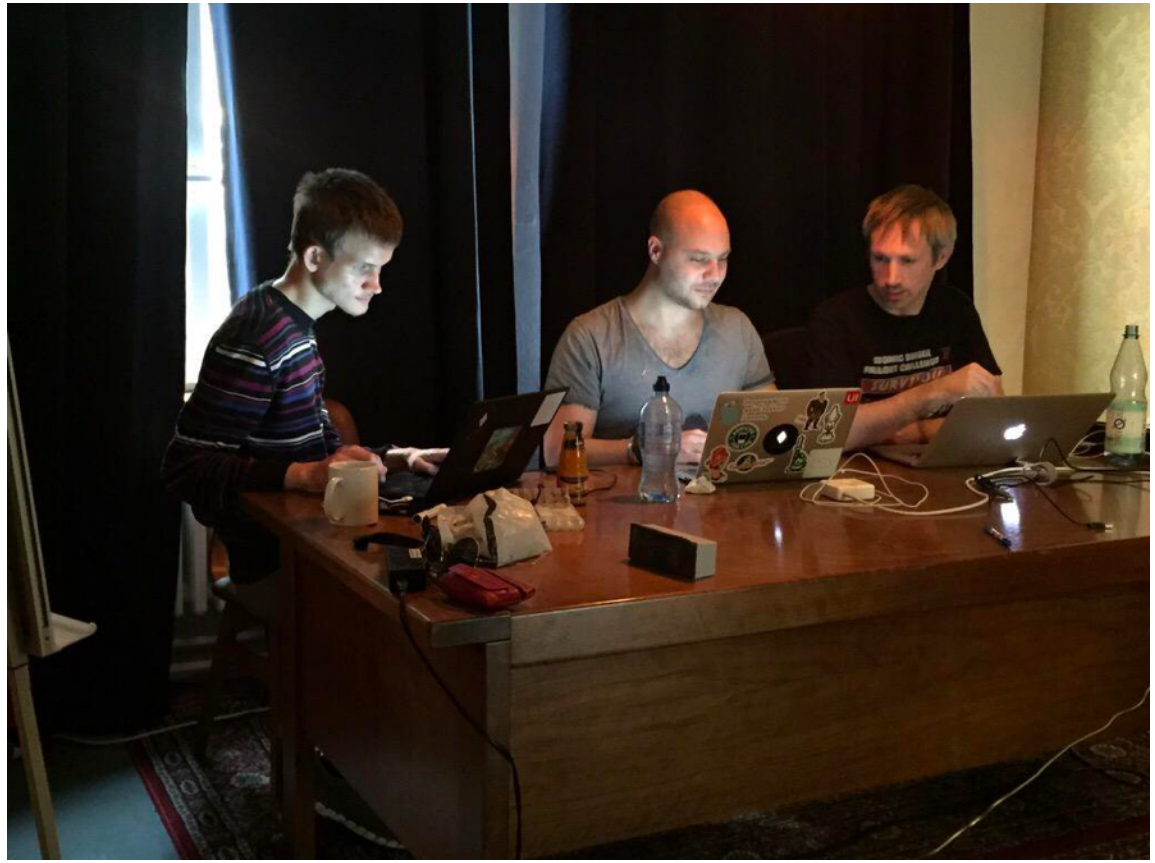
以太坊協議

ethereum

Co-Founder

- Ethereum was proposed in late 2013 by Vitalik Buterin, a cryptocurrency researcher and programmer.

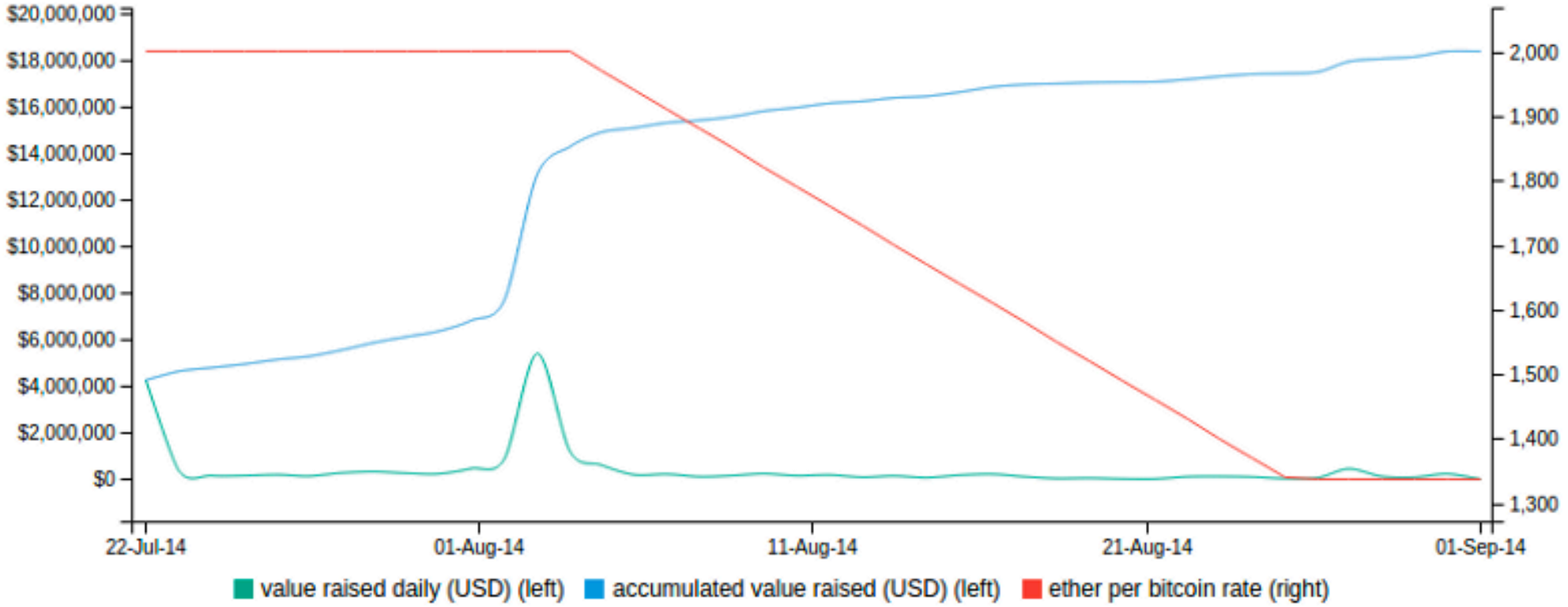
The Ethereum Launch Process



https://vitalik.ca/

Crowdsale

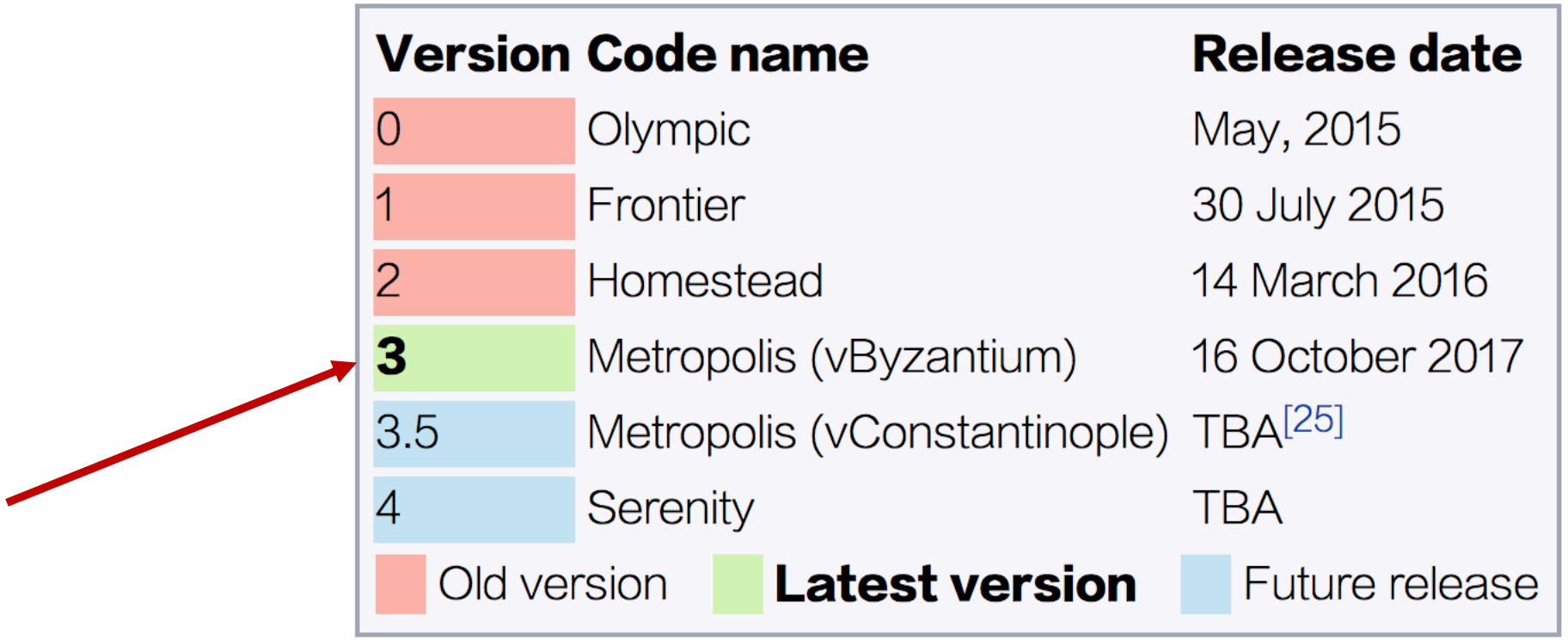
- Run for 42 days, between Jul-Sep in 2014.
- The sale price was 2000 ETH for 1 BTC for the first 14 days, and then started increasing linearly, finishing at 1337 ETH for 1 BTC.



https://en.wikipedia.org/wiki/Ethereum

Ethereum

- The system went live on 30 July 2015



Version	Code name	Release date
0	Olympic	May, 2015
1	Frontier	30 July 2015
2	Homestead	14 March 2016
3	Metropolis (vByzantium)	16 October 2017
3.5	Metropolis (vConstantinople)	TBA[25]
4	Serenity	TBA

Old version | **Latest version** | Future release

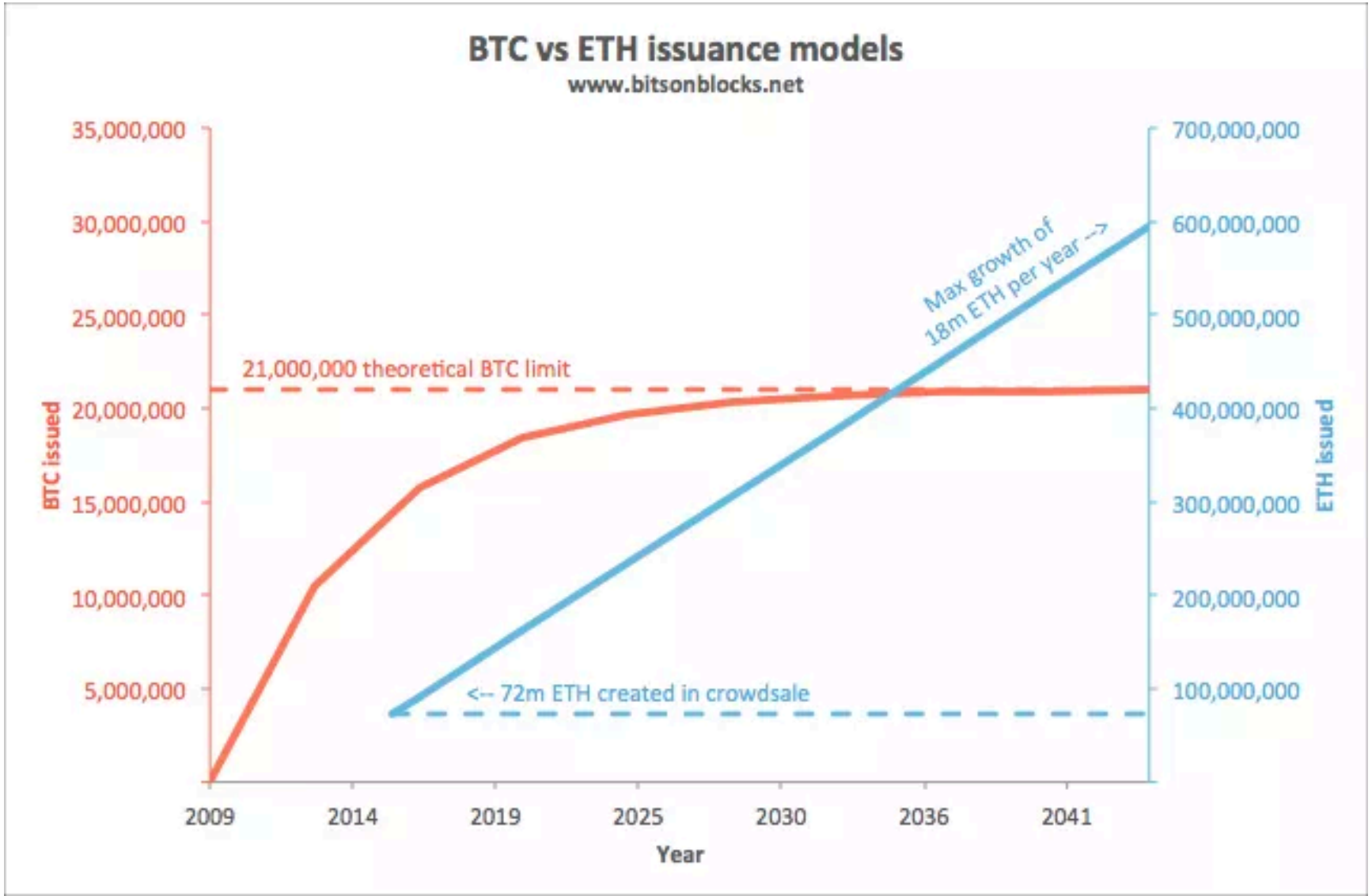
Ethereum

- 以太坊基金會, 瑞士
- 柏林, 新加坡

Cryptocurrency	Ether
Ticker symbol	ETH
Timestamping scheme	Proof-of-work (Ethash)
Hash function	Keecak
Issuance	Block and Uncle/Ommer reward
Block reward	3 ETH → 2 ETH
Block time	14-15 seconds on average
Block explorer	https://etherscan.io/ https://www.etherchain.org/

https://bitsonblocks.net/2016/10/02/gentle-introduction-ethereum/

ETH issuance model



ETH issuance model (Early stage)

Total ETH = Pre-mine + Block rewards + Uncle rewards + Uncle inclusion reward

- Pre-mine: 以太坊正式運營之間，預先產出7200萬個以太幣，之後限制每年的生成以不超過四分之一的預先發行量，即1800萬個以太幣
- Block reward: 目前每一個區塊獎勵是5個以太幣，一個區塊的生成速度約14秒，換言之，一年相當於有225萬個區塊 (365x24x60x60/14)，1125萬個以太幣生成
- Uncle reward: 在以太坊中，有些區塊在同一時間被其他礦工所發現，但卻不是接在主鏈上，稱之為Uncle區塊，相當於比特幣中的孤兒區塊(Orphan block)。這類區塊可能被之後的區塊所參照，此外這類區塊的獎勵為正常獎勵的八分之七，即5x(7/8)=4.375個以太幣。依據Uncle區塊的產生速率，一年約有接近700萬個以太幣生成
- Uncle inclusion reward: 打包 Uncle block 的礦工，可得現有以太幣的 1/32。

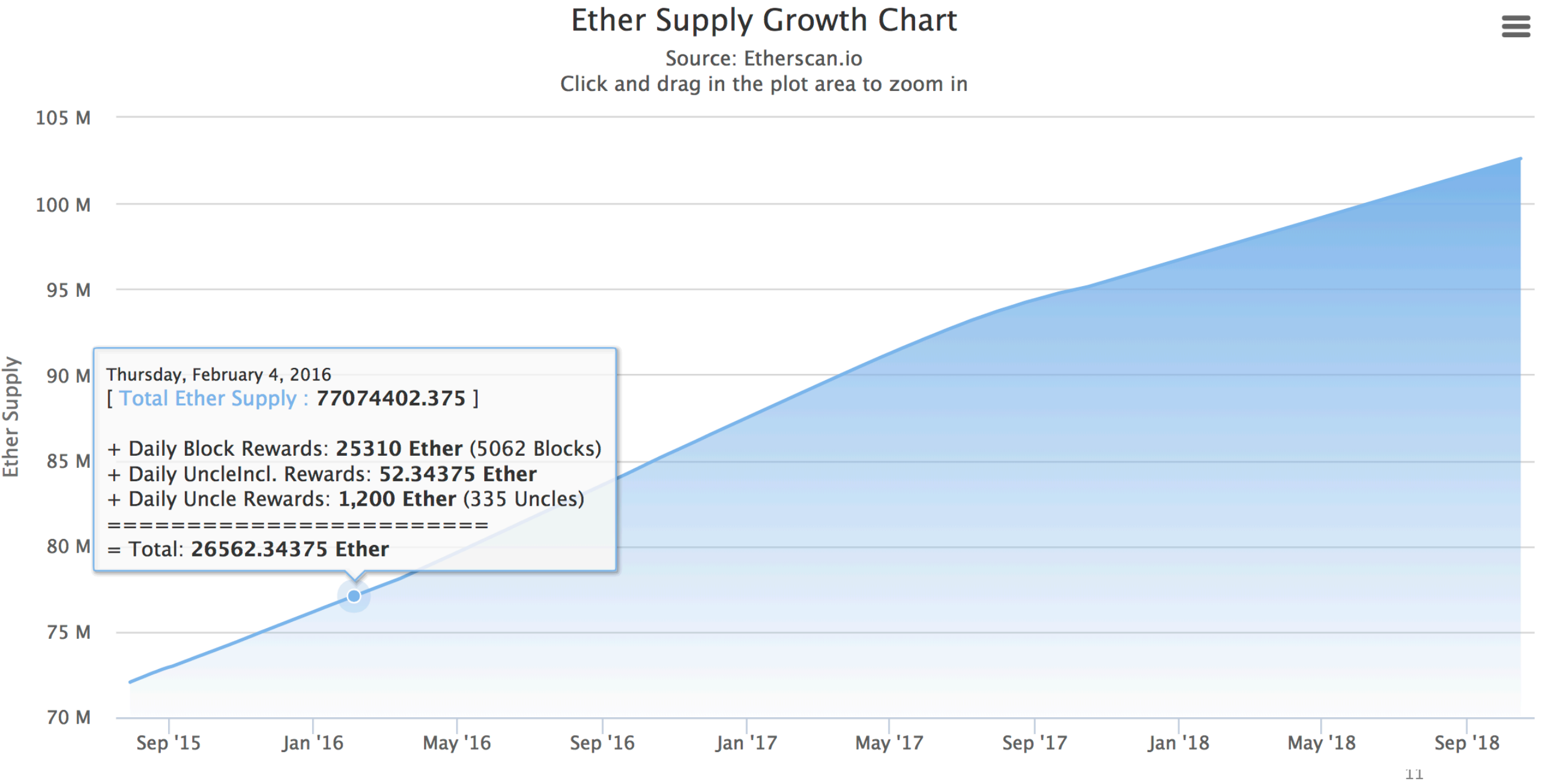
Current issuance

- Total ETH = Pre-mine + Block rewards + Uncle rewards + Uncle inclusion reward

- Block reward: 目前每一個區塊獎勵是 2 個以太幣，一個區塊的生成速度約 12 秒
- Uncle reward: 沒進入到主鏈的合法區塊
  - (Uncle Number + 8 - Block Number) * Miner's Reward / 8
  - Example: https://etherscan.io/block/8570623
  - 2.2401790918 Ether (**2** + **0.1776790918** + **0.0625**)
    - 1 Uncle block
- Uncle inclusion reward: 打包 Uncle block 的礦工，可得現有以太幣的 2*1/32

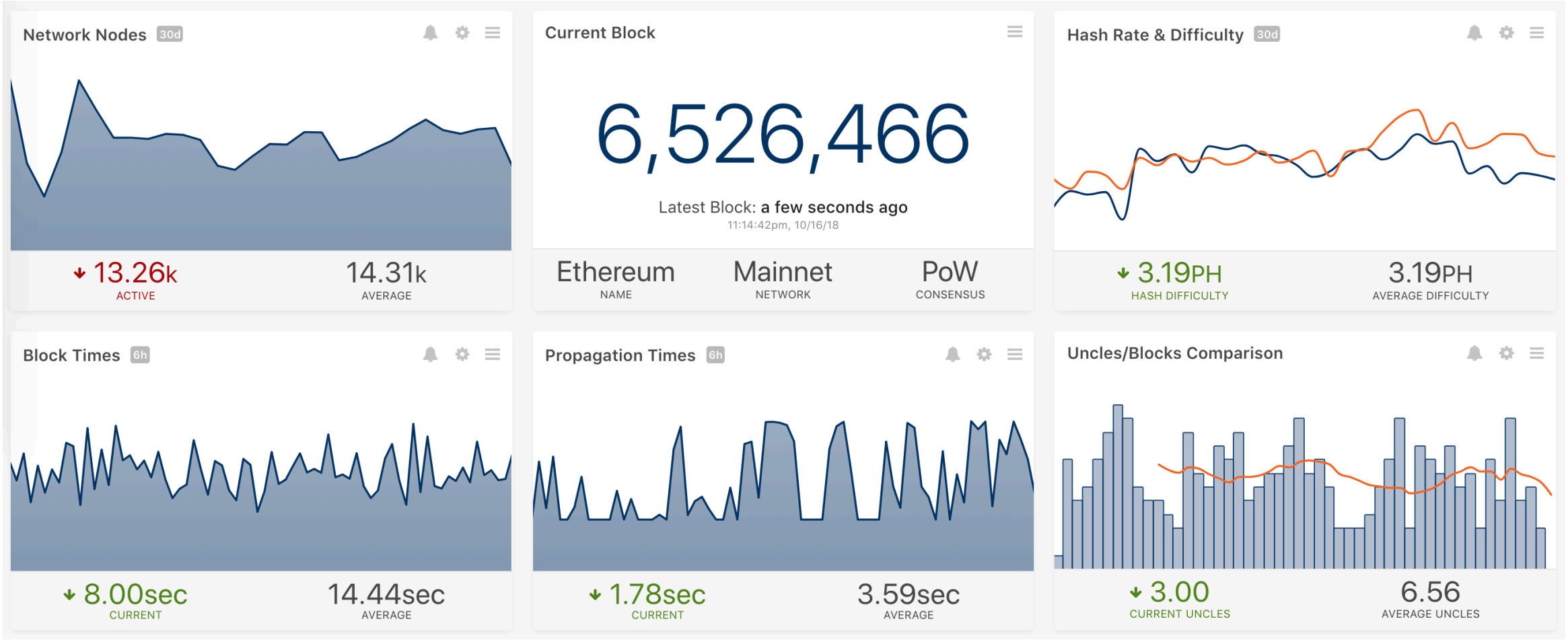
- 我是礦工，我可以打包盡可能多的 uncle block 嗎？
- 為什麼會有 uncle block？
- 我什麼時候會收到 reward？

https://etherscan.io/chart/ethersupplygrowth



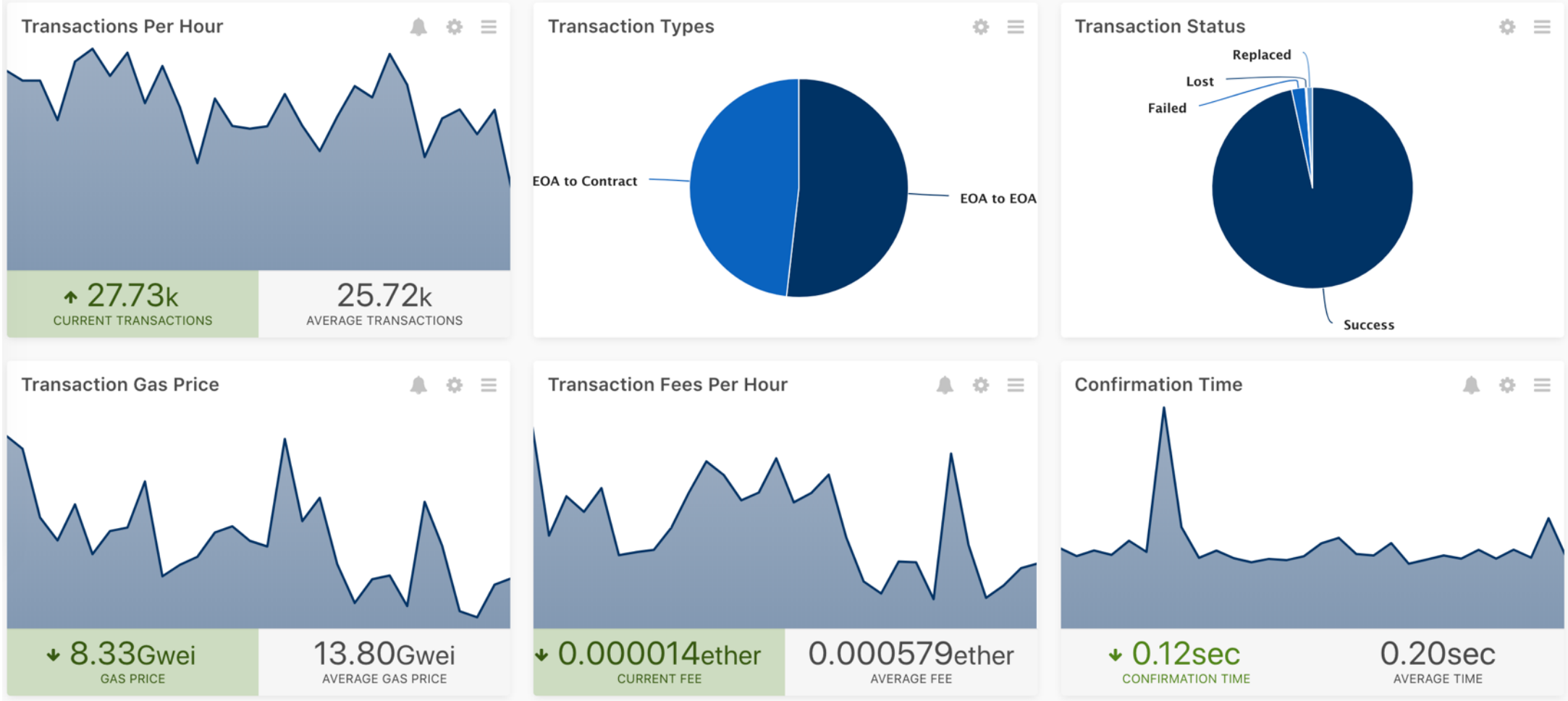
https://amberdata.io/dashboards/infrastructure

Real-time chart



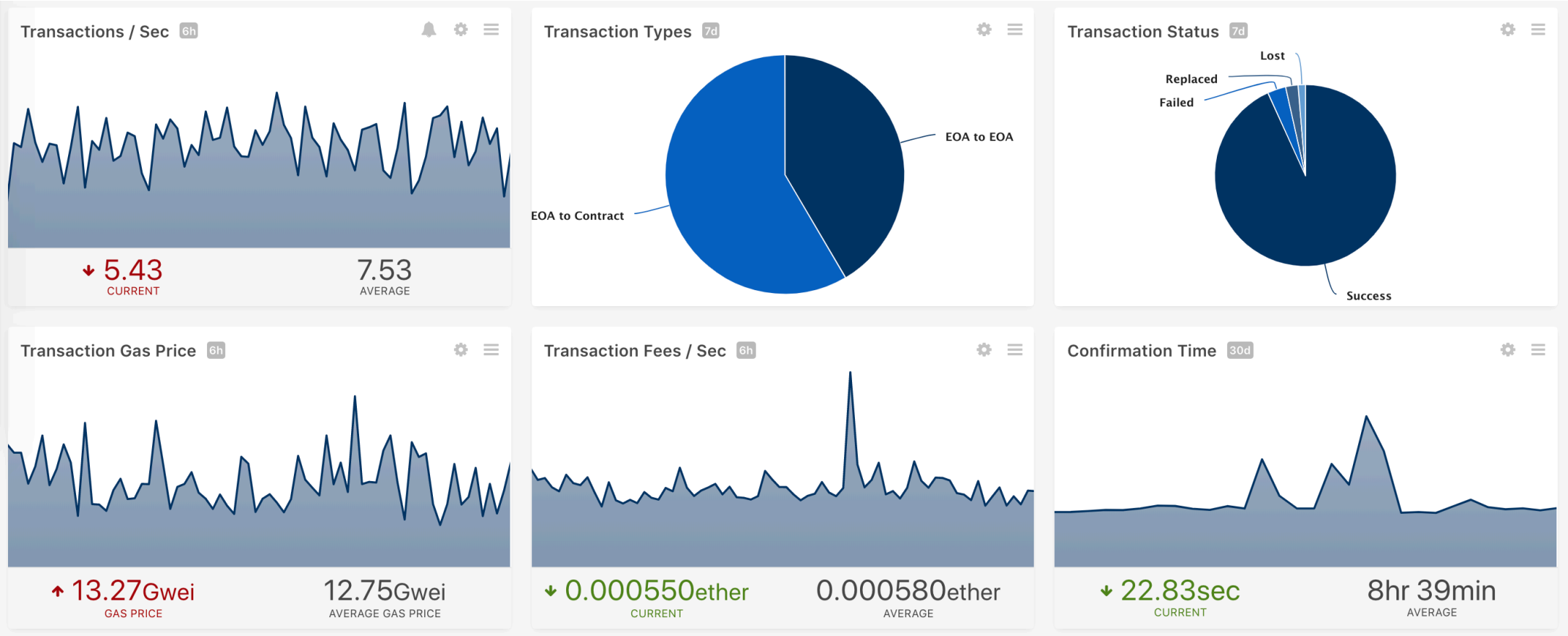
https://amberdata.io/dashboards/transactions

Real-time chart (04/13)

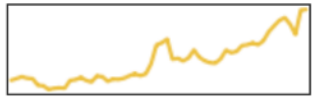


https://amberdata.io/dashboards/transactions

Real-time chart (10/16)



Market Cap (04/13)

1	Bitcoin	$128,855,294,369	$7,591.63	$8,128,550,000	16,973,337 BTC	9.46%	
2	Ethereum	$45,827,548,286	$463.96	$2,223,630,000	98,775,209 ETH	10.17%	
3	Ripple	$21,559,650,761	$0.551475	$1,003,880,000	39,094,520,623 XRP *	10.66%	
4	Bitcoin Cash	$12,047,639,672	$705.81	$377,630,000	17,069,288 BCH	7.84%	
5	EOS	$6,888,240,014	$8.73	$2,449,500,000	788,650,607 EOS *	22.26%	
6	Litecoin	$6,792,943,416	$121.20	$488,687,000	56,049,238 LTC	5.20%	
7	Cardano	$4,946,599,861	$0.190789	$417,519,000	25,927,070,538 ADA *	19.50%	
8	Stellar	$4,105,206,958	$0.221290	$94,621,700	18,551,253,819 XLM *	8.48%	
9	NEO	$3,941,782,000	$60.64	$299,360,000	65,000,000 NEO *	9.57%	

Market Cap (10/16)

1	Bitcoin	$114,120,630,871	$6,586.94	$4,262,774,091	17,325,300 BTC	-0.95%	···
2	Ethereum	$21,541,732,106	$209.94	$1,671,918,175	102,607,647 ETH	0.13%	···
3	XRP	$18,158,558,751	$0.453991	$579,573,353	39,997,634,397 XRP *	2.44%	···
4	Bitcoin Cash	$7,945,305,505	$456.48	$317,342,590	17,405,463 BCH	-0.70%	···
5	EOS	$4,909,679,793	$5.42	$447,487,941	906,245,118 EOS *	-0.68%	···
6	Stellar	$4,253,582,997	$0.225163	$42,737,692	18,891,162,191 XLM *	0.26%	···
7	Litecoin	$3,225,043,903	$54.89	$289,043,006	58,750,702 LTC	-0.31%	···
8	Tether	$2,210,951,893	$0.979849	$2,880,043,971	2,256,421,736 USDT *	1.54%	···
9	Cardano	$1,957,754,417	$0.075510	$27,064,310	25,927,070,538 ADA *	1.11%	···

Community

ASSETH

Enterprise Ethereum Alliance

ENTERPRISE ETHEREUM ALLIANCE

- 與 Ethereum Foundation 沒有任何關係

Background

- Public chain, any one can become the miner (PoW)
  - Ethereum Foundation
  - Parity Technologies
  - ConsenSys

- Consortium chain, only the permissioned member can propose block
  - Enterprise Ethereum Alliance
  - https://media.consensys.net/what-the-eea-and-hyperledger-collaboration-means-for-enterprise-blockchain-development-31580012cb2

https://web.archive.org/web/20150627031414/http://vbuterin.com/ultimatescripting.html

Why Ethereum?

Ultimate Scripting: A Platform for Generalized Financial Contracts on Mastercoin

0.1. Introduction

Perhaps the key advantage of Mastercoin over the raw Bitcoin protocol is the potential to include much more advanced transaction types, including transactions that specify behavior based on future information well off into the future. For example, Mastercoin joins Ripple in being one of the only two major cryptocurrency networks that include the ability for users to make binding exchange offers as a type of transaction. From there, the Mastercoin Foundation intends to integrate even more complex contracts, including bets, contracts for difference and on-blockchain dice rolls. However, up until this point Mastercoin has been taking a relatively unstructured process in developing these idea, essentially treating each one as a separate "feature" with its own transaction code and rules. This document outlines an alternative way of specifying Mastercoin contracts which follows an open-ended philosophy, specifying only the basic data and arithmetic building blocks and allowing anyone to craft arbitrarily complex Mastercoin contracts to suit their own needs, including needs which we may not even anticipate.

0.2. Specification

The underlying idea behind this specification is to allow anyone to create a contract which pays out according to an arbitrary formula. The formula will be defined in a Bitcoin-like stack-based scripting language, consisting of numbers and opcodes.

The evaluation algorithm is as follows:

```
dataStack = []
opStack = script
while len(opStack) > 0:
    var op = opStack.pop()
    if typeof(op) == 'opcode': eval(dataStack,op)
    else: dataStack.push(op)
return dataStack.pop()
```

Where `eval` is defined for each opcode below. Any error (eg. division by zero) will make the script return `FAIL`, and result in the entire transaction being treated as invalid by the Mastercoin network. All variables will be signed 64-bit integers, and all arithmetic operations wrap around (that is, if the underlying arithmetic operation returns `R`, the value pushed is `((R + 2^63) % 2^64) - 2^63`.

For simplicity, we define `ds[-1]` to be the topmost value on the dataStack, `ds[-2]` the second topmost, etc. The opcodes are in two categories: arithmetic operations and data operations. Arithmetic operations are:

- `SWAP`: Swaps `ds[-1]` and `ds[-2]` in place
- `DUP`: Pushes a copy of `ds[-1]` onto the stack, thereby duplicating it
- `DROP`: Pops `ds[-1]` and does nothing
- `ADD`: Pops `ds[-1]` and `ds[-2]` and pushes `ds[-1] + ds[-2]`

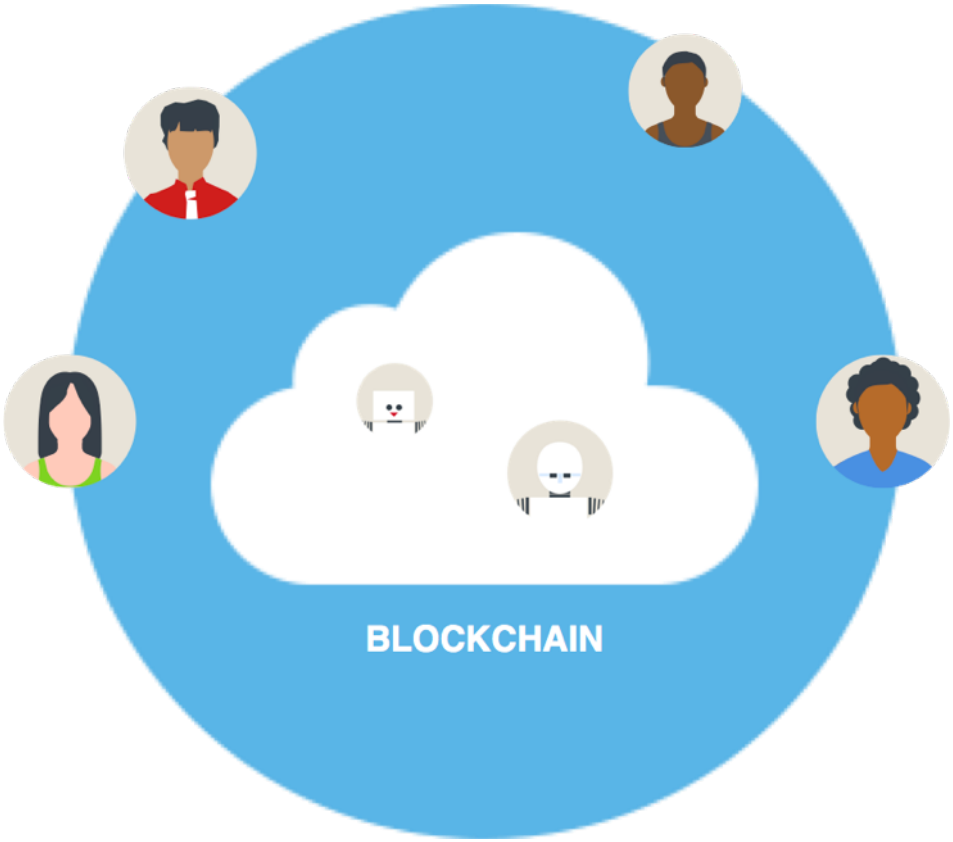
區塊鏈

- 起於中本聰的白皮書 (2008)
- 比特幣系統上線，被設計作為電子貨幣 (2009)
- 白皮書中沒有區塊鏈一詞
- 區塊鏈 (Blockchain) 技術被提出 (201X)
- 以太坊 (Ethereum) 被提出 (2013)
- 以太坊 ICO (Initial Coin Offering) (2014)
- 以太坊上線，智能合約的概念 (2015)
- ICO 盛行 (2017)

以太幣

Why Ethereum?

- 受比特幣啟發 (電子貨幣)
- 提供一個分散式應用平台 (區塊鏈為底)
- 智能合約 (DApp)
  - Token is one of the DApp



什麼是智能合約

- 由程式碼直接控制數字資產或業務流程
  - 程式碼對合約具有相同理解
  - 不因合約多次更迭而有所混淆
  - 對外部資料看法一致 (油價或股價等)
- 智能不是指人工智能
- 智能合約(目前)不是法律合約
- 程式碼僅執行複雜的規則和條件，不具有學習能力
  - 電梯
  - 販賣機

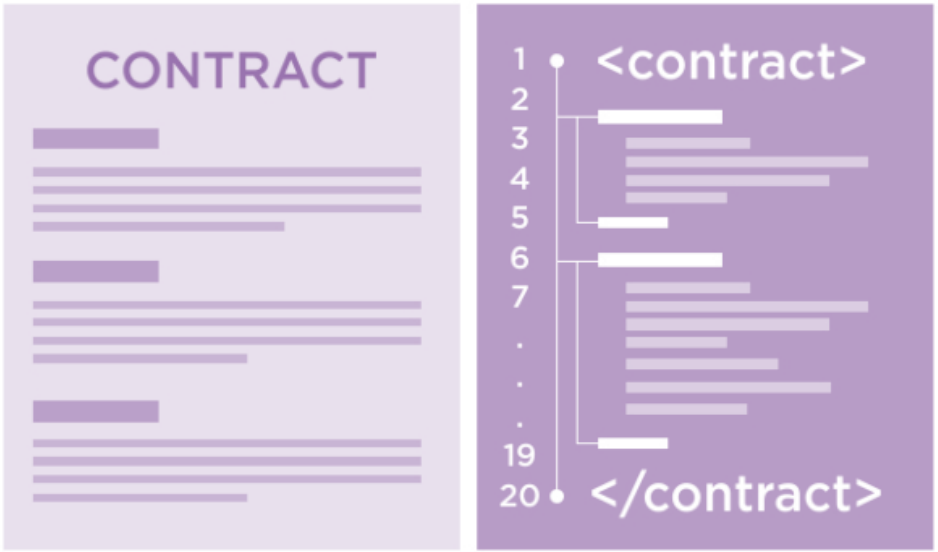


Image: https://bitsonblocks.net

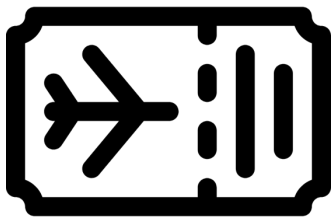
販賣機

飛機延誤險

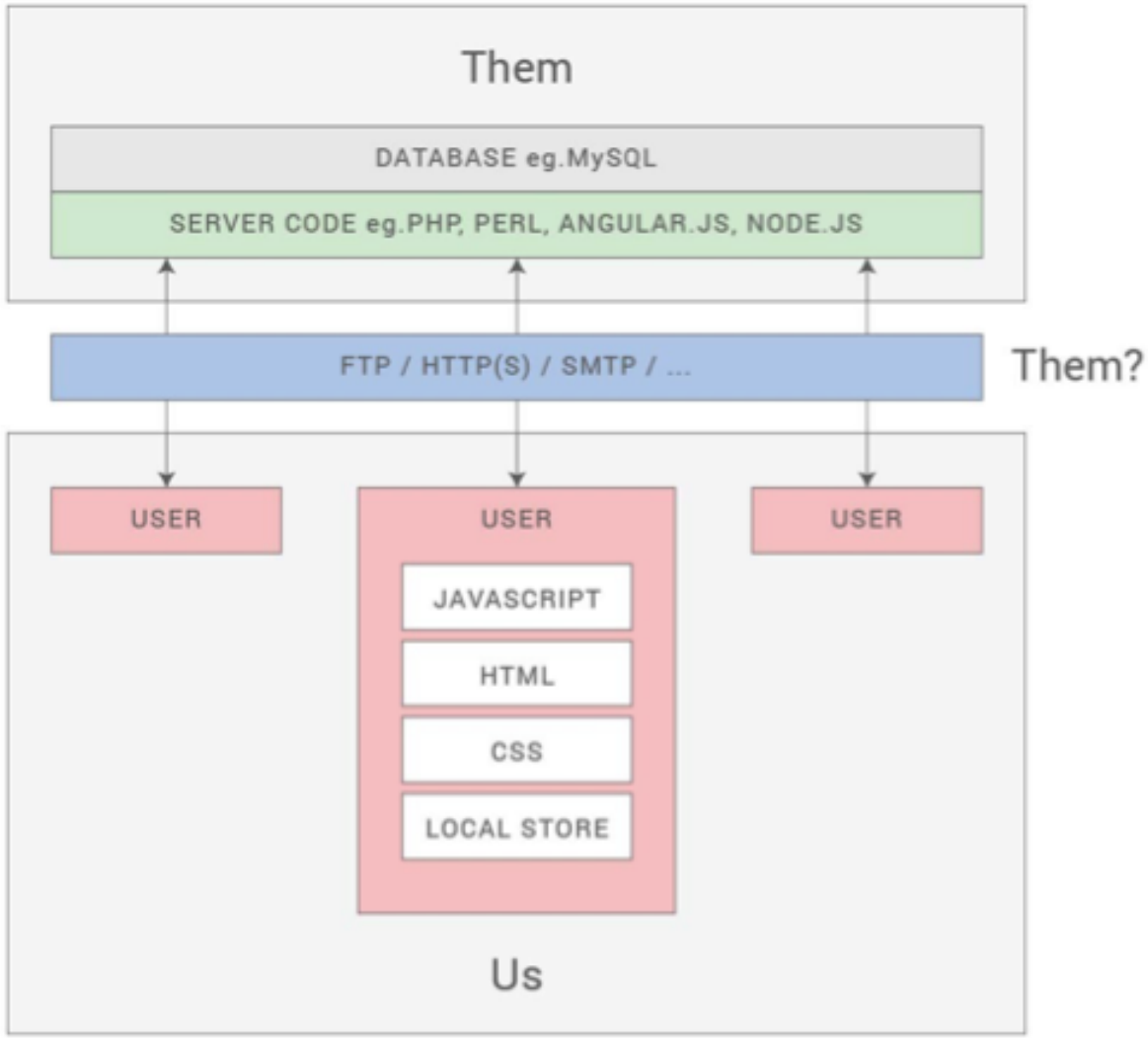
富邦人壽

EVA AIR 長榮航空

台新銀行



Web 2.0

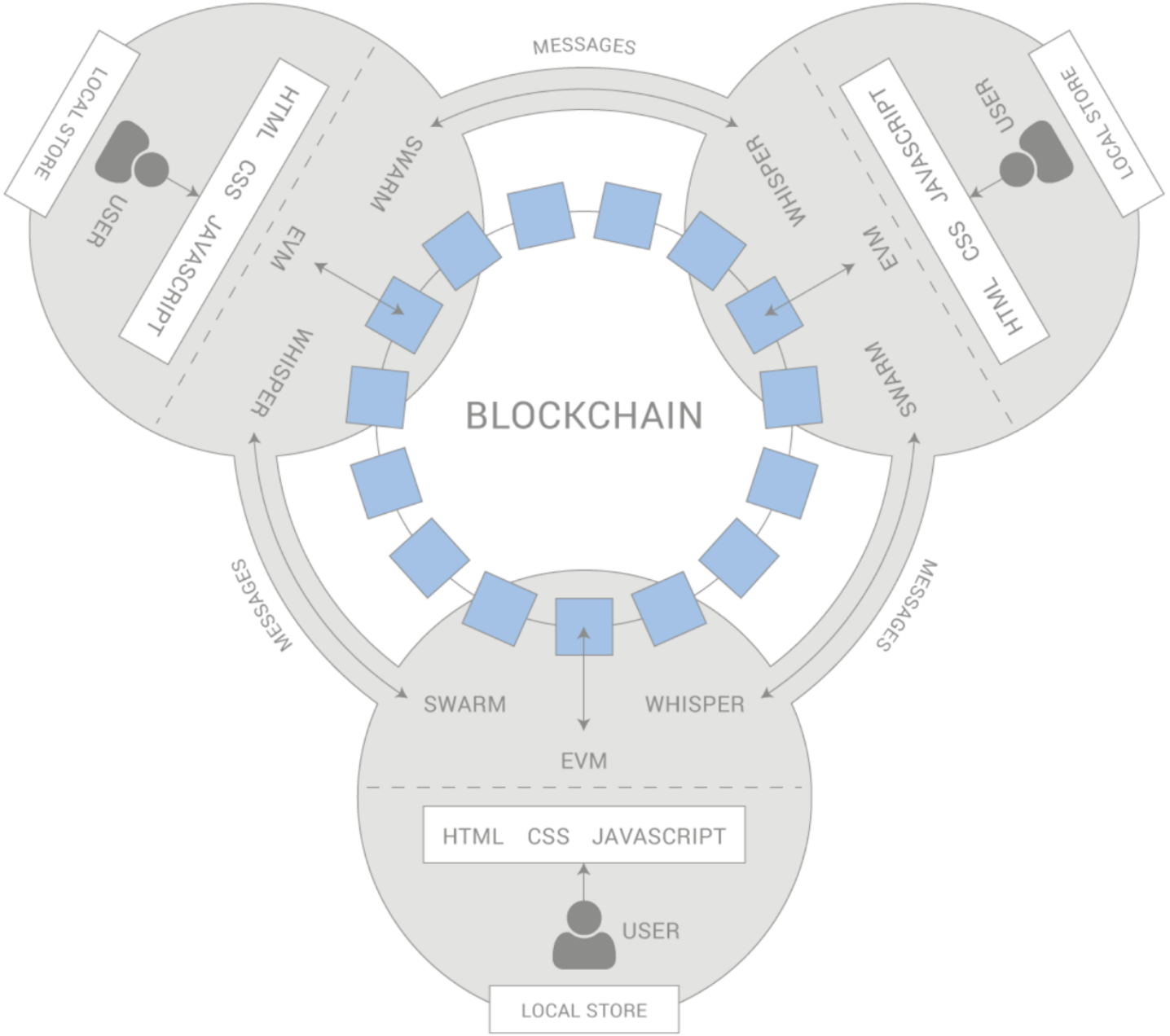


Web 2.0, How it is

The Ethereum Experience with Dr. Gavin Wood, CTO

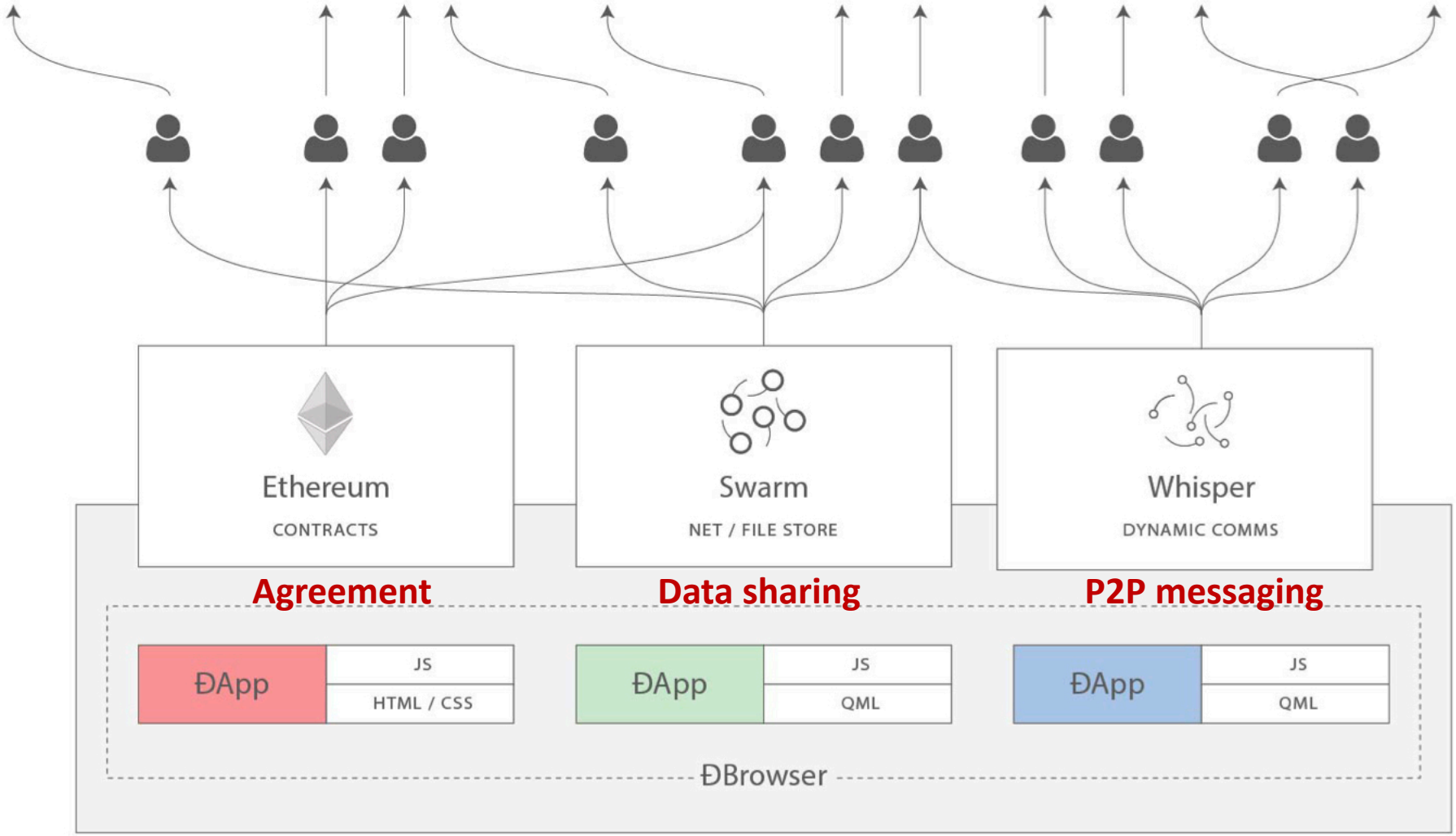
https://www.slideshare.net/ethereum/the-ethereum-experience/18

DApp



https://www.slideshare.net/ethereum/the-ethereum-experience/42

Web 3.0



https://www.slideshare.net/ethereum/the-ethereum-experience/43

比特幣有何限制？

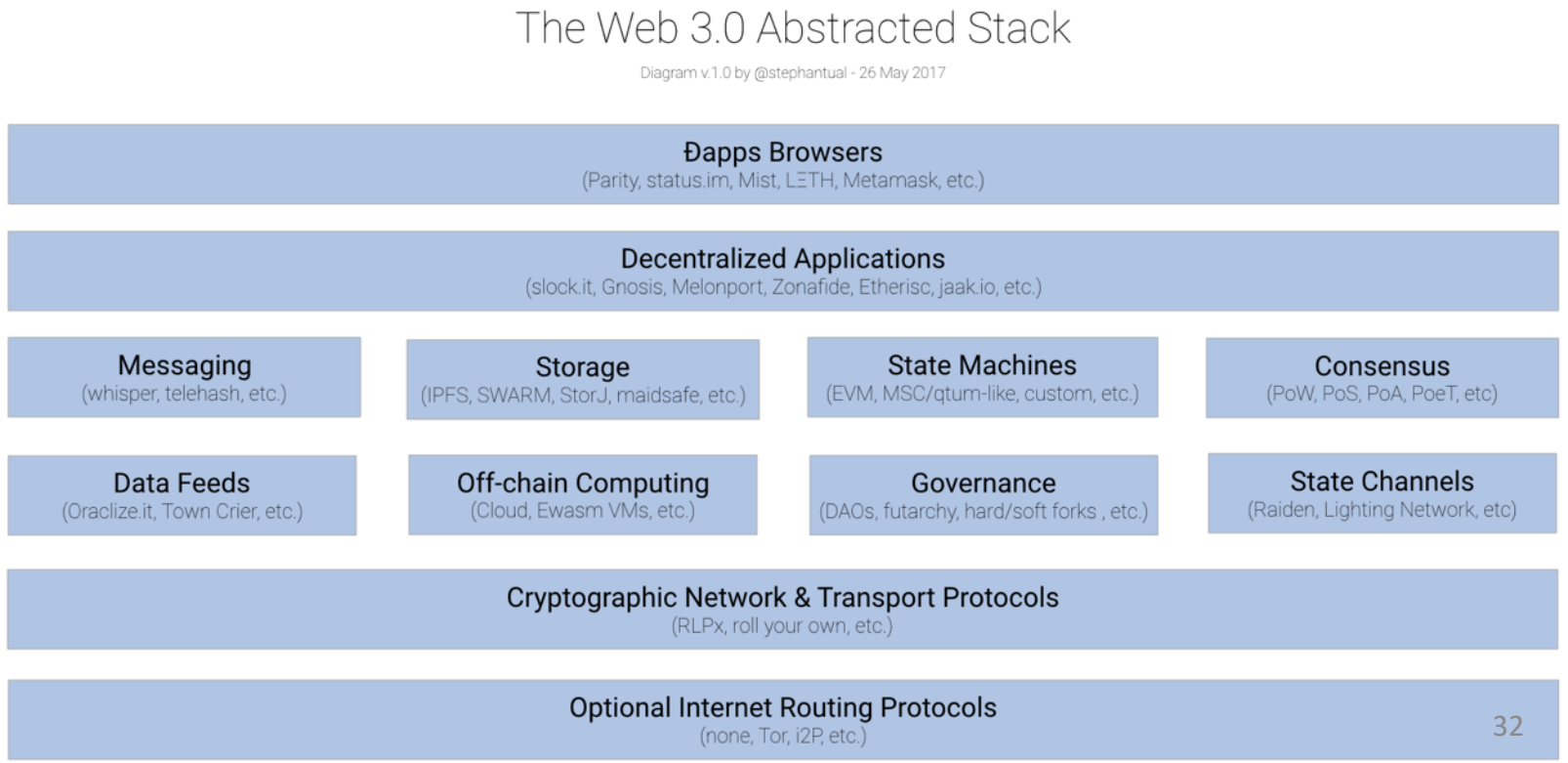
- Bitcoin
  - UTXO
  - OP_RETURN
  - Pricing and Fairness
  - Bitcoin script

https://karl.tech/intro-to-ethereum/

https://github.com/ethereum/wiki/wiki/Design-Rationale

協議設計原則

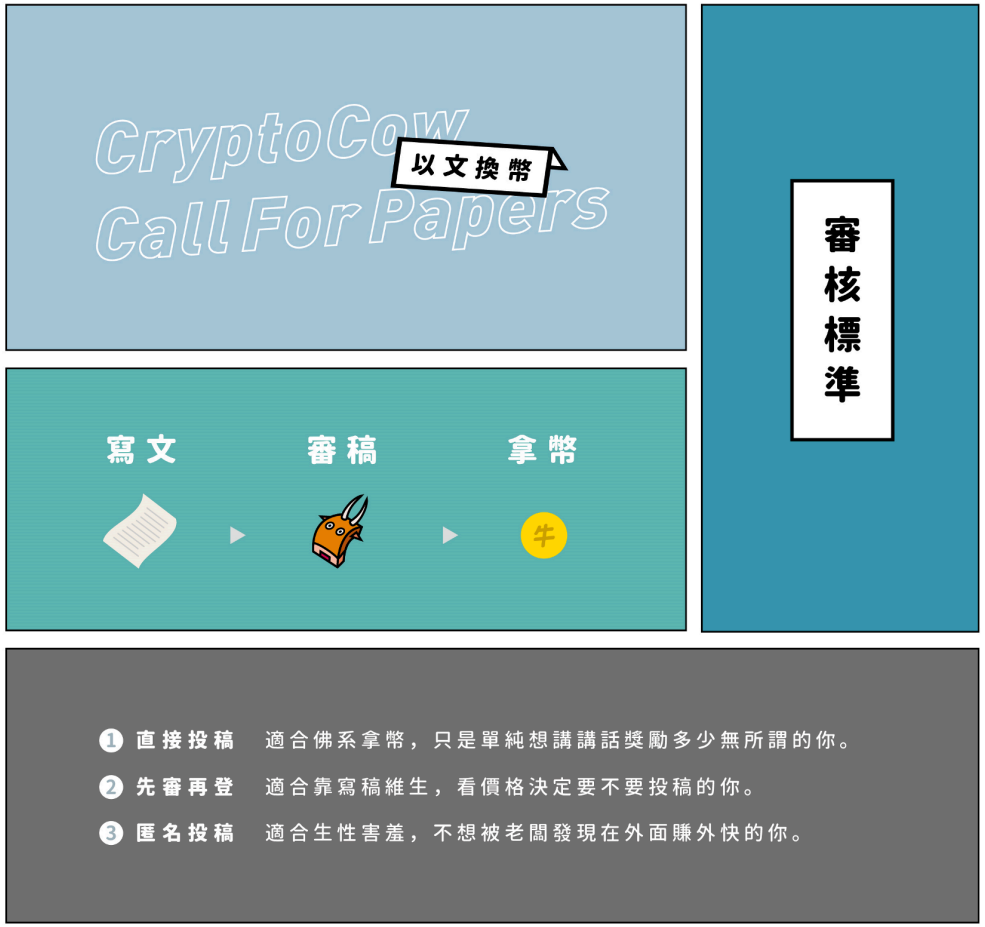
- 三明治模型
  - 應用層的介面設計對使用者要儘可能簡單 (Language, Interface)
  - 複雜的系統設計放在中間層
  - 核心共識



https://github.com/ethereum/wiki/wiki/Design-Rationale

協議設計原則

- 自由度
  - 不限制使用者如何使用以太坊協議
  - 保持網路中立, 對交易或合約一視同仁
  - 透過設置手續費來課徵污染稅 (Pigouvian tax)
    - https://medium.com/cryptocow/ethereum-transaction-fee-economics-8214242d5024
    - https://github.com/ethereum/research/blob/master/papers/pricing/ethpricing.pdf
    - https://www.youtube.com/watch?v=7vuTtvshR34



https://medium.com/cryptocow/cryptocow-submit-rules-a62b9ccc1efc

https://github.com/ethereum/wiki/wiki/Design-Rationale

協議設計原則

- 一般化
  - 協議的特性與操作碼要能最大限度地體現底層的概念
  - 以便功能能隨意被組合

https://github.com/ethereum/wiki/wiki/Design-Rationale

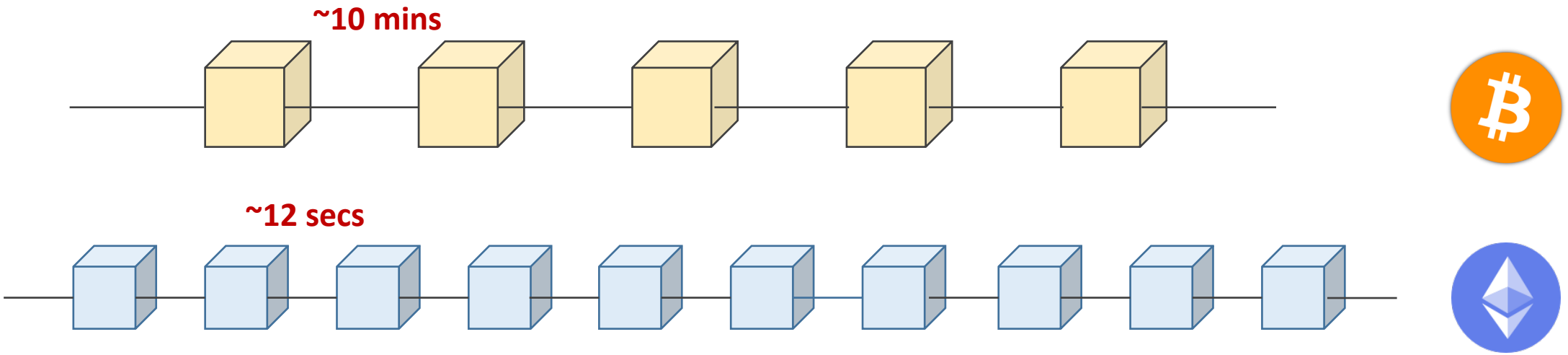
協議設計原則

- 不具任何特色
  - 為了遵循一般化原則，拒絕在協議中提供某種高階功能
  - 反之，使用者可以在合約中建立任何 sub-protocol
  - 例如: 沒有 TimeLock, Multisig 功能

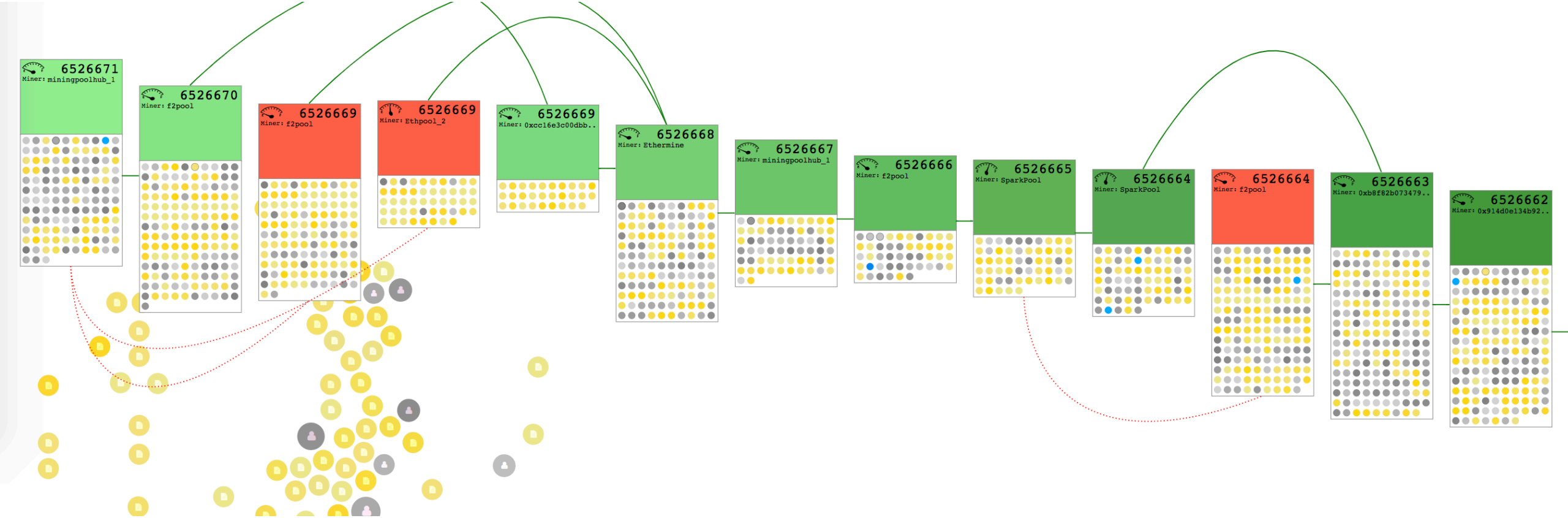
https://github.com/ethereum/wiki/wiki/Design-Rationale

協議設計原則

- 不排斥承擔風險
  - 如果風險增加提供更多好處，願意承擔更多風險
  - 例如
    - 一般的狀態轉換
    - 50 倍快的區塊生成時間
    - 高效的共識



Fork



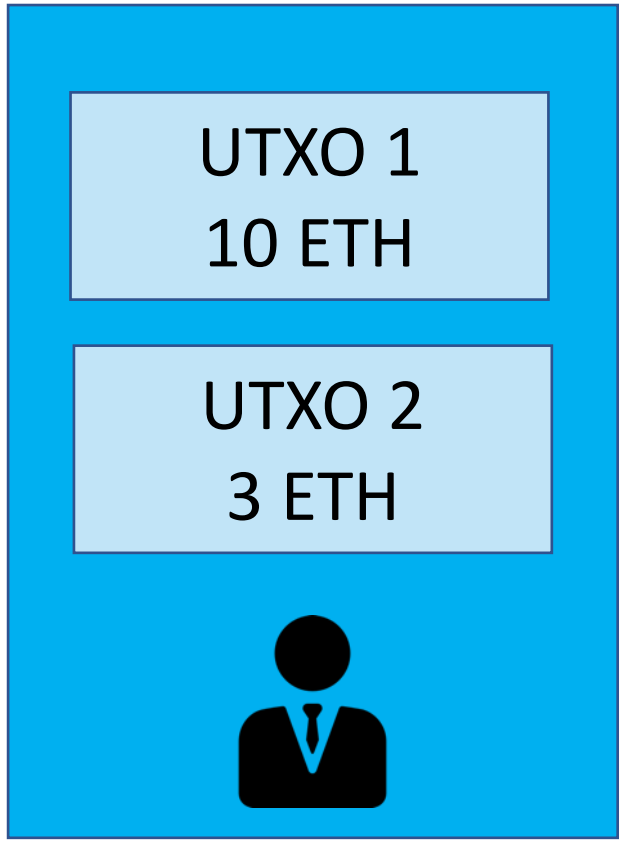
http://ethviewer.live/

Blockchain-level protocol

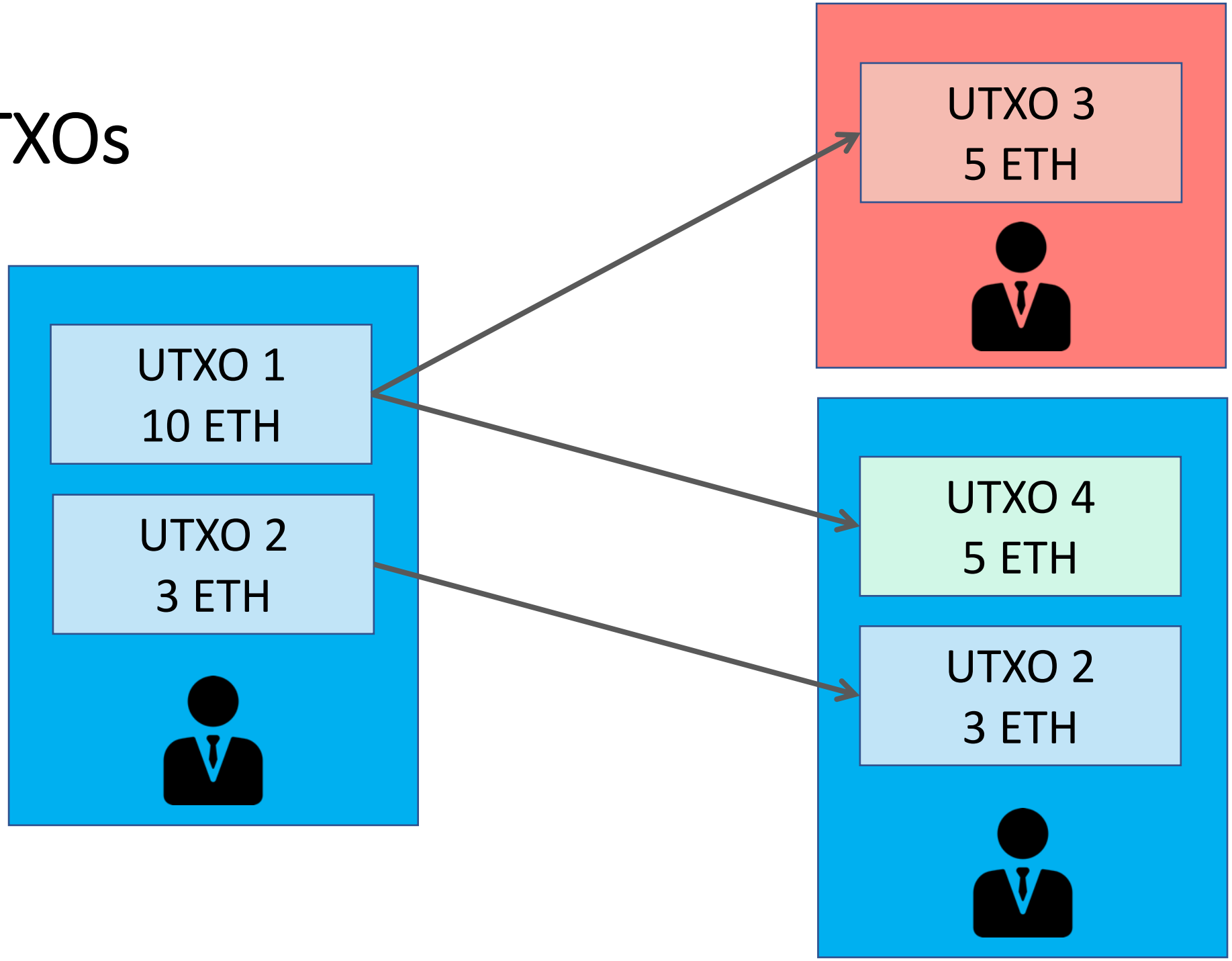
新的設計

- Account
- Merkle Patricia tree/trie (MPT)
- Recursive Length Prefix (RLP)
- Ether/gas/fee
- EVM

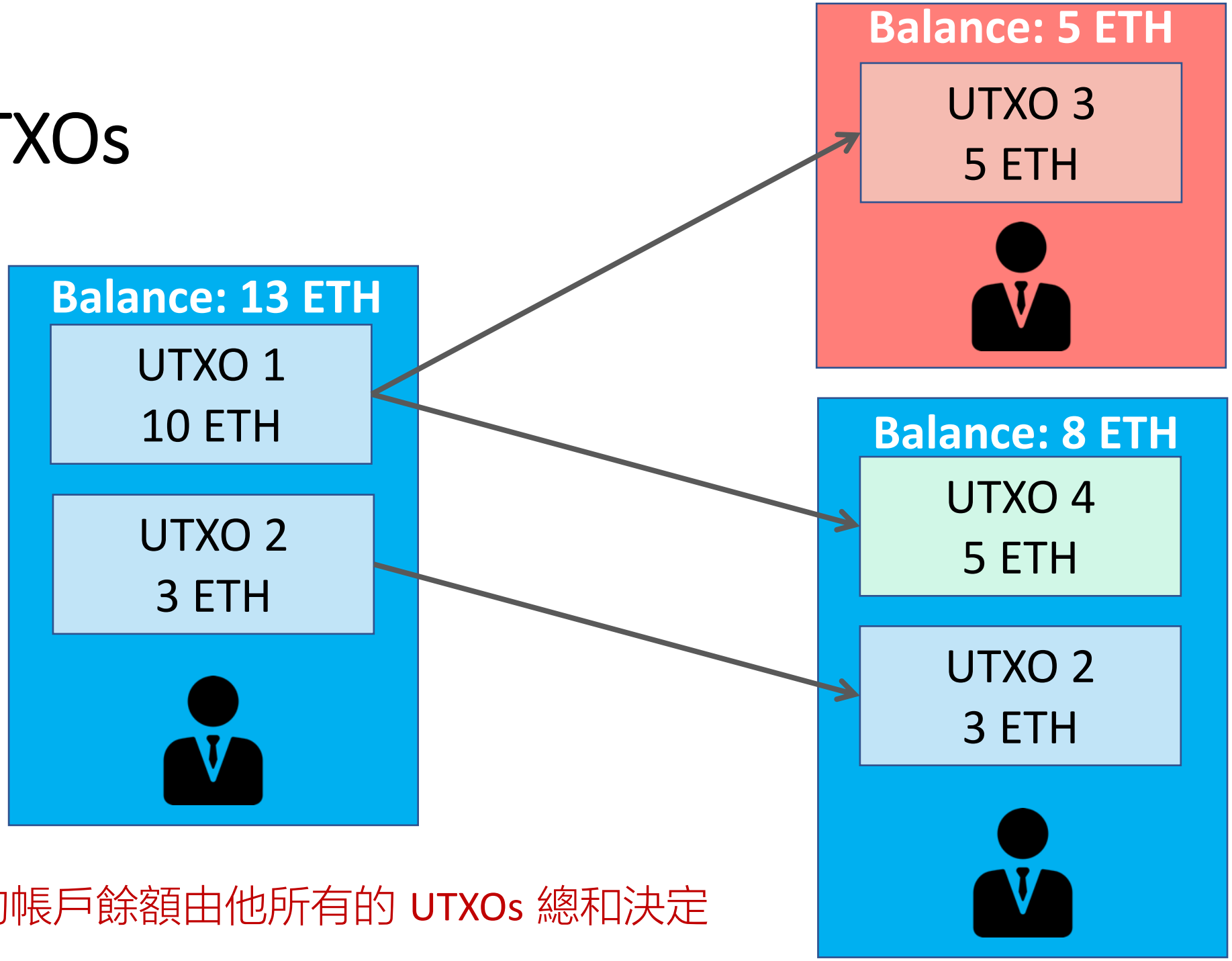
UTXOs



UTXOs

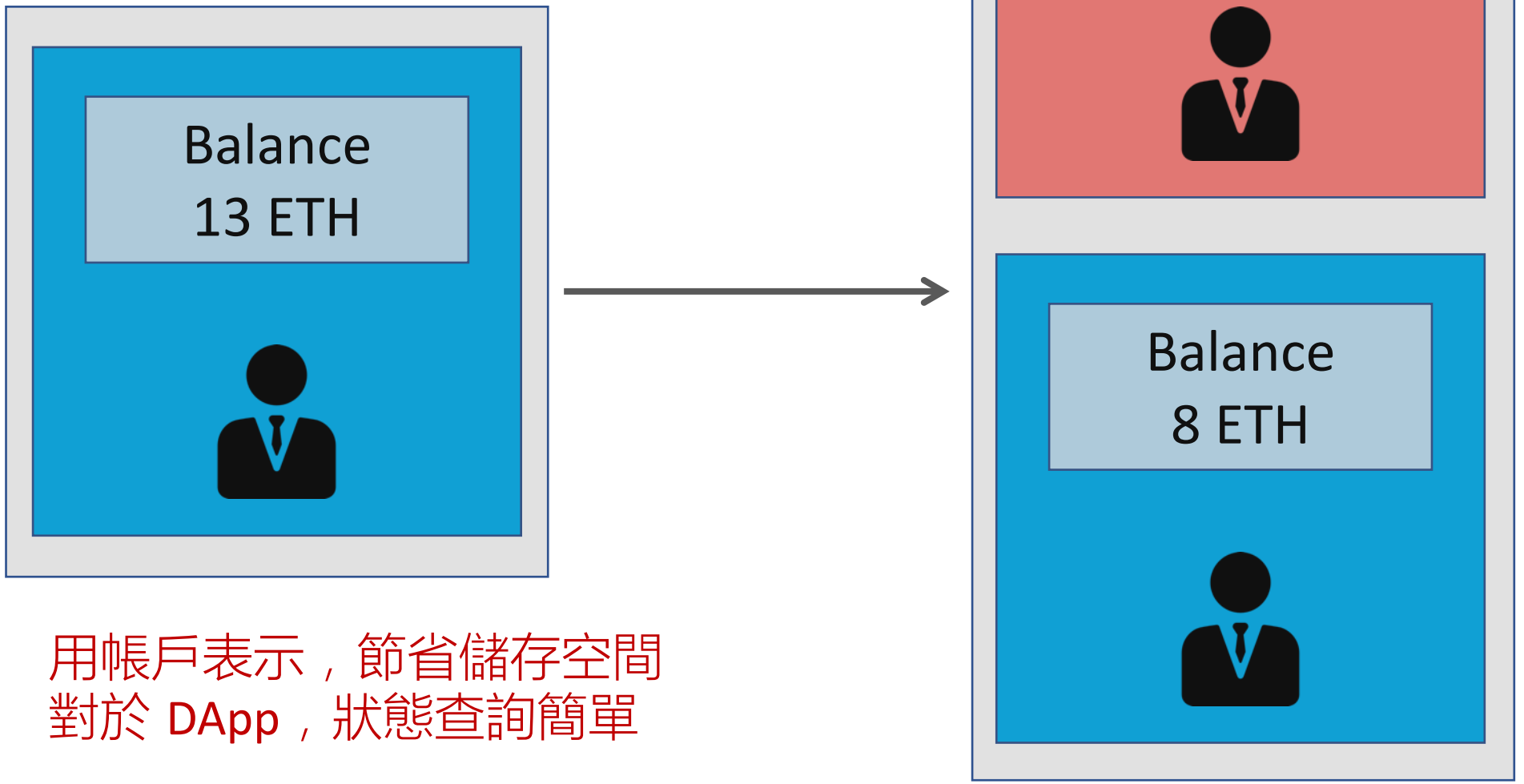


UTXOs



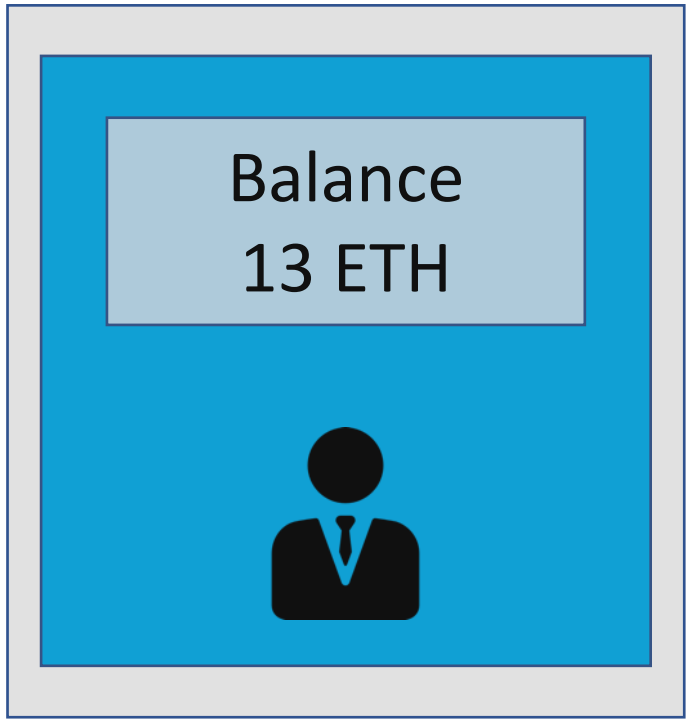
使用者的帳戶餘額由他所有的 UTXOs 總和決定

Accounts



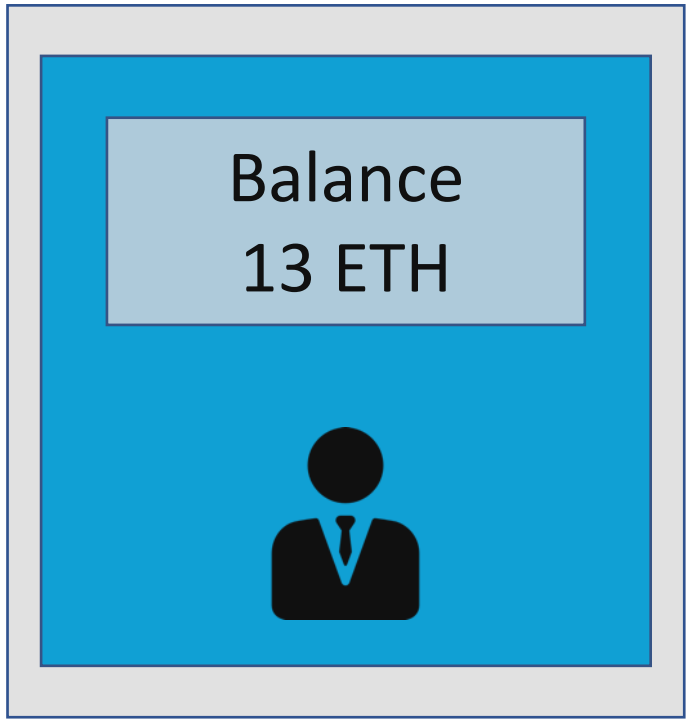
用帳戶表示，節省儲存空間
對於 DApp，狀態查詢簡單

Accounts



用帳戶表示，節省儲存空間
對於 DApp，狀態查詢簡單
隱私性相對較差
擴展性差

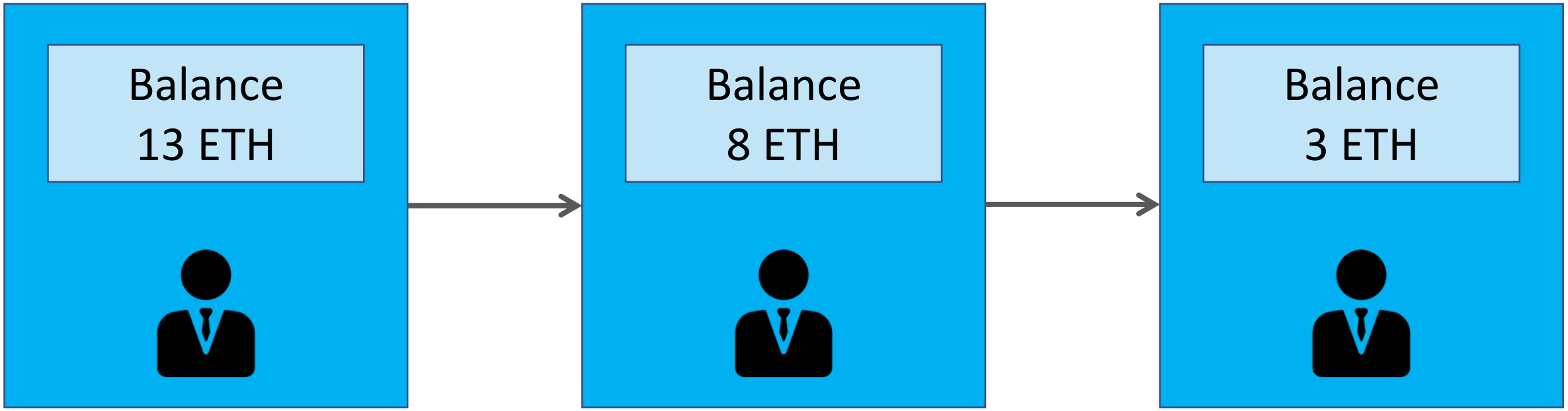
Accounts



有 replay attack 的風險
回想一下，之前提過交易的流水號

Accounts

透過 nonce 來防止 replay attack
不可跳號，nonce 依序執行



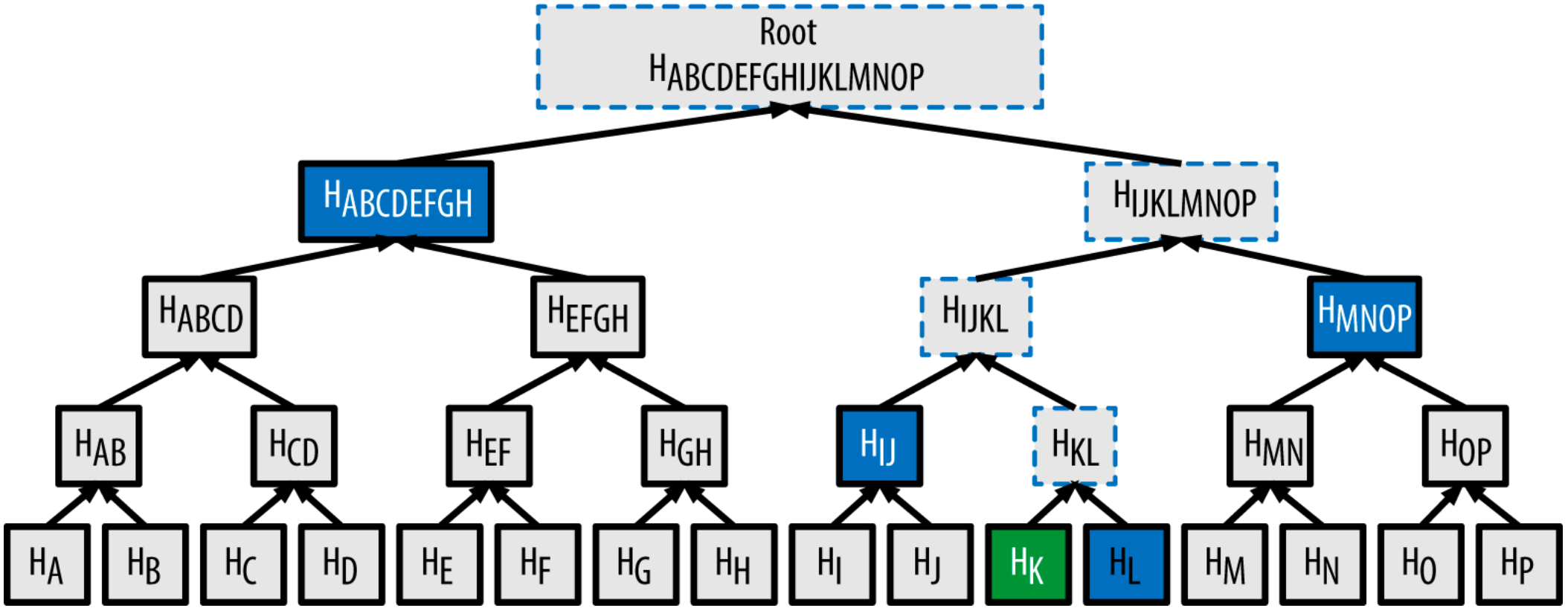
Tx 1 (nonce 0)

Tx 2 (nonce 1)

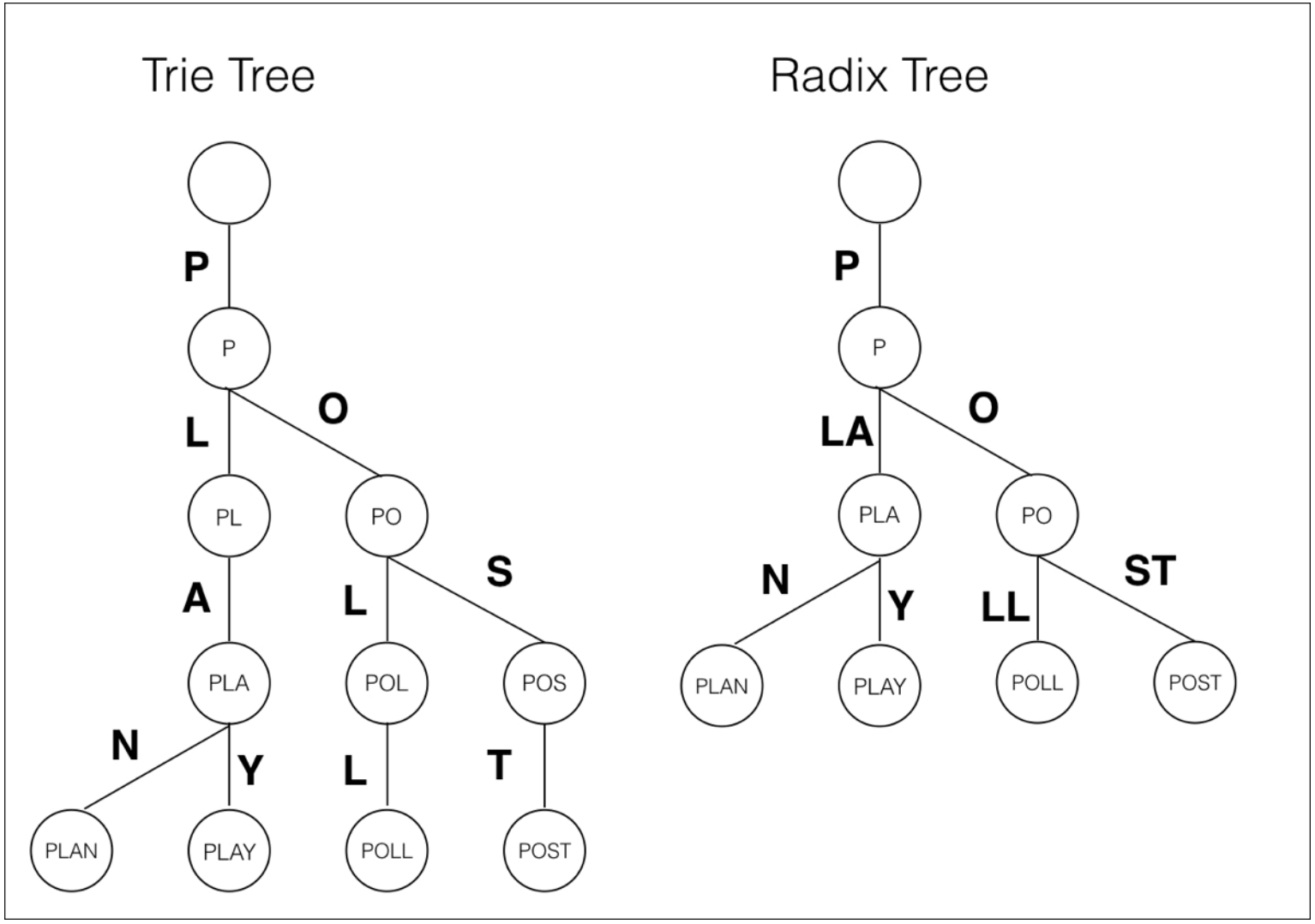
Merkle Patricia tree/trie

- 結合兩種樹狀結構，對 key/value 搜尋優化
  - Radix tree
  - Merkle tree

Merkle tree


Root
HABCDEFGHIJKLMNOP
HABCDEFGH
HIJKLMNOP
HABCD
HEFGH
HIJKL
HMNOP
HAB
HCD
HEF
HGH
HIJ
HKL
HMN
HOP
HA
HB
HC
HD
HE
HF
HG
HH
HI
HJ
HK
HL
HM
HN
HO
HP

Radix tree



https://www.safaribooksonline.com/library/view/swift-data-structure/9781785884504/ch06s07.html

How does Ethereum work, anyway?

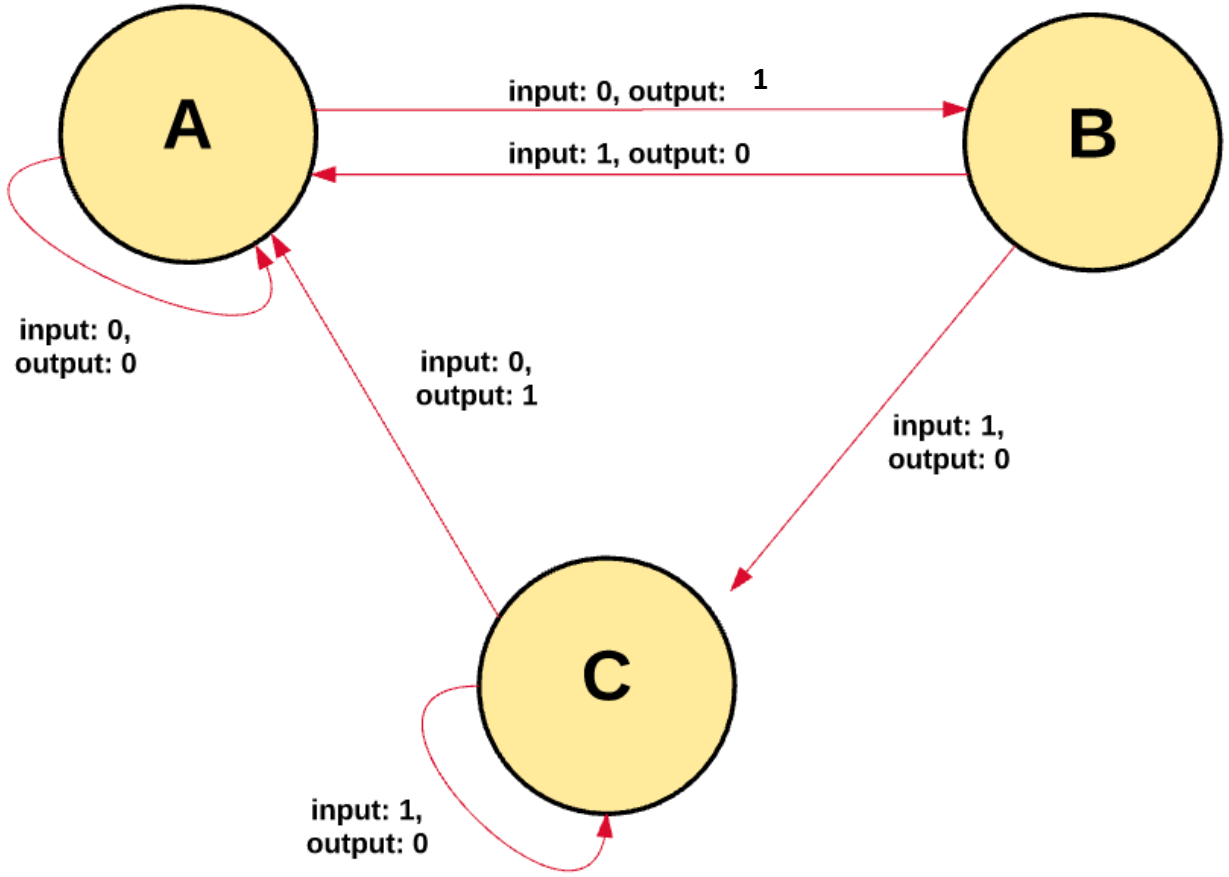
Author: Preethi Kasireddy

https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

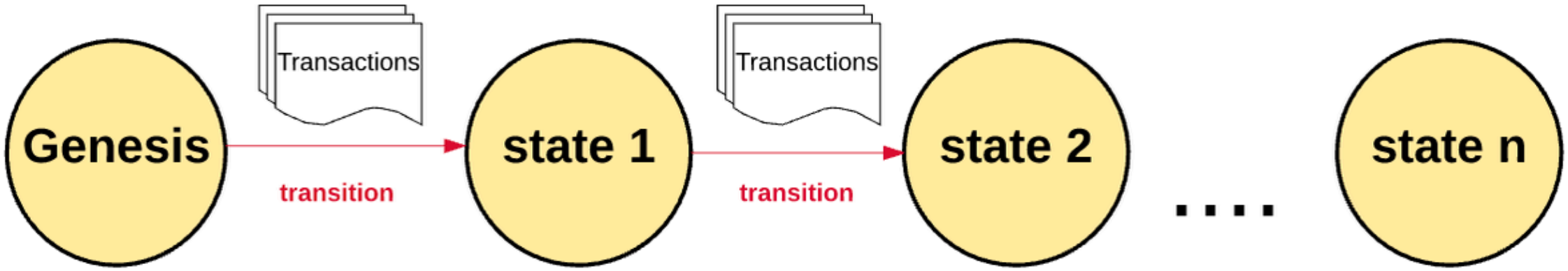
Blockchain

- A blockchain is a ”cryptographically secure transactional singleton machine with shared-state.”
  - cryptographically secure
  - transactional singleton machine
  - with shared-state.

Transaction-based state machine



From genesis state to current state

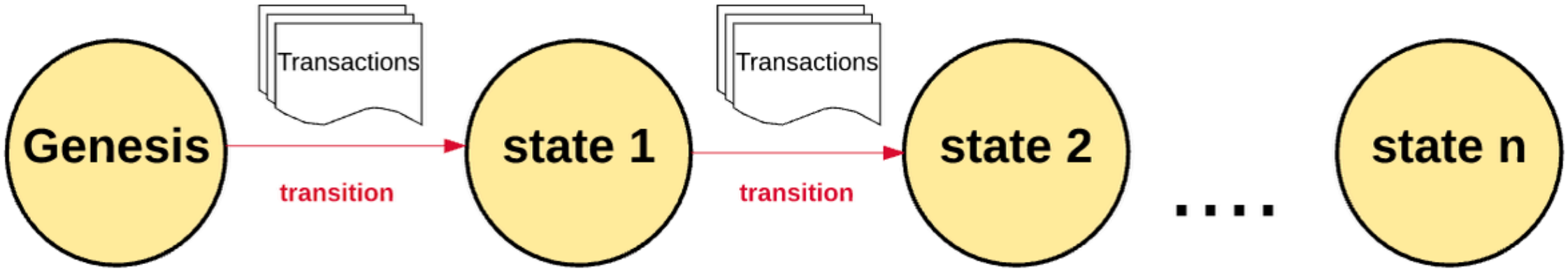


A → B: 20
C → D: 30
E: 100

A: 100
B: 100
C: 100
D: 100

A: 80
B: 120
C: 70
D: 130
E: 100

Global state

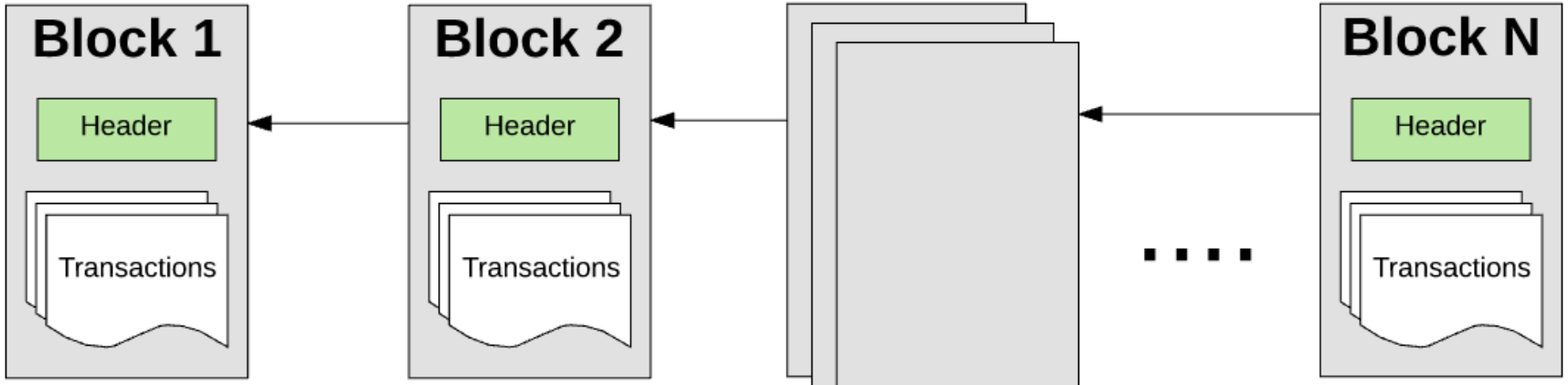


Storage

A: 100
B: 100
C: 100
D: 100

A: 80
B: 120
C: 70
D: 130
E: 100

A: 80
B: 120
C: 70
D: 130
E: 100
F: 100

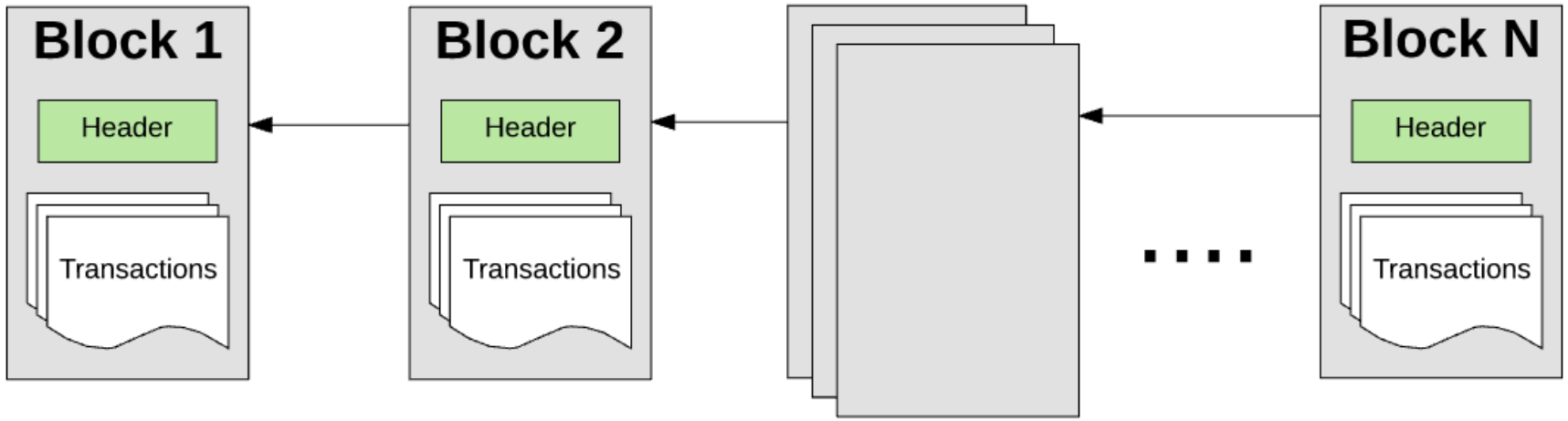


A → B: 20
C → D: 30
E: 100

F: 100

Mining

- 驗證交易合法, 透過 Proof of Work
- 從上一個狀態轉換成下一個狀態

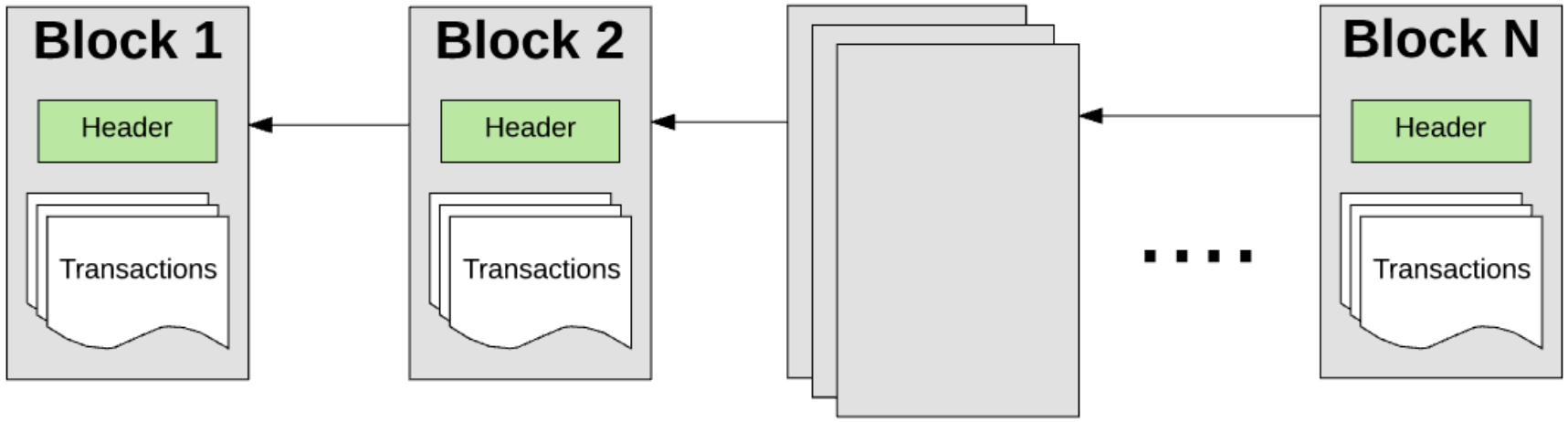


Block reward

- 驗證交易並形成新區塊的礦工，會得到獎勵
- 獲得以太幣 (Ether) 做為獎賞

Block size (Gas)

~8,000,029 gas



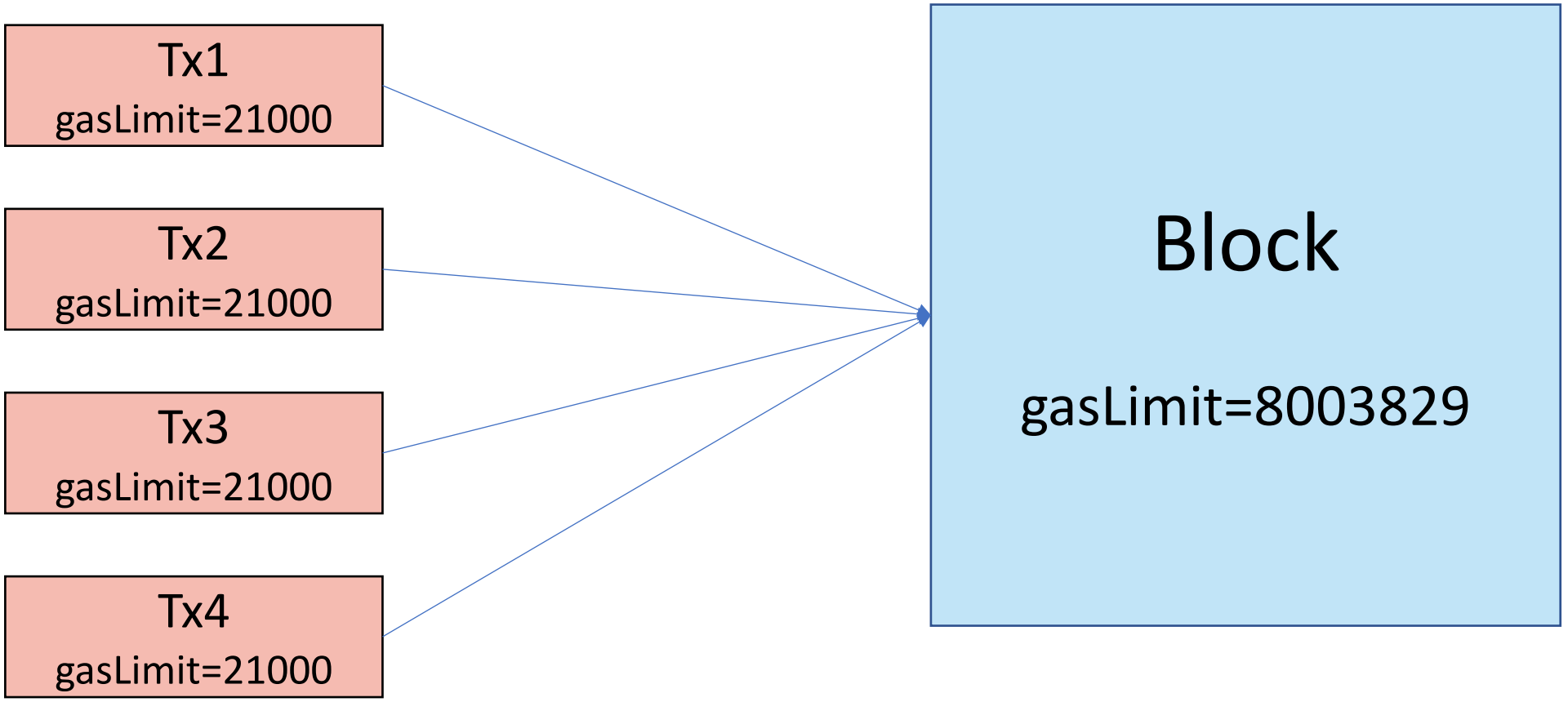
https://github.com/djrtwo/evm-opcode-gas-costs/blob/master/opcode-gas-costs_EIP-150_revision-1e18248_2017-04-12.csv#L62

EVM (opcode - gas cost)

Value	Mnemonic	Gas Used	Subset	Removed from stack	Added to stack	Notes
0x00	STOP	0	zero	0	0	Halts execution.
0x01	ADD	3	verylow	2	1	Addition operation
0x02	MUL	5	low	2	1	Multiplication operation.
0x03	SUB	3	verylow	2	1	Subtraction operation.
0x04	DIV	5	low	2	1	Integer division operatio
0x05	SDIV	5	low	2	1	Signed integer division c
0x06	MOD	5	low	2	1	Modulo remainder opera
0x07	SMOD	5	low	2	1	Signed modulo remainde
0x08	ADDMOD	8	mid	3	1	Modulo addition operati
0x09	MULMOD	8	mid	3	1	Modulo multiplication op
0x0a	EXP	(exp == 0) ? 10 : (10 + 10 * (1 + log256(exp)))		2	1	Exponential operation.

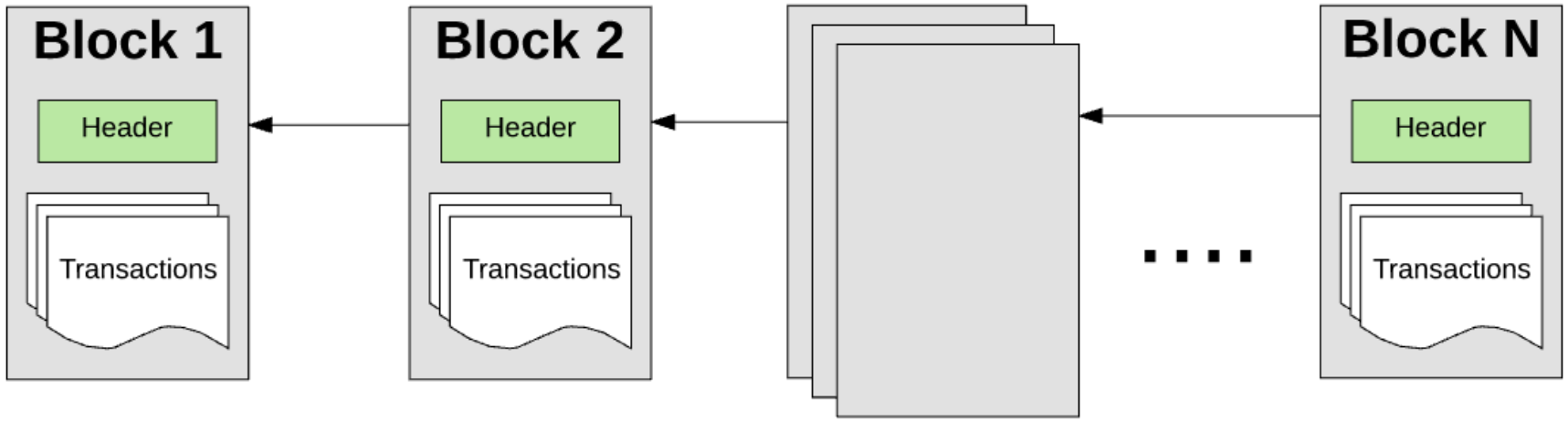
Block & gasLimit

Block 可容納的交易最大數不可超過 Block 的 gasLimit
ex: 8003829/21000 ~= 381



Block time

~8,000,029 gas



~12 secs

Blocks

View All

Block 6526905 >47 secs ago	Mined By MiningPoolHub_1 **41 Txns** in 2 sec Block Reward 3.0248 Ether
Block 6526904 >49 secs ago	Mined By DwarfPool_1 **89 Txns** in 21 sec Block Reward 3.05948 Ether
Block 6526903 >1 min 10 secs ago	Mined By MiningPoolHub_1 **65 Txns** in 5 sec Block Reward 3.04938 Ether

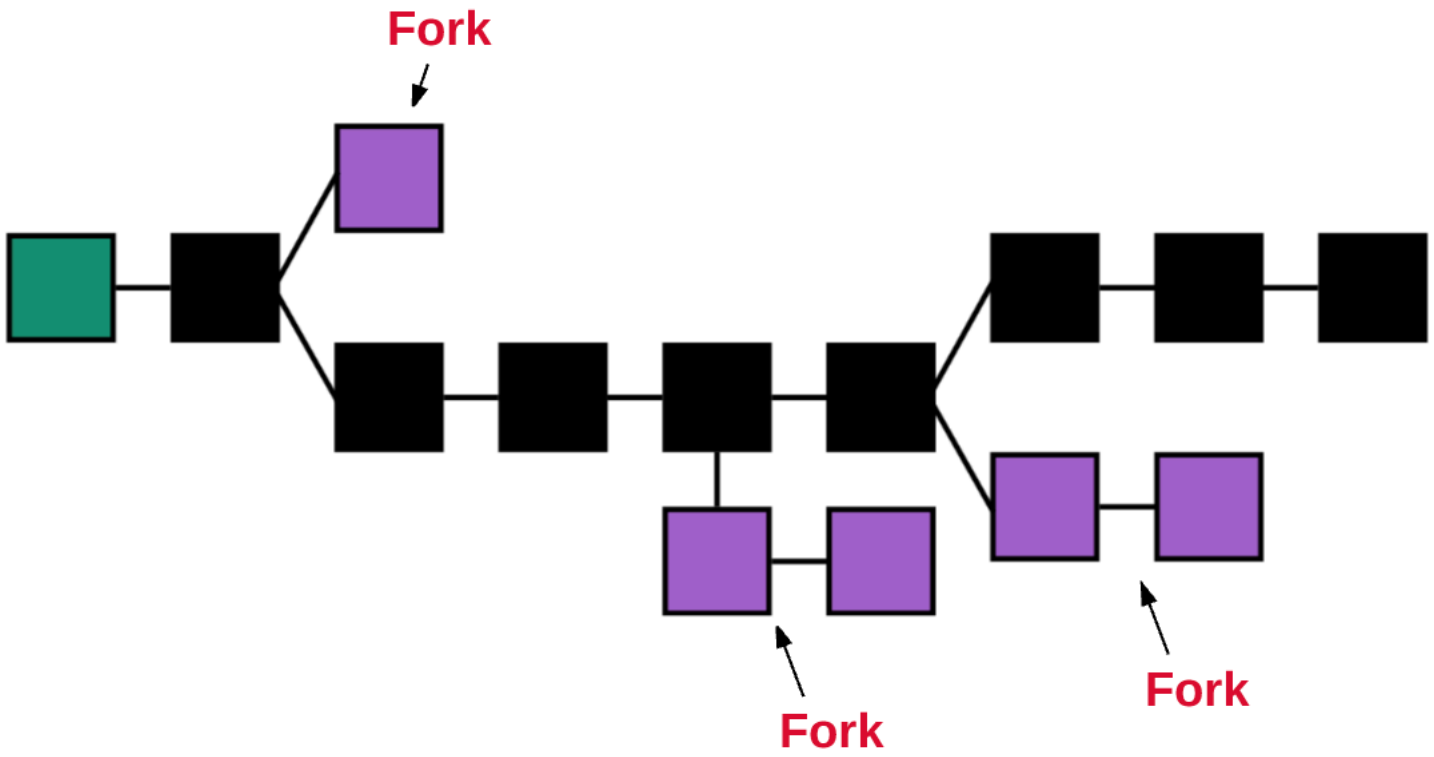
Block #6526886 To #6526910 (Total of 6526911 Blocks)

First | Prev | Page **1** of **261077** | Next | Last

Height	Age	txn	Uncles	Miner	GasUsed	GasLimit	Avg.GasPrice	Reward
6526910	28 secs ago	61	0	MiningPoolHub_1	7990649 (99.88%)	8000029	5.24 Gwei	3.04186 Ether
6526909	37 secs ago	138	1	SparkPool	7996442 (99.96%)	8000029	3.23 Gwei	3.11961 Ether
6526908	38 secs ago	24	0	Ethermine	1466444 (18.33%)	8000029	37.42 Gwei	3.05487 Ether
6526907	54 secs ago	67	0	Ethermine	7981592 (99.77%)	8000029	3.72 Gwei	3.02967 Ether
6526906	1 min ago	23	1	MinerallPool	1124849 (14.05%)	8003875	34.56 Gwei	3.13262 Ether
6526905	1 min ago	82	0	Ethermine	7983629 (99.80%)	7999992	3.24 Gwei	3.02587 Ether
6526904	1 min ago	89	0	DwarfPool_1	3872901 (48.46%)	7992222	15.36 Gwei	3.05948 Ether
6526903	1 min ago	65	0	MiningPoolHub_1	7999898 (100.00%)	8000029	6.17 Gwei	3.04938 Ether
6526902	1 min ago	104	0	SparkPool	7979635 (99.75%)	8000029	1.77 Gwei	3.01414 Ether
6526901	1 min ago	103	0	SparkPool	4421679 (55.27%)	8000029	15.84 Gwei	3.07004 Ether
6526900	1 min ago	103	0	SparkPool	7992210 (99.90%)	8000029	4.27 Gwei	3.03414 Ether
6526899	1 min ago	83	0	Ethermine	7991895 (99.90%)	8000029	8.42 Gwei	3.06729 Ether
6526898	2 mins ago	135	1	MiningPoolHub_1	7987068 (99.84%)	8000029	6.05 Gwei	3.14205 Ether
6526897	2 mins ago	181	0	Nanopool	7986851 (99.84%)	8000029	18.83 Gwei	3.15041 Ether

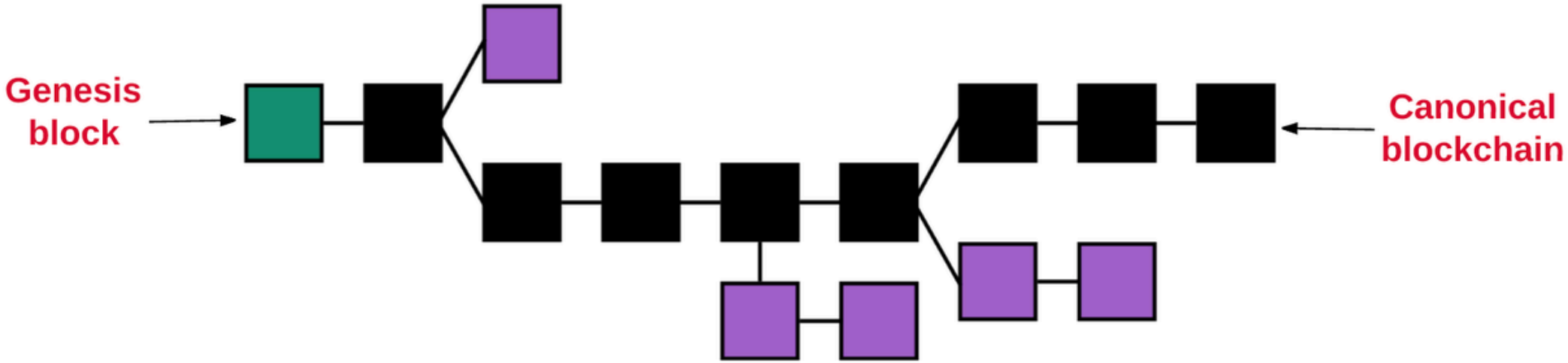
鏈的一致性

- GHOST = Greedy Heaviest Observed Subtree protocol



GHOST

Ethereum uses only uses longest chain, instead of the actual heaviest chain proposed in GHOST.



https://ethereum.stackexchange.com/questions/13378/what-is-the-exact-longest-chain-rule-implemented-in-the-ethereum-homestead-p

Uncle reward

R = 3 ETH

- $(U_n + 8 - B_n) * R \div 8$

Maximum of 2 Uncles is allowed per block

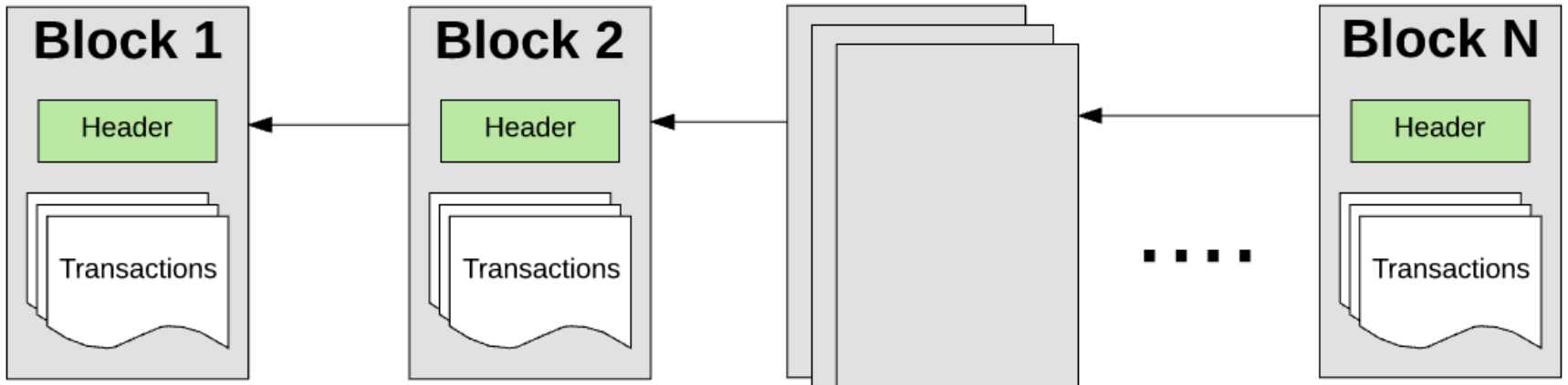
Block Height	UncleNumber	Age	Miner		Reward
6526925	6526923	2 mins ago	Nanopool	(23-25+8)*3/8	2.25 Ether
6526917	6526916	3 mins ago	0xd4383232c8d1db…		2.625 Ether
6526916	6526915	4 mins ago	SparkPool		2.625 Ether
6526914	6526912	4 mins ago	SparkPool		2.25 Ether
6526909	6526908	5 mins ago	Nanopool		2.625 Ether
6526906	6526905	6 mins ago	MiningPoolHub_1		2.625 Ether
6526898	6526895	7 mins ago	SparkPool	(95-98+8)*3/8	1.875 Ether

名詞

- 帳戶 (accounts)
- 狀態 (state)
- gas 和手續費 (gas and fees)
- 交易 (transactions)
- 區塊 (blocks)
- 交易執行 (transaction execution)
- 挖礦 (mining)
- 工作量證明 (proof of work)

帳戶

- 在以太坊中，全局的共享狀態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)



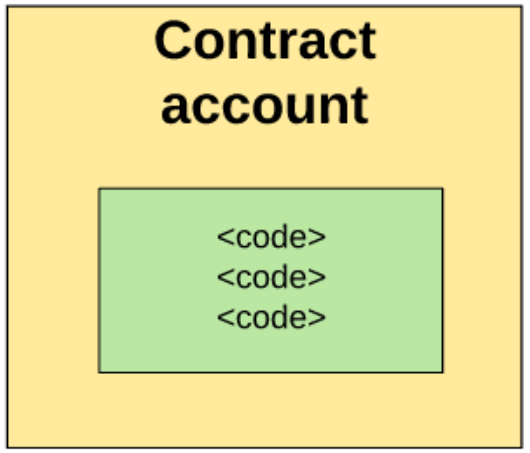
A → B: 20
C → D: 30
E: 100

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

https://media.consensys.net/how-to-store-digital-assets-on-ethereum-a2bfdcf66bd0

帳戶類型

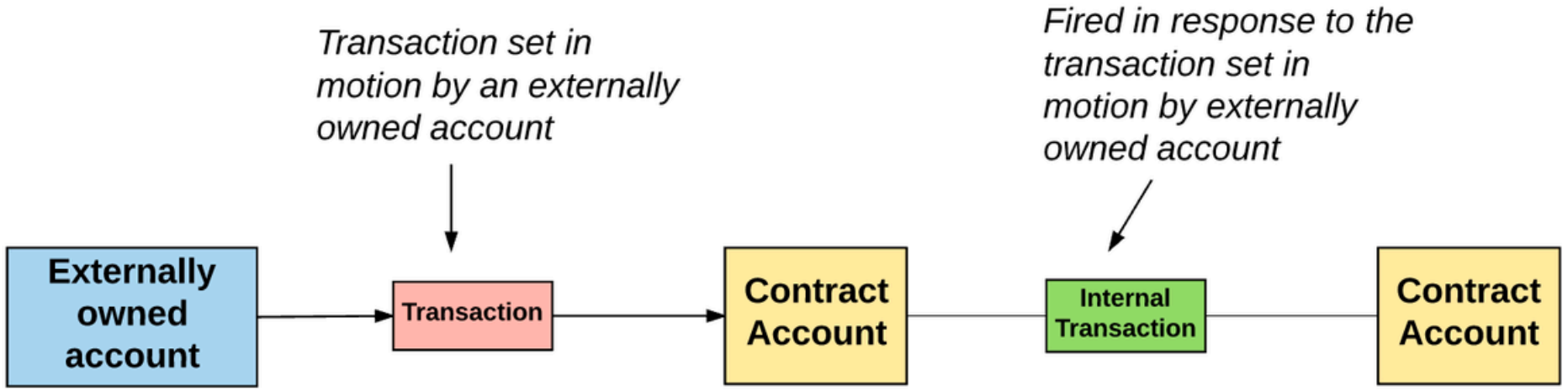
- 在以太坊中，帳戶主要分兩種類型
- 外部帳戶 (External owned account, EOA), 由私鑰進行控制
- 合約帳戶 (Contract account), 與合約程式碼相關聯



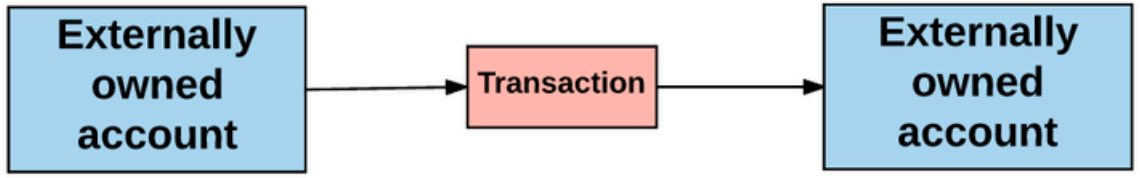
contractAddress = sha3(rlp_encode([address, nonce]))[14:]

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

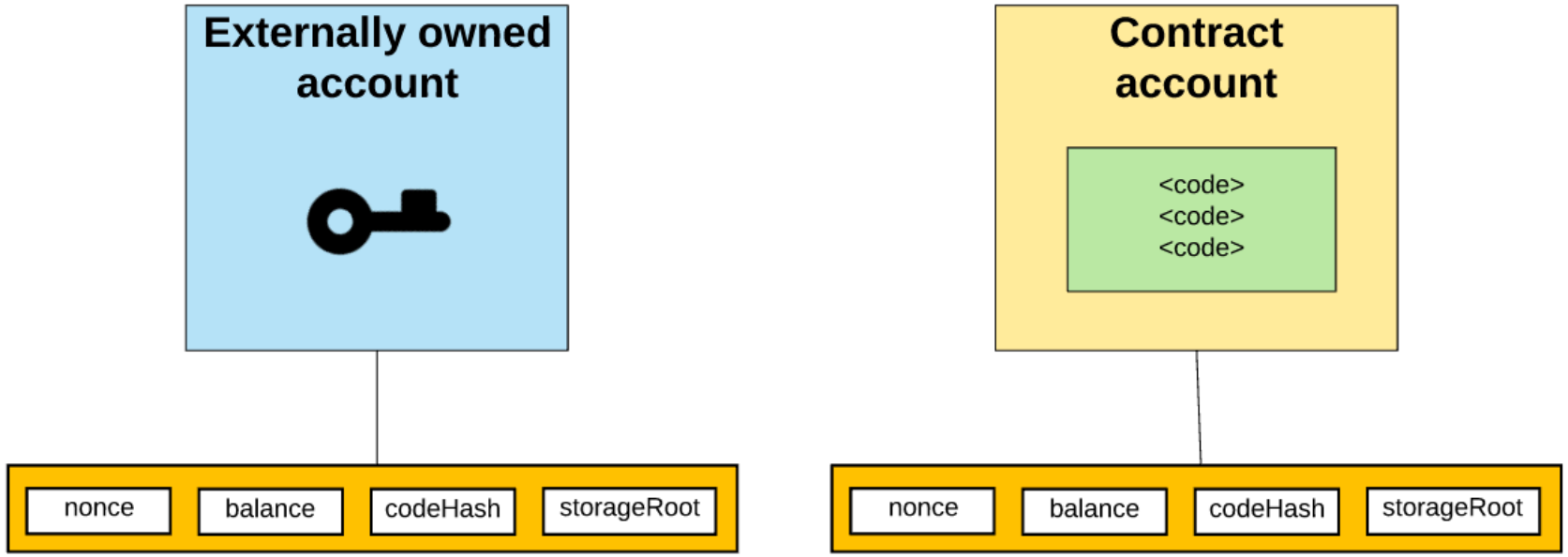


兩個外部帳戶之間的訊息傳遞只是一個簡單的價值轉移



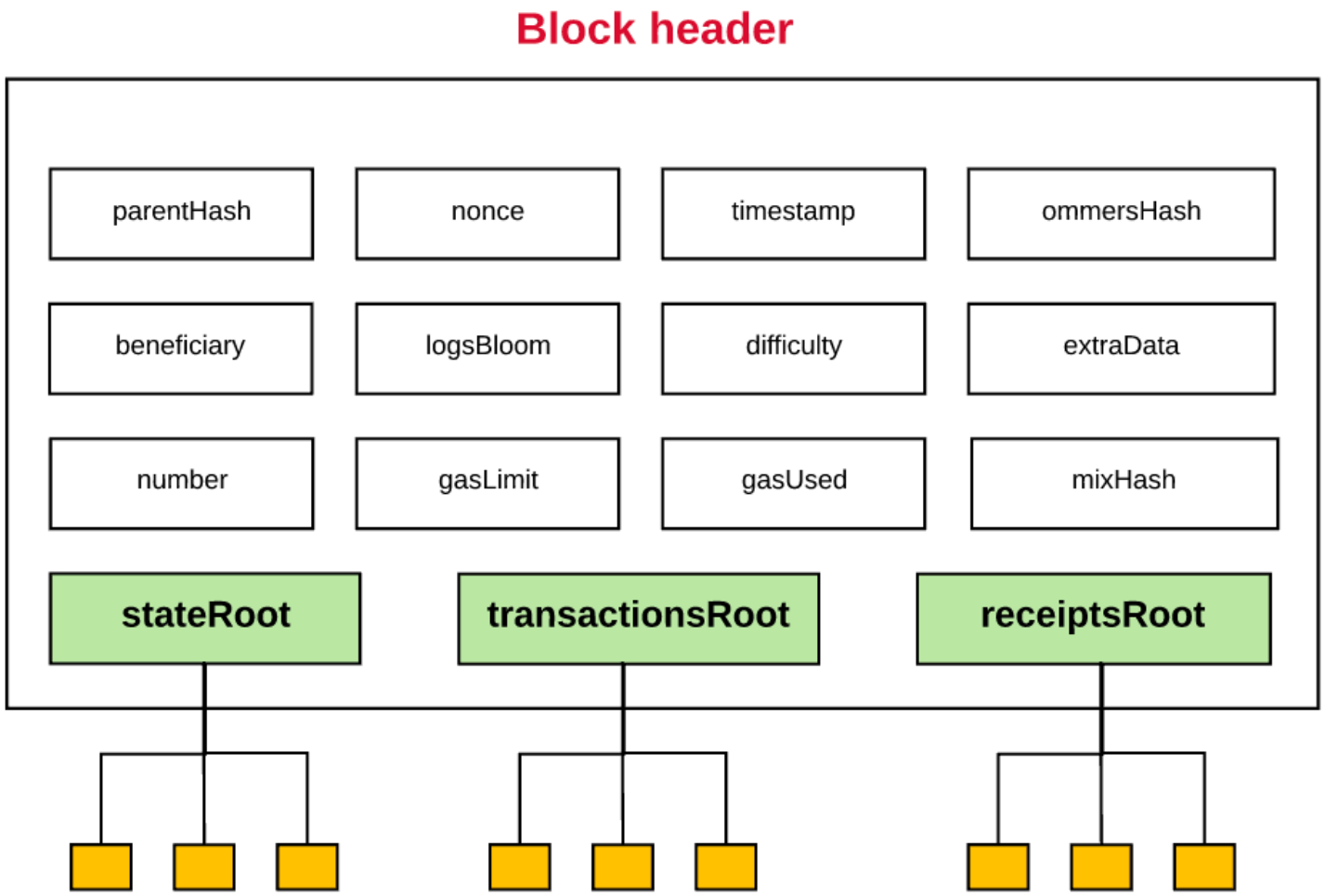
帳戶狀態 (Account state)

- Nonce
- Balance
- codeHash
- storageRoot

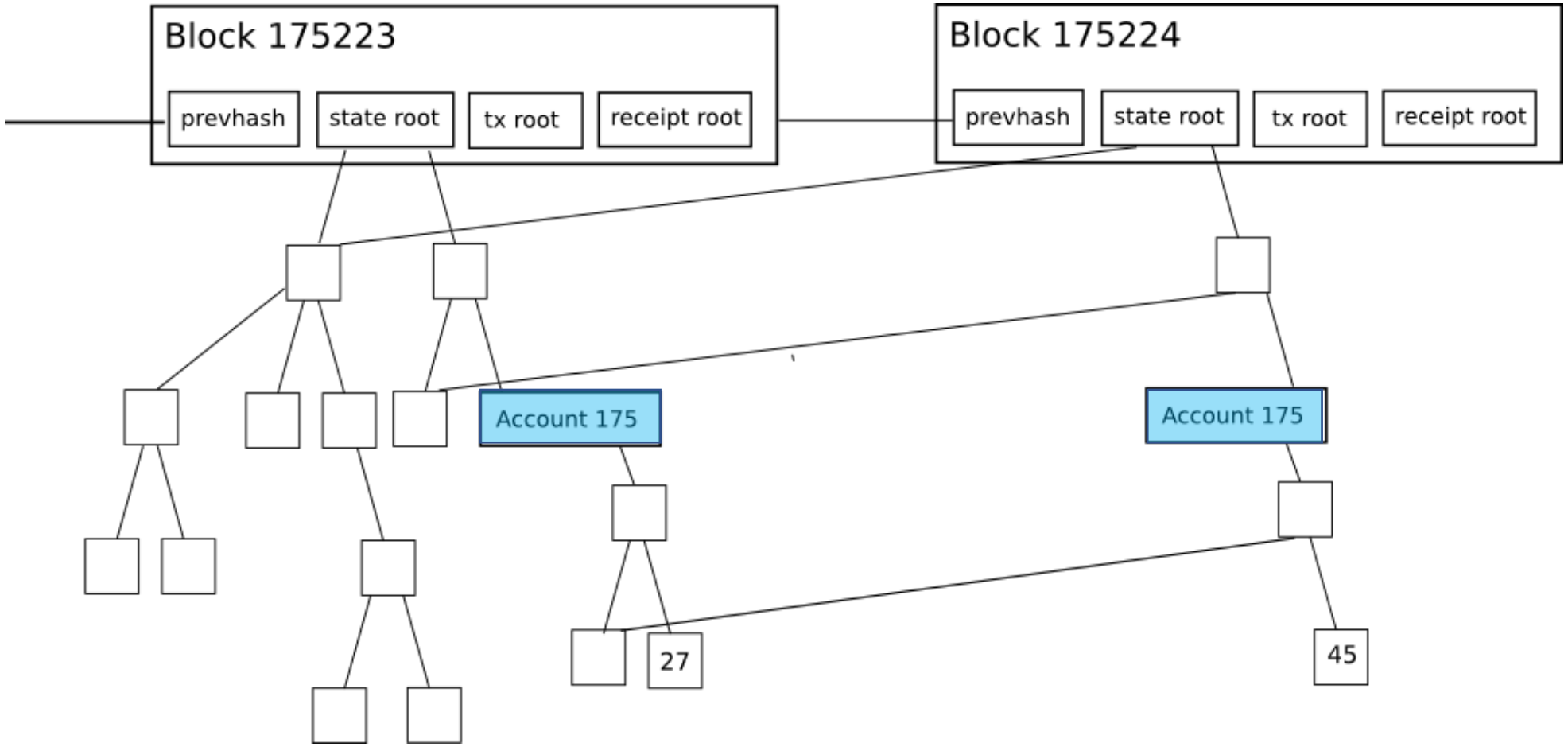


> eth.getCode("0xfec2394d5952828b00f4420caa7d0cd887846b09")

Block header

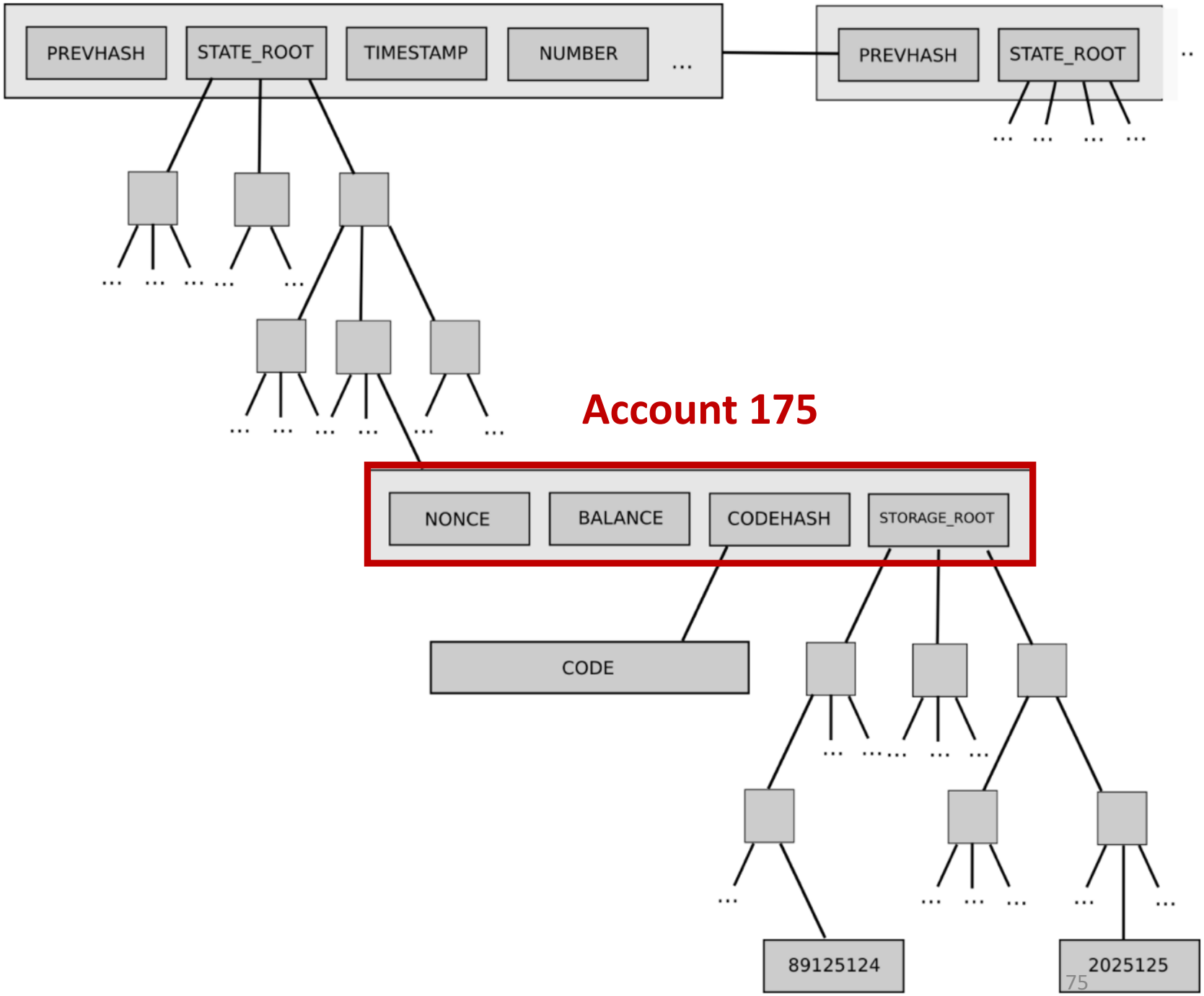


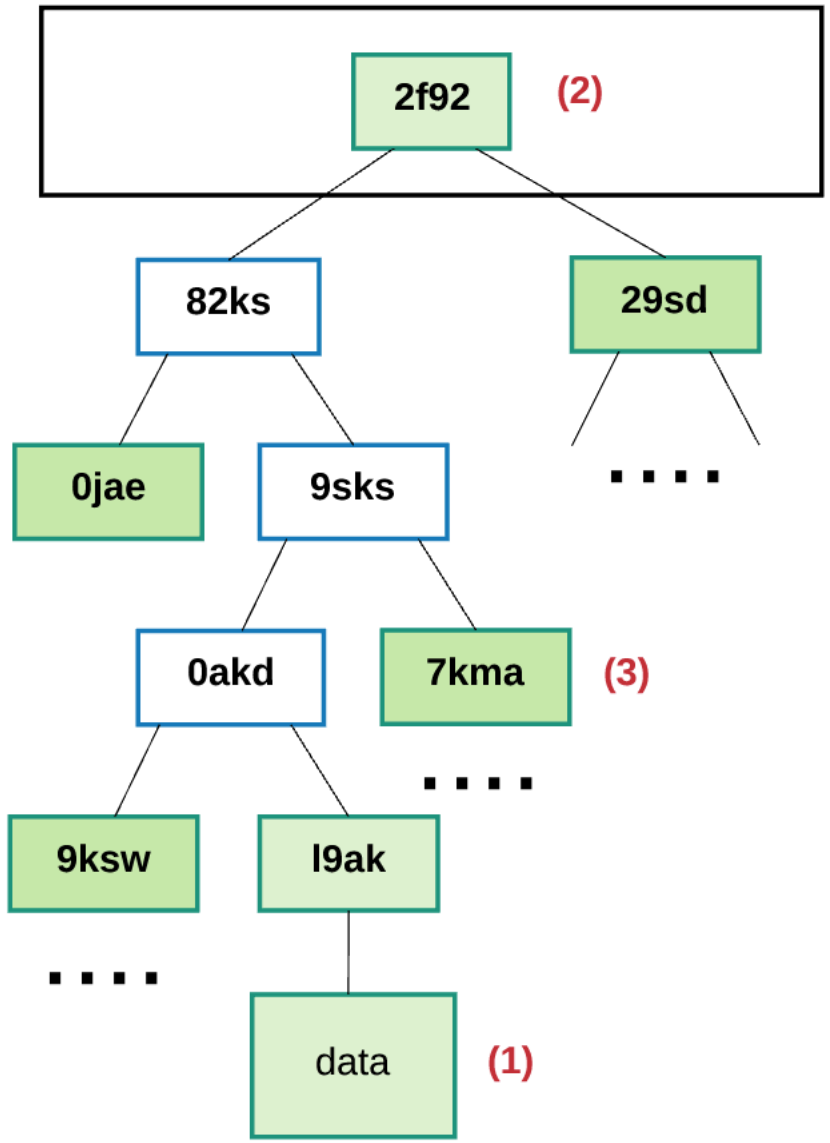
Global state

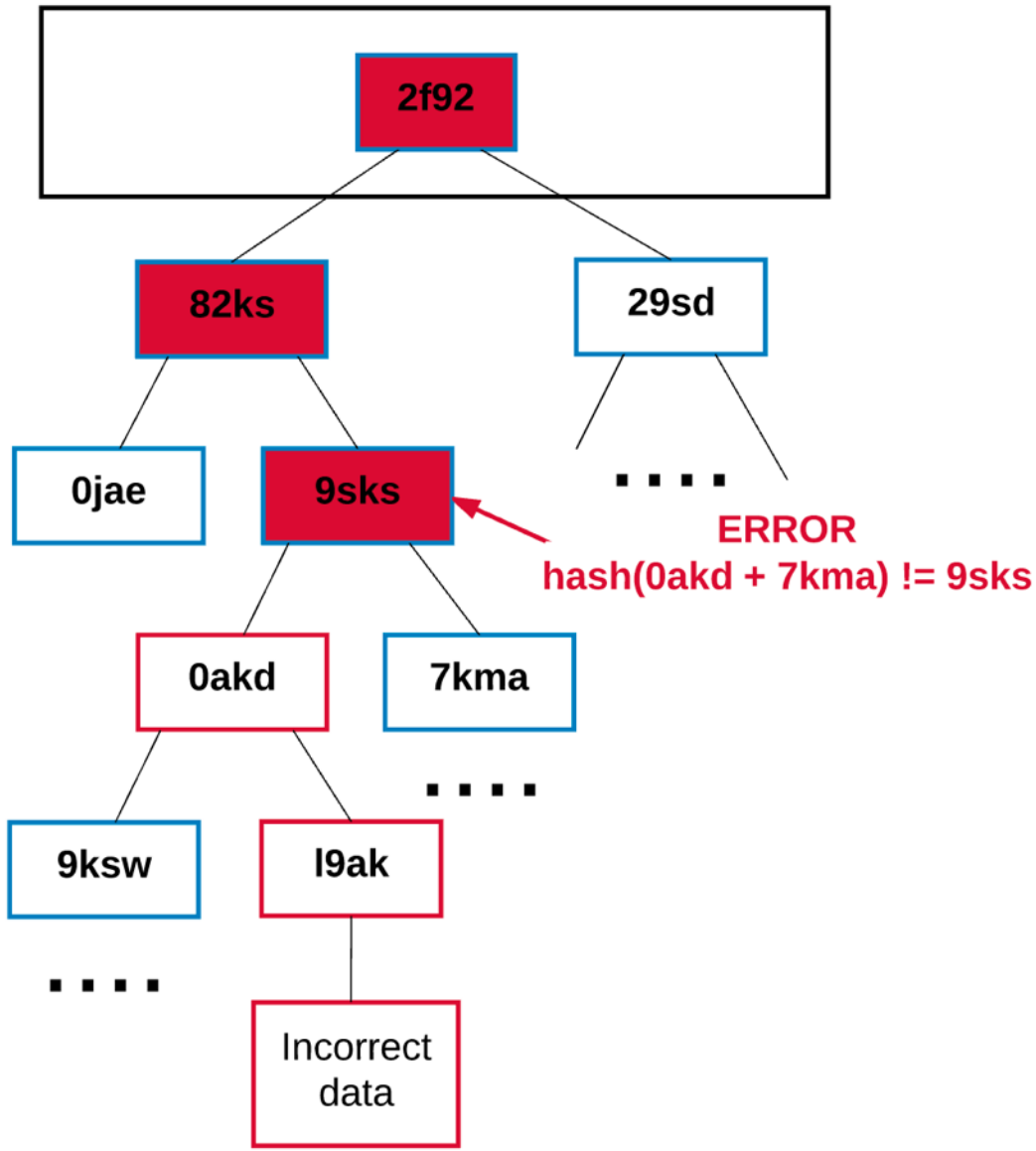


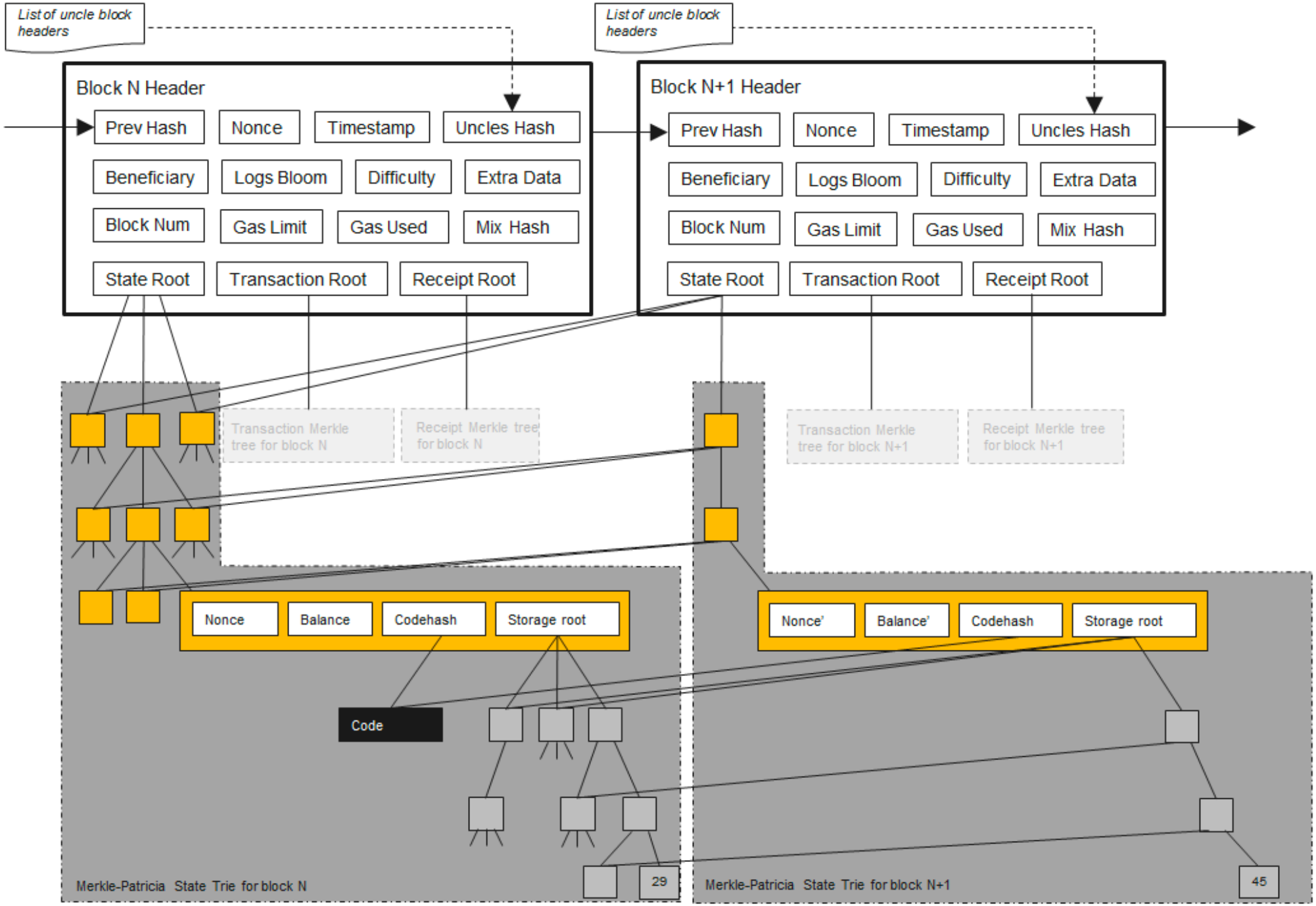
Account state

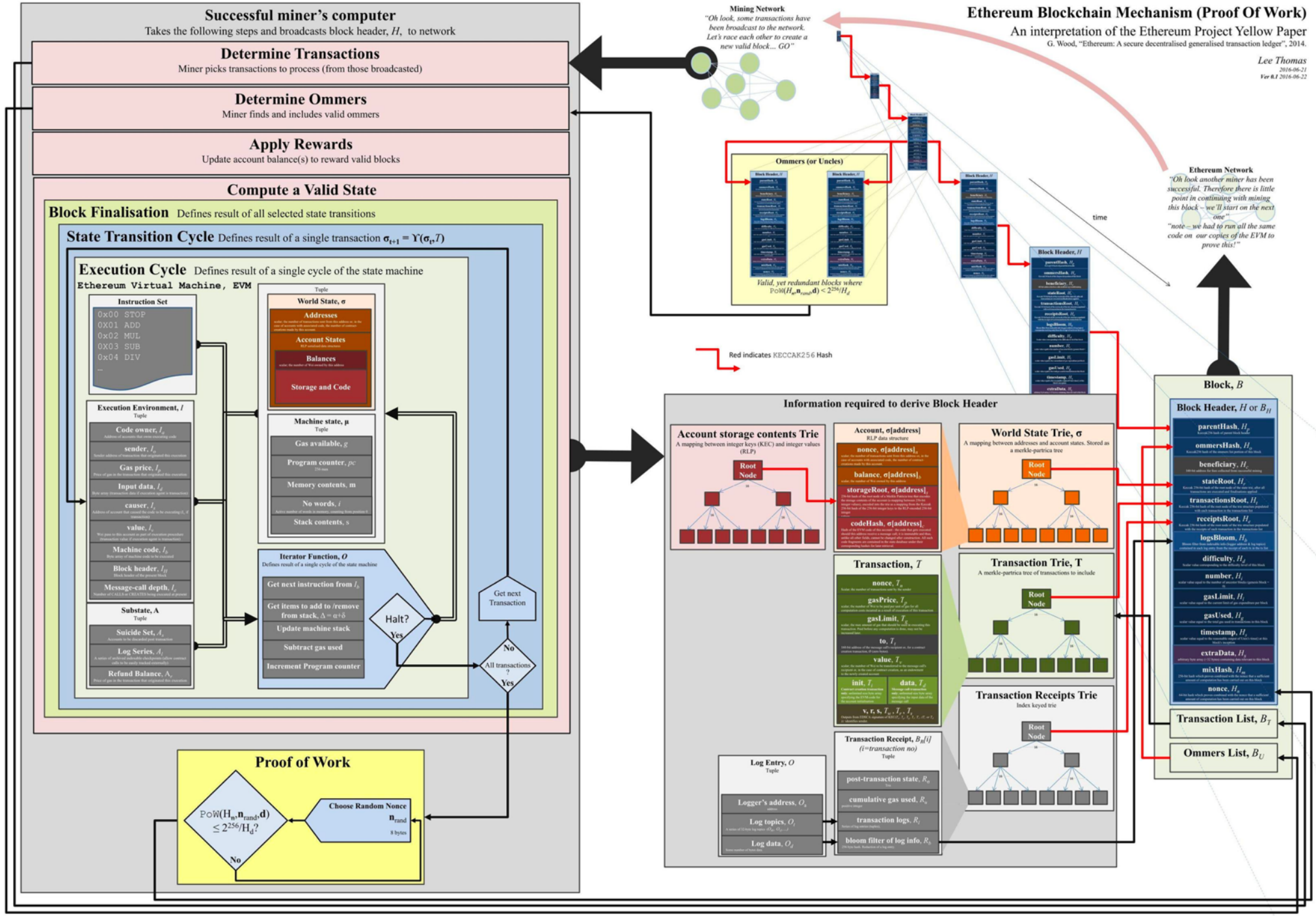
- nonce
- balance
- codeHash
- storageRoot




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
29
45
Merkle-Patricia State Trie for block N
Merkle-Patricia State Trie for block N+1


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
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
Suicide Set, As
Log Series, Aj
Refund Balance, Ar
World State, σ
Addresses
Account States
Balances
Storage and Code
Machine state, μ
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
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
Transaction Receipts Trie
Index keyed trie
Transaction Receipt, BR[i] (i=transaction no)
post-transaction state, Rσ
cumulative gas used, Ru
transaction logs, Rl
bloom filter of log info, Rb
Log Entry, O
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

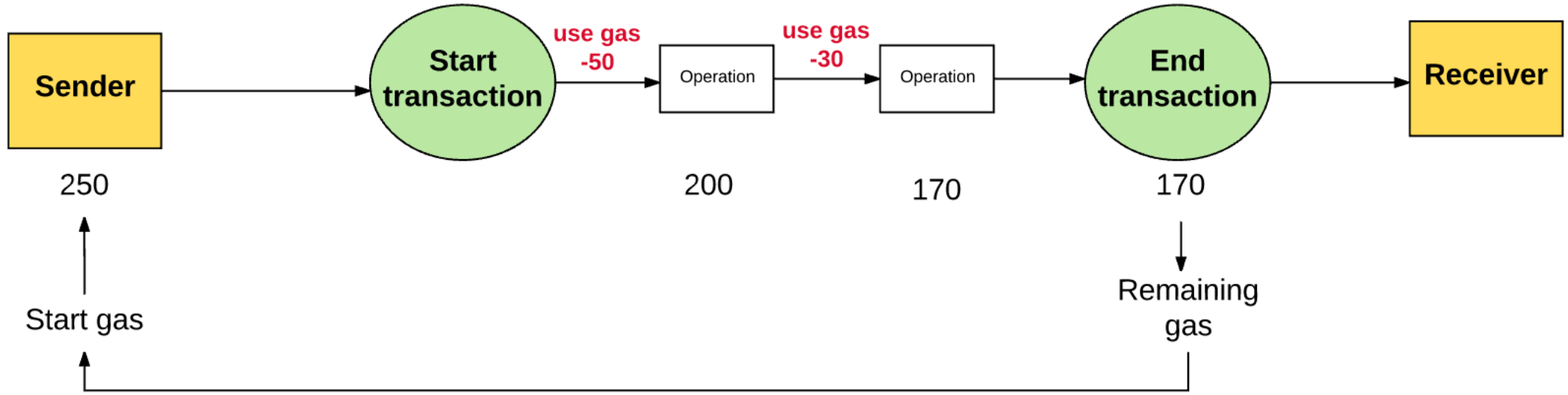
Gas & fee

- gas limit: 50000
- gas price: 20gwei

Transaction execution

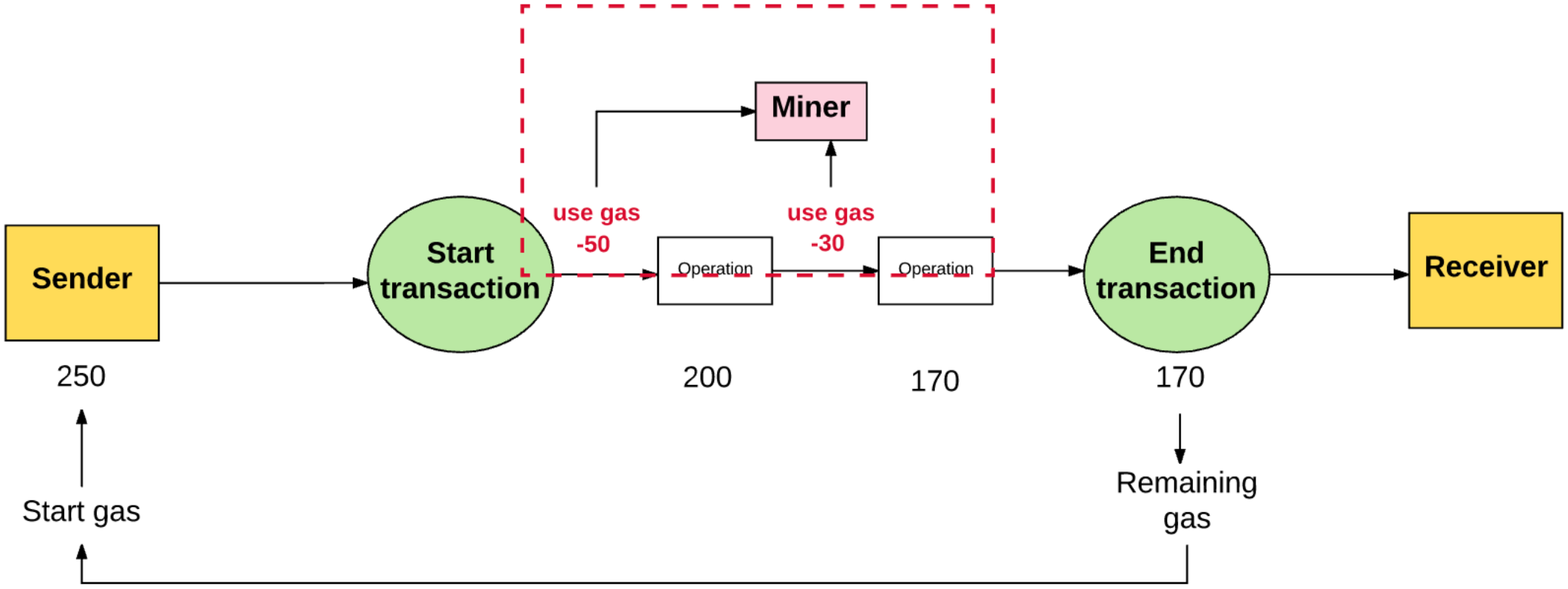
ex: 1+2=3

Add
Equal

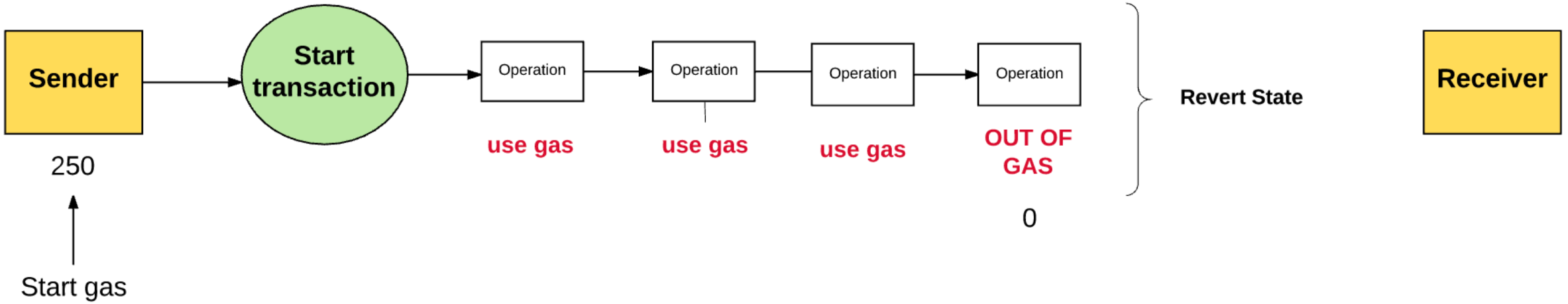


Fee

- Beneficiary address, which is the miner's address



Out of gas


Sender
Start transaction
Operation
Operation
Operation
Operation
Revert State
Receiver
use gas
use gas
use gas
OUT OF GAS
250
0
Start gas

Out of gas

Overview | Comments

Transaction Information

TxHash:	0x71e3cdd6c9f393655f19cf820d454300aa9ffdbf6b290ff3aad402273208bed4
Block Height:	4154614 (1266212 block confirmations)
TimeStamp:	240 days 10 hrs ago (Aug-14-2017 12:09:03 AM +UTC)
From:	0x9c67e141c0472115aa1b98bd0088418be68fd249 (HitBtc)
To:	Contract 0x03985e1e158e28192ec2b37e6e598d30f386574c ⚠ Warning! Error encountered during contract execution [**Out of gas**] ☹
Value:	20.10635272 Ether ($8,373.29) - [CANCELLED] ℹ
Gas Limit:	21026

單位

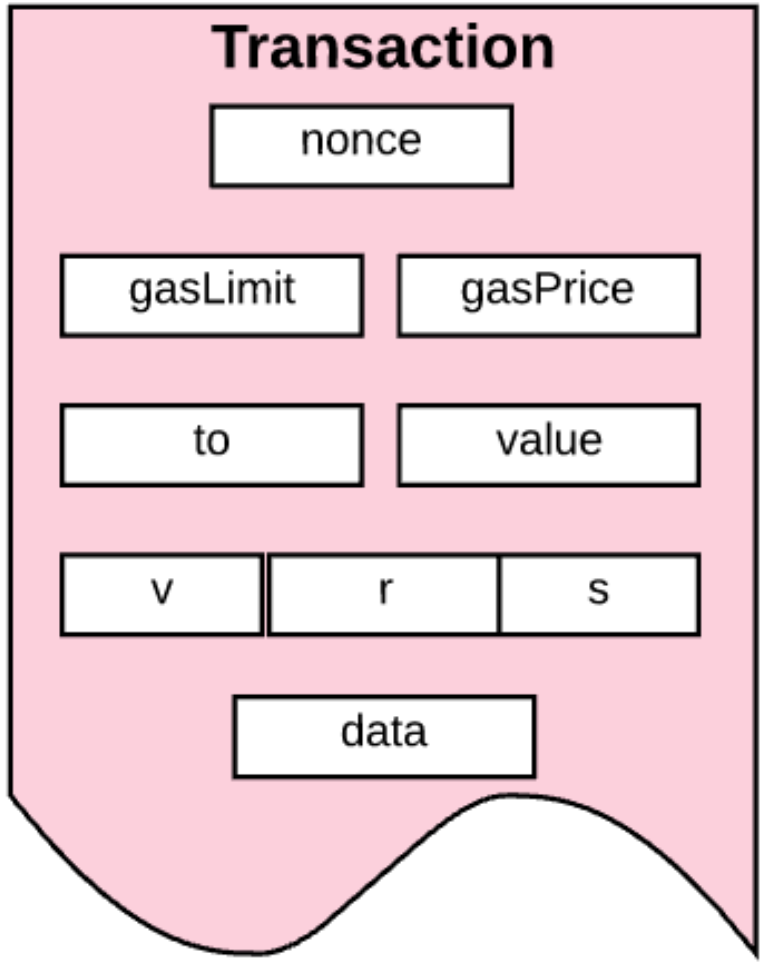
```
'wei': '1',
'kwei': '1000',
'ada': '1000',
'femtoether': '1000',
'mwei': '1000000',
'babbage': '1000000',
'picoether': '1000000',
'gwei': '1000000000',
'shannon': '1000000000',
'nanoether': '1000000000',
'nano': '1000000000',
'szabo': '1000000000000',
'microether': '1000000000000',
'micro': '1000000000000',
'finney': '1000000000000000',
'milliether': '1000000000000000',
'milli': '1000000000000000',
'ether': '1000000000000000000',
'kether': '1000000000000000000000',
'grand': '1000000000000000000000',
'einstein': '1000000000000000000000',
'mether': '1000000000000000000000000',
'gether': '1000000000000000000000000000',
'tether': '1000000000000000000000000000000'
```

Transactions

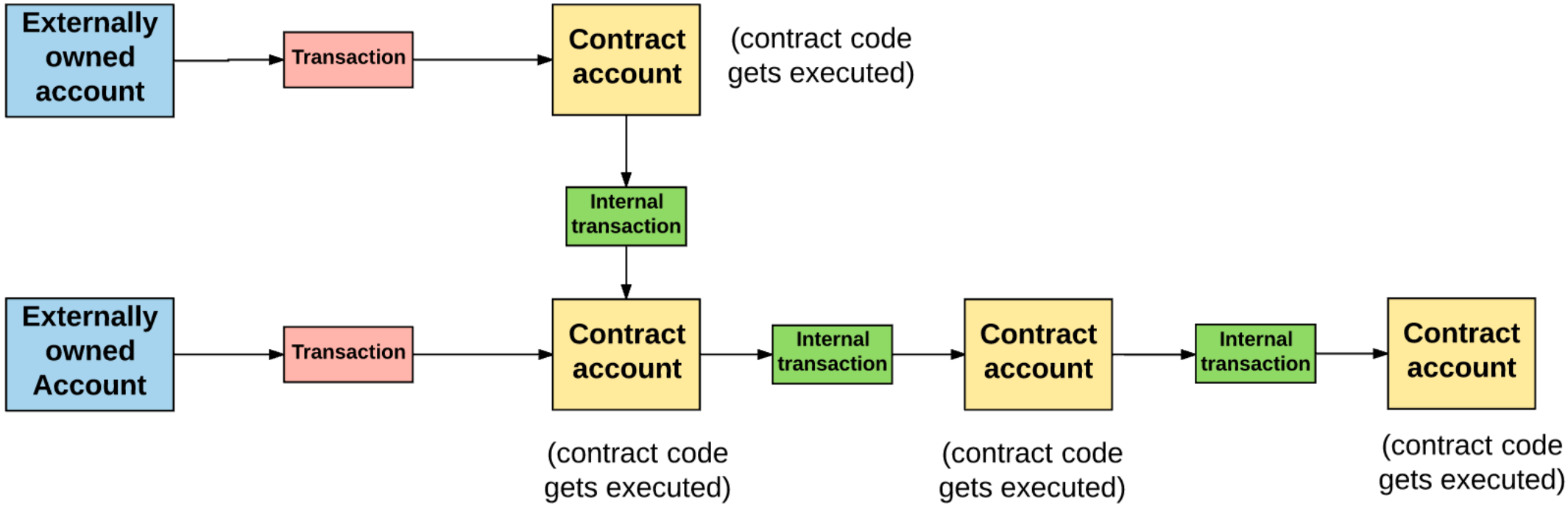
- 以太坊是一個基於交易的狀態機
- 在不同帳戶間發生交易，會讓狀態發生改變
- 交易是外部世界與以太坊內部狀態的橋樑

- 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



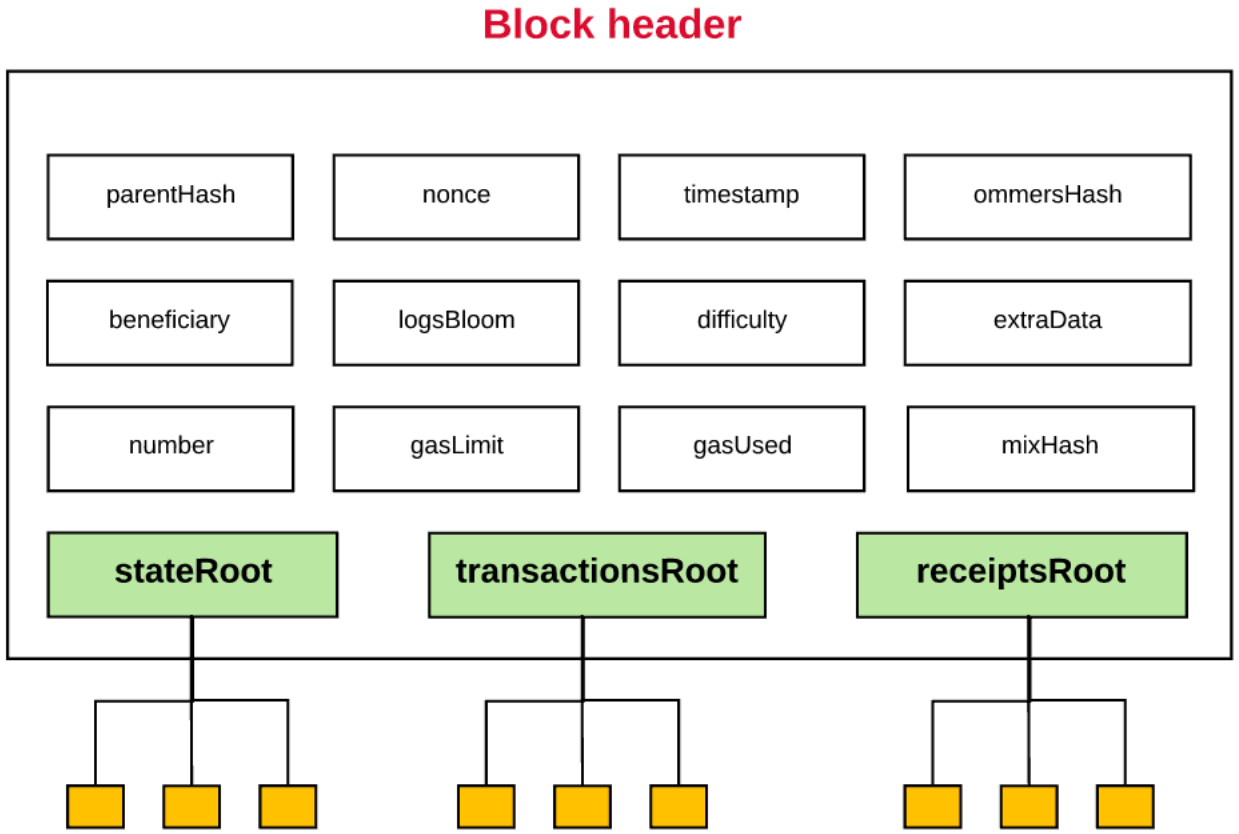
Internal transaction



Block header

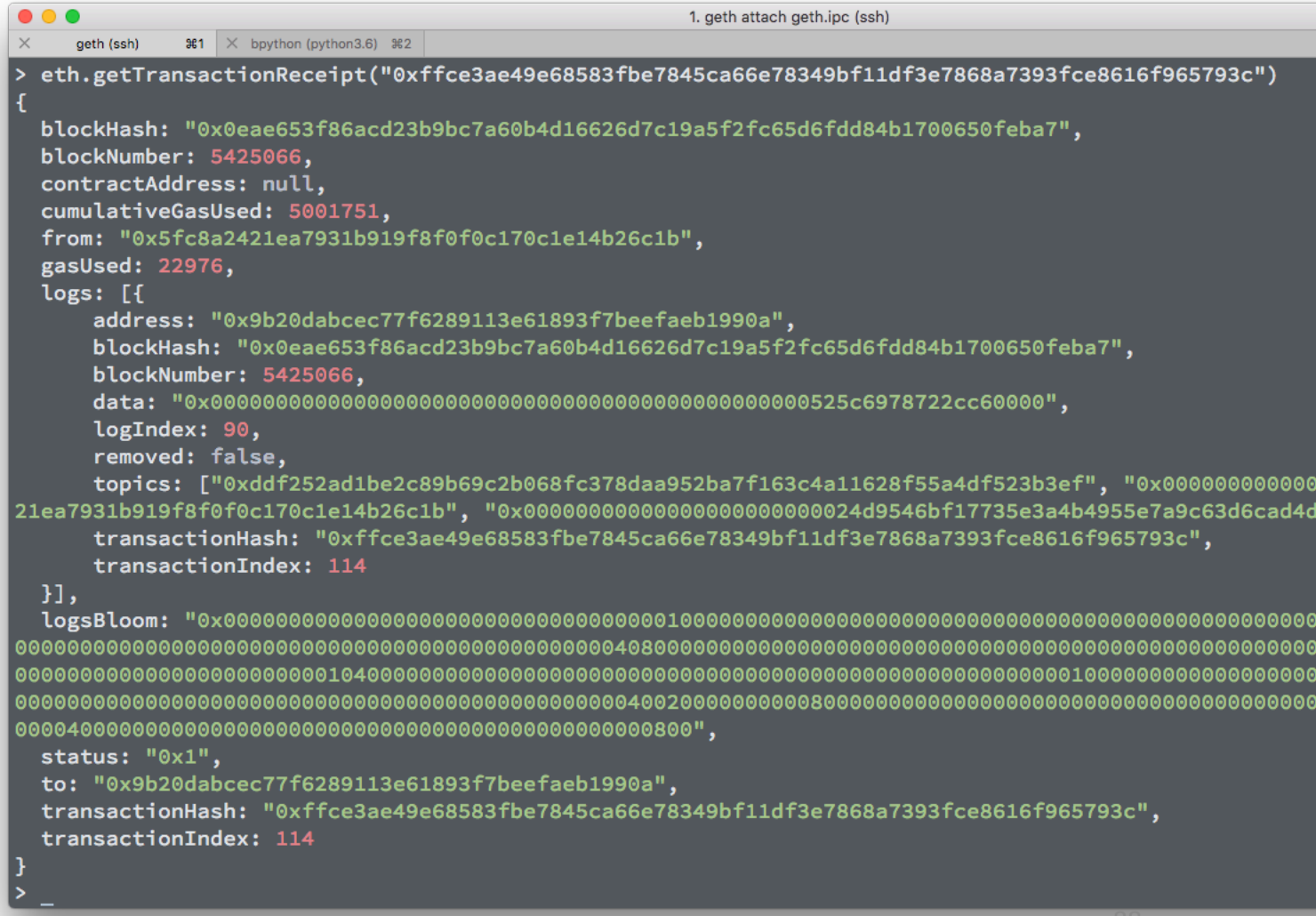
- state (**stateRoot**)
- transactions (**transactionsRoot**)
- receipts (**receiptsRoot**)

- Merkle Patricia tries



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



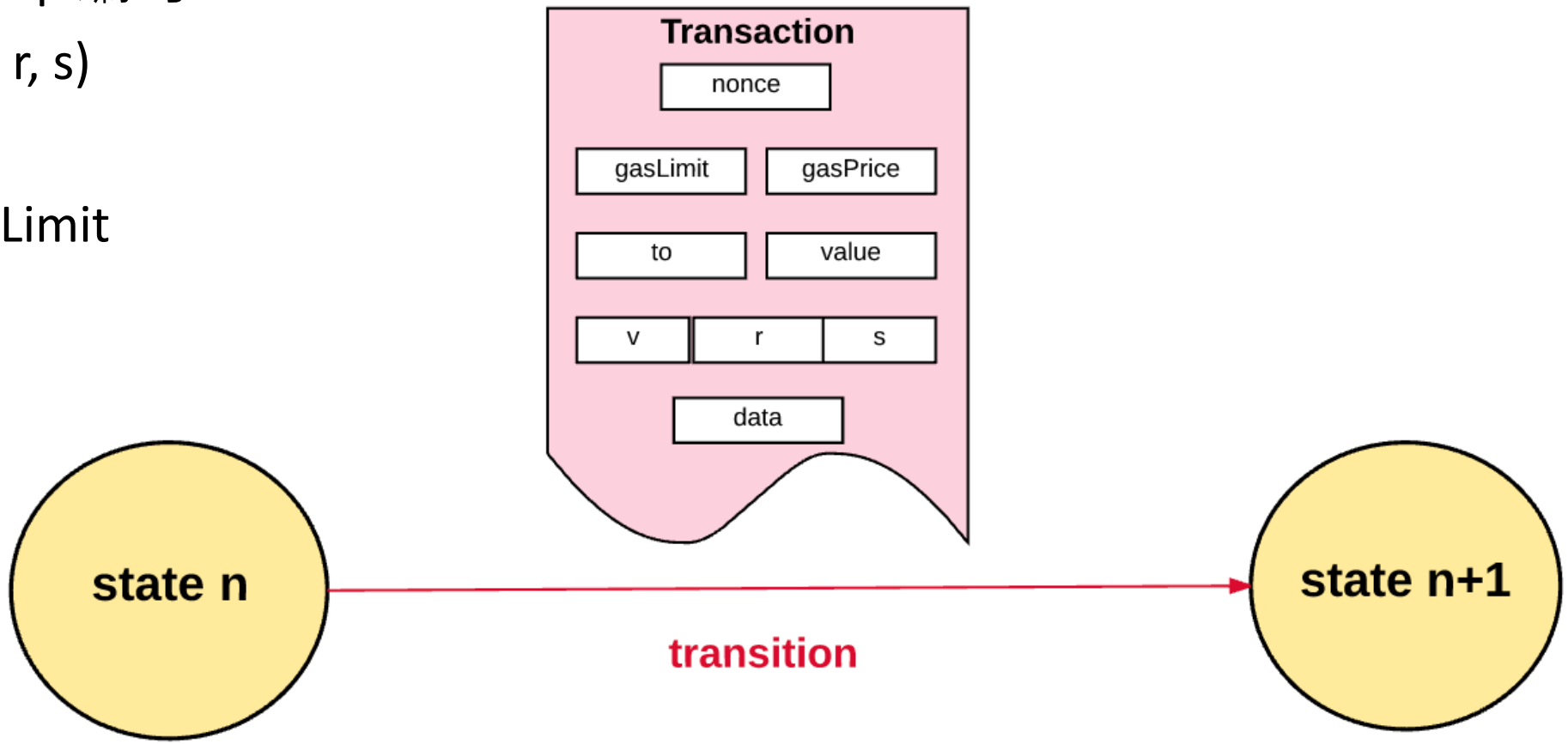
Logs

- 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

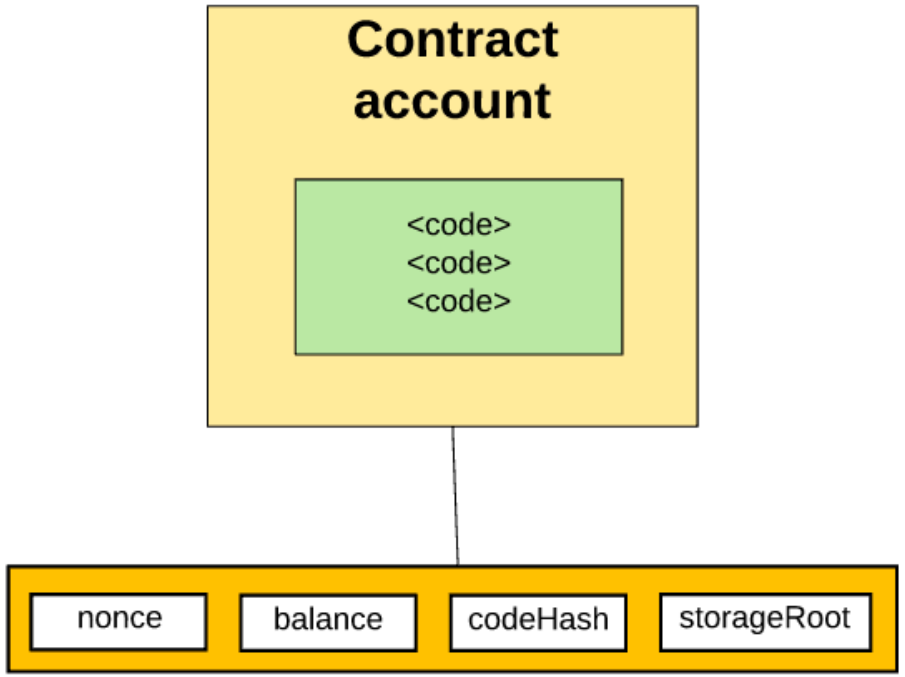
Transaction Execution

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



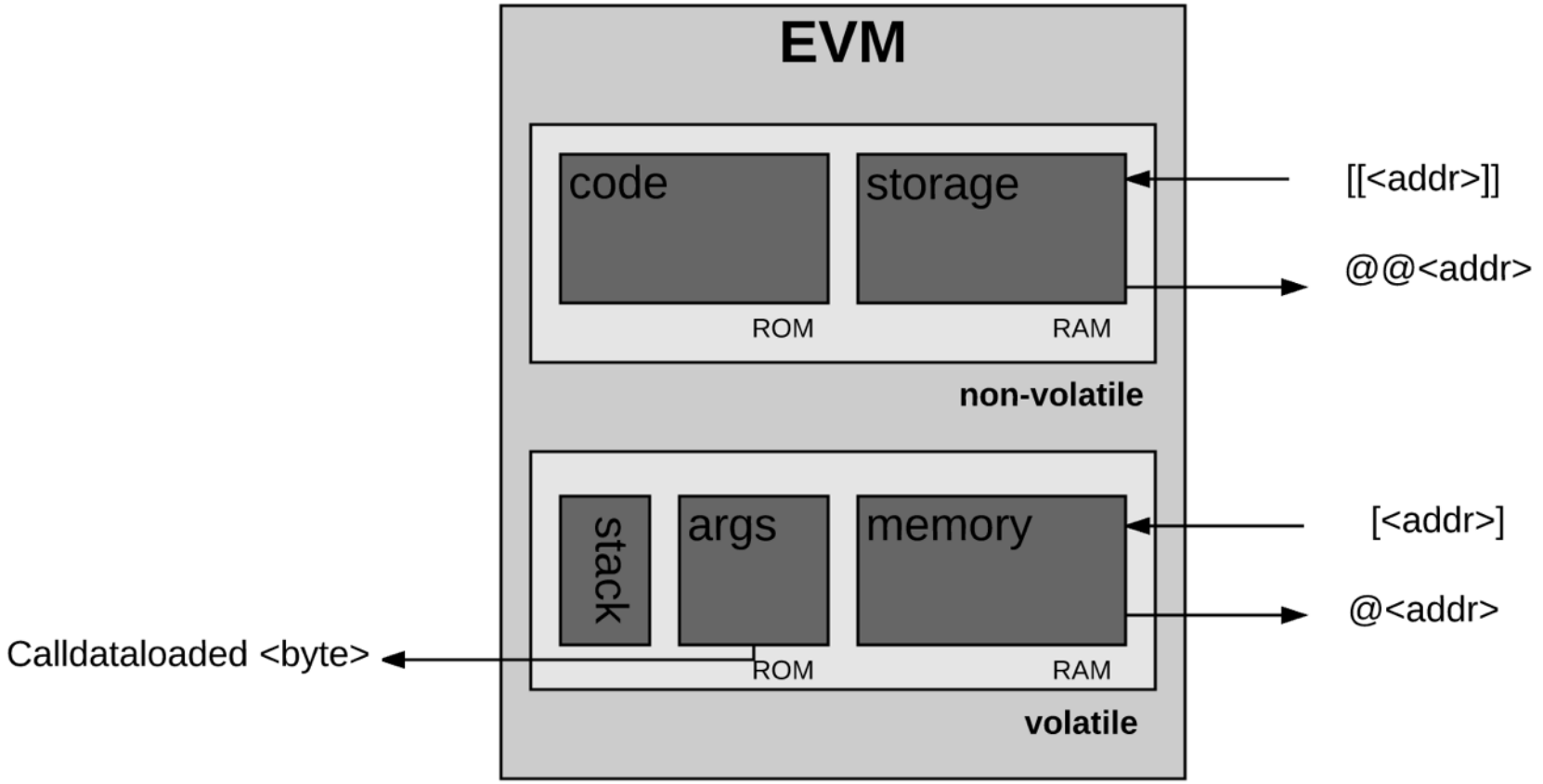
Contract creation

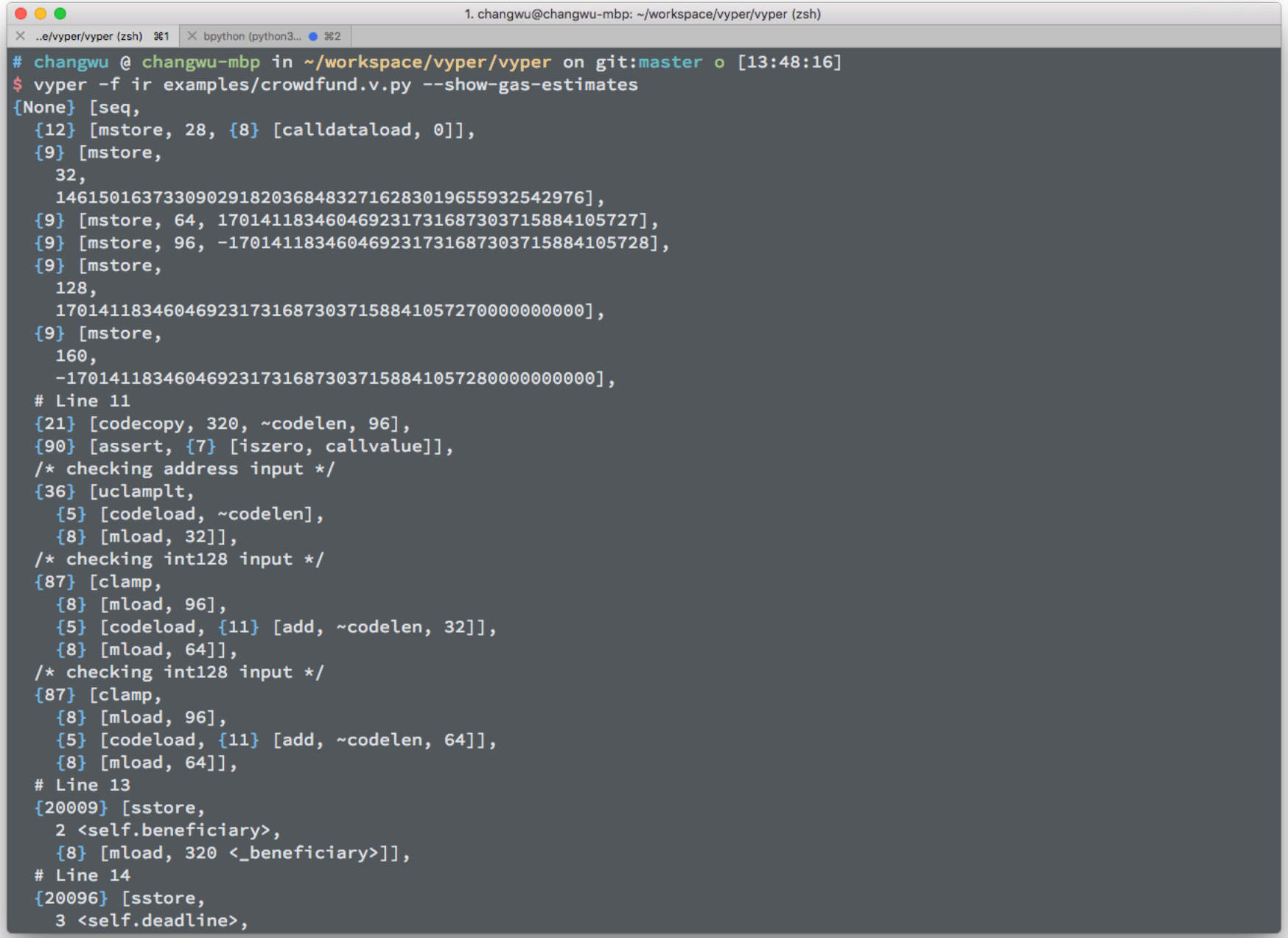
- Contract-creating transaction = Create a new contract account



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>,

Ether & Gas

- Ether: 原生貨幣
- Gas: 工作單位

- 以太坊上的“交易”像是執行一段小程式，這段程式會幫忙驗證”錢是否正確被轉出去”
- 因為是一段小程式，需要大家(礦工)幫你驗證，為了不失公平性，gas 是這段小程式需要耗費的工作量
- 因為礦工不是無償幫忙驗證，需要收點手續費

Ether & Gas

- 驗證一筆交易需要耗費工作量, 而 gas 是工作量的最小單位,

- 計價公式
  - 工錢(手續費) = 工作量 x 每單位的計價

- 假設 gasPrice = 0.00000001 Ether = 10gwei (每單位的計價)
- 假設交易總耗費 21000 gas (工作量)
- 手續費
  - 21000(gas) * 0.00000001(Ether/gas) = 0.00021 Ether (手續費)
  - 按匯率算，若 Ether ~= 400 USD, 0.00021 * 400 = 0.084 USD (手續費)

- 若指定的 gas 量太少？
- 若 gasPrice 太低會如何？

Ether & Gas

- Ether: 是貨幣單位
- Gas: 是工作量的計價單位

- 那為什麼有 Ether，還要定義 gas？還有 gasPrice？
  - 因為如果不把兩個參數解藕，當 Ether 大漲時，手續費會變得不合理

Gas Limit (startgas)

- 每筆交易在執行後會耗費工作量 (gasUsed)
- 在使用者提交交易前，需要設定 gasPrice 與 gasLimit

- 假設設定 gasLimit=35000，但實際用量 gasUsed=21000 (程式真正消耗的工作量)
- 這個 gasLimit 是個系統保護機制，因為在電腦科學上，很難保證或衡量一段程式，執行後一定會終止，如果不能終止就會造成電腦卡住。又或是如果駭客發現一個漏洞，本來一段程式碼是在做轉錢的動作，因為沒有設定終止的條件，最終可被利用將錢無止盡的轉出

- 假設 21000 的工作量是將你的錢轉出 5 ETH ，假設想再次轉帳 5 ETH ，第二次會被擋下，因為需要額外 21000 的工作量，因為起初指定的 gasLimit 只有 35000。(實際上轉兩次 5 ETH 的工作量很小，這裡只是舉例)

Gas Limit (startgas)

- 如果交易在執行過程，耗費的工作量超過指定的 gasLimit=35000

- 這筆交易會視為無效，並回復到之前的狀態
- 另外，礦工會收走手續費 ，35000(gas) * 0.00000001(Ether/gas) = 0.00035 Ether

- 錯誤訊息: transaction is out of gas

Ether & Gas

- 手續費: (startgas - gas_remaining) * gasprice
- gas_remaining: 退回給使用者

- 基本的轉帳交易: 21000 gas
- 複雜的合約交易會用掉較多 gas
- 一筆交易的 gasLimit 可能會接近一個 Block 的 gasLimit
- 但是若交易的 gasLimit 高於 Block 的 gasLimit，則該交易不會進到區塊

EVM (Ethereum virtual machine)

- The Ethereum virtual machine is the engine in which transaction code gets executed, and is the core differentiating feature between Ethereum and other systems.
  - Simplicity
  - Total determinism
  - Space savings
  - Specialization to expected applications
  - Simple security
  - Optimization-friendliness

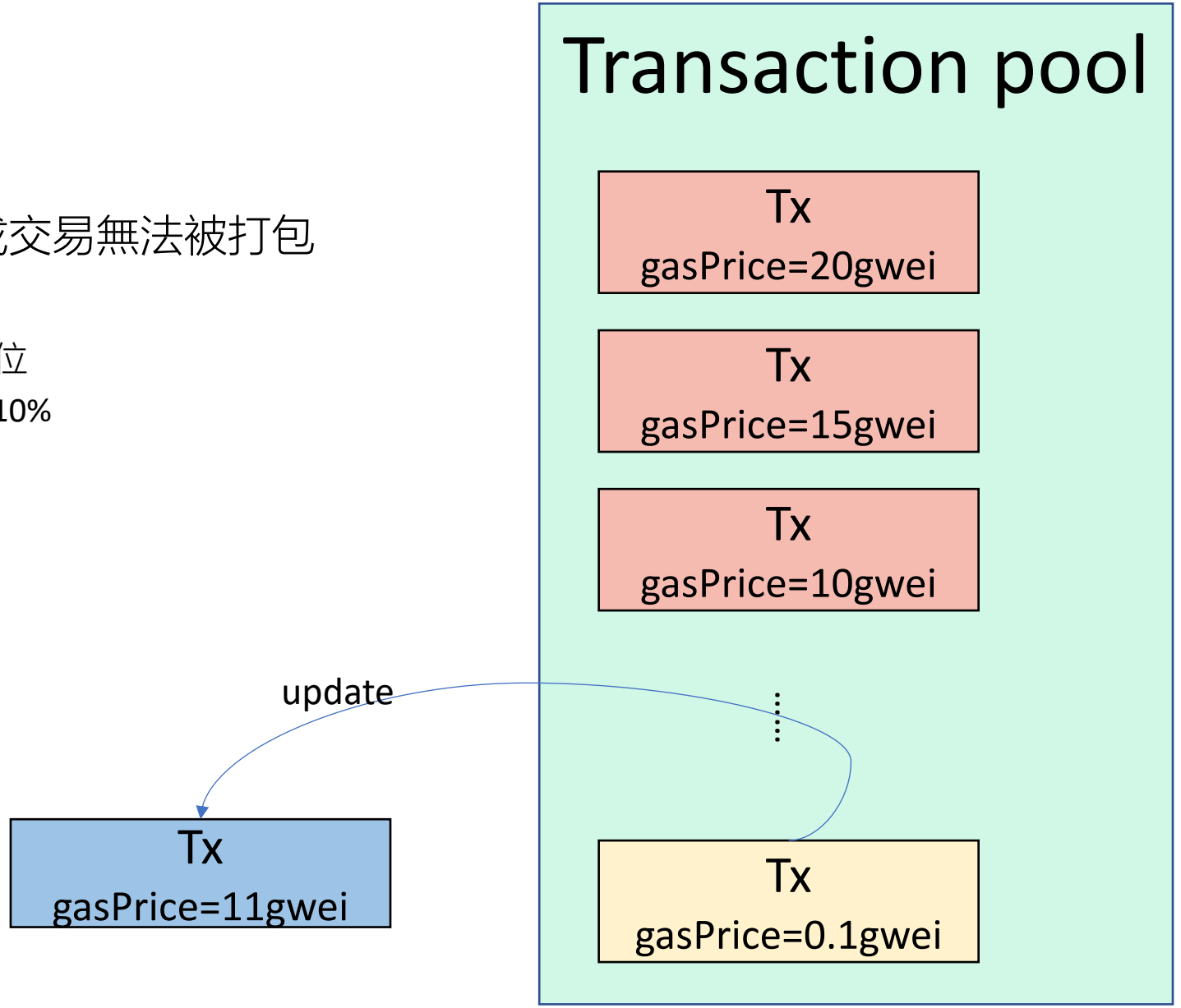
https://github.com/djrtwo/evm-opcode-gas-costs/blob/master/opcode-gas-costs_EIP-150_revision-1e18248_2017-04-12.csv#L62

EVM (opcode - gas cost)

Value	Mnemonic	Gas Used	Subset	Removed from stack	Added to stack	Notes
0x00	STOP	0	zero	0	0	Halts execution.
0x01	ADD	3	verylow	2	1	Addition operation
0x02	MUL	5	low	2	1	Multiplication operation.
0x03	SUB	3	verylow	2	1	Subtraction operation.
0x04	DIV	5	low	2	1	Integer division operatio
0x05	SDIV	5	low	2	1	Signed integer division c
0x06	MOD	5	low	2	1	Modulo remainder opera
0x07	SMOD	5	low	2	1	Signed modulo remainde
0x08	ADDMOD	8	mid	3	1	Modulo addition operati
0x09	MULMOD	8	mid	3	1	Modulo multiplication op
0x0a	EXP	(exp == 0) ? 10 : (10 + 10 * (1 + log256(exp)))		2	1	Exponential operation.

交易卡住

- 交易使用過低的 gasPrice，造成交易無法被打包
- 解決
  - 重新送出原交易, 調整 gasPrice 欄位
  - 手續費必須高於原交易手續費的 10%
- [x] 10GWei —> 10.5GWei
- [o] 10GWei —> 11GWei



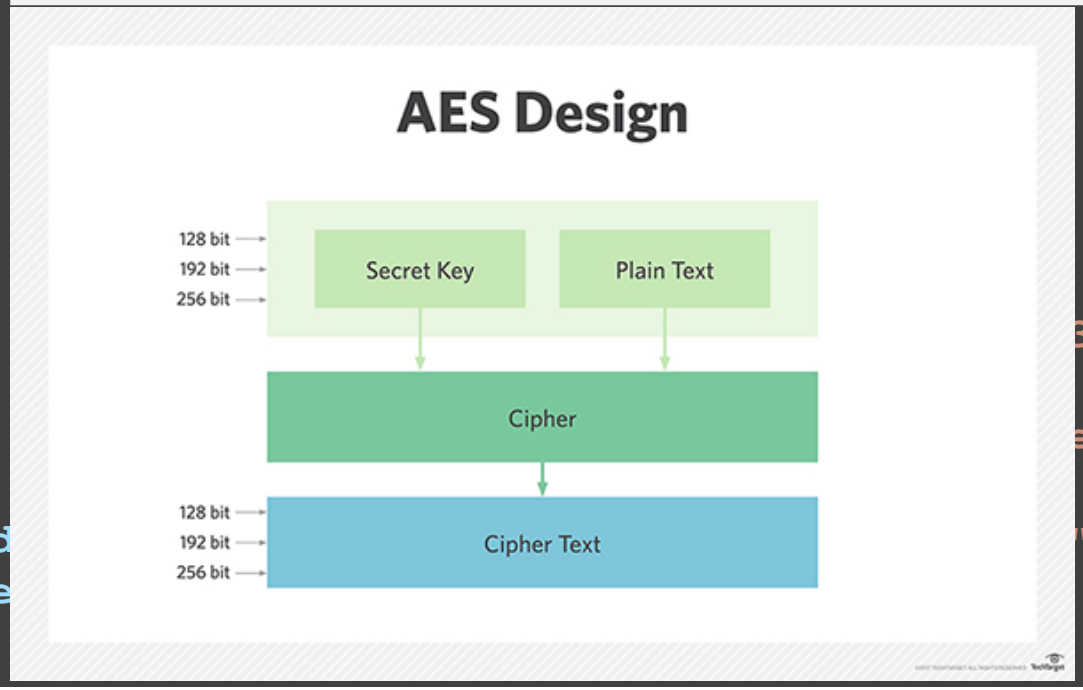
https://etherscan.io/tx/0x971c20f805eee8a256d4710b5a1980144a90cf0d8ef018ff2fc3dfe483456f67

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

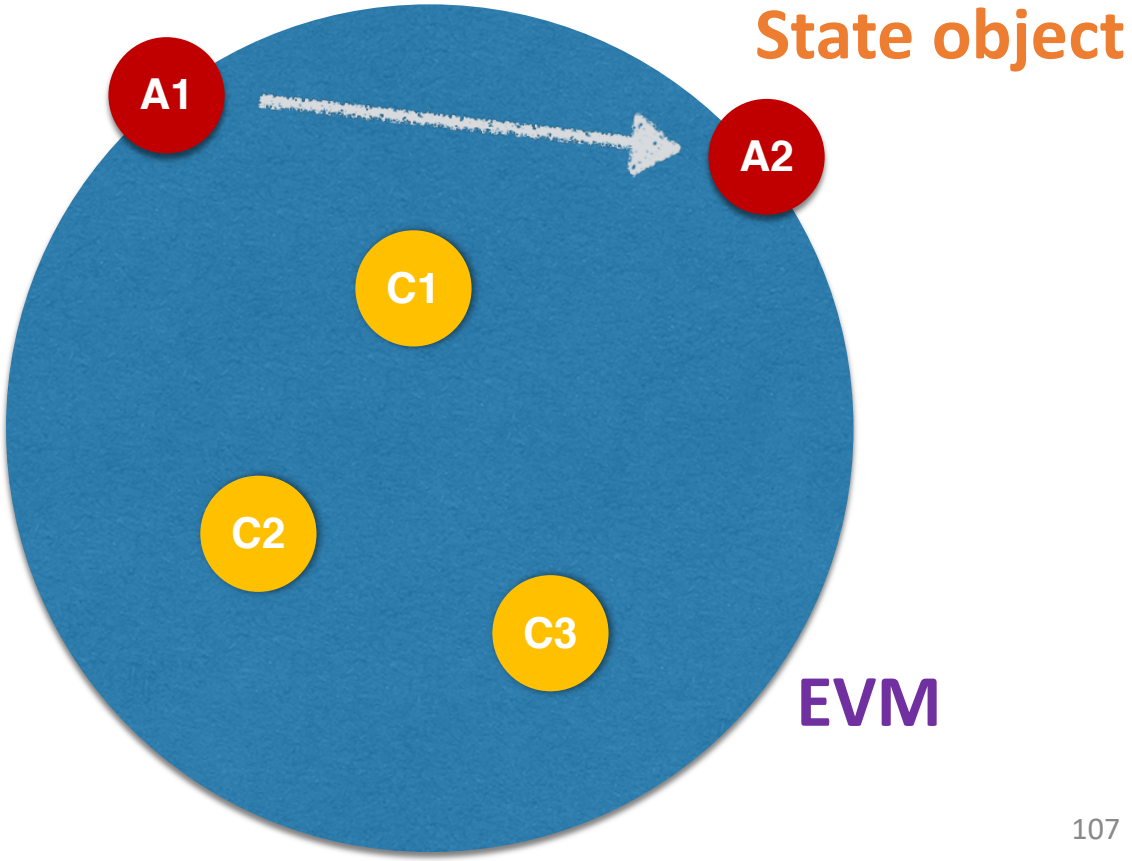
為了檢查使用者的 password 正確與否, 使用 mac 來驗證一致性

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

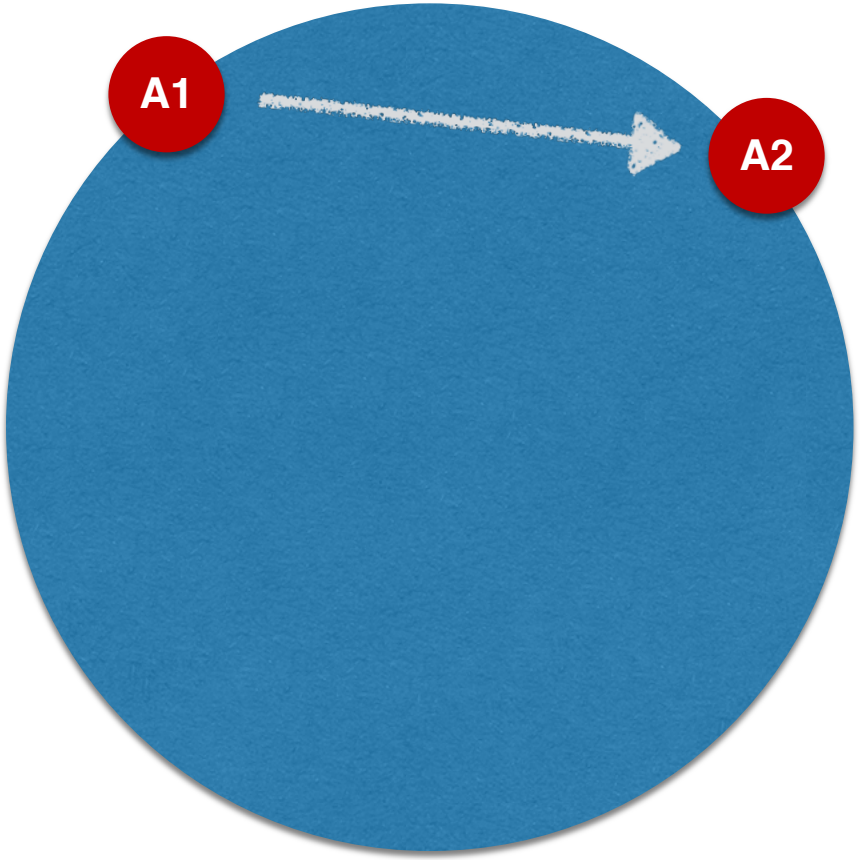
- 外部帳戶 (Externally owned account, EOA)
- 合約帳戶 (Contract account)
  - Nonce
  - Balance
  - codeHash
  - storageRoot



Ethereum

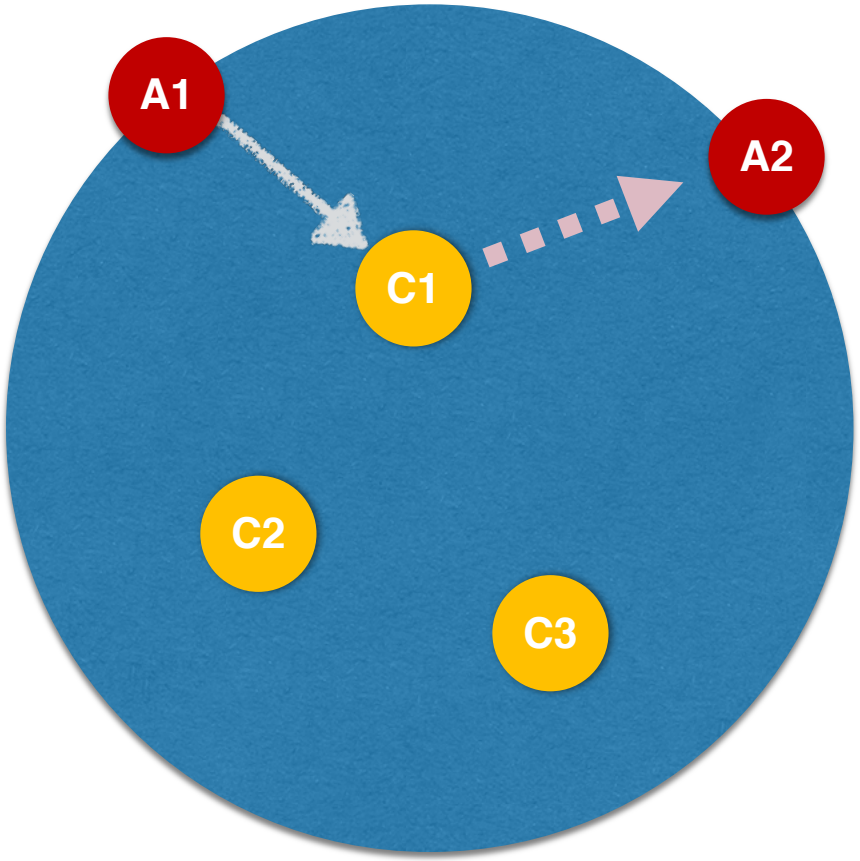
address: '0x556657e4691afe833b9c6978056706fbe095f4ad'
address: '0x556662d50c8789885fb5dc2263c332134fdf95c8'

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



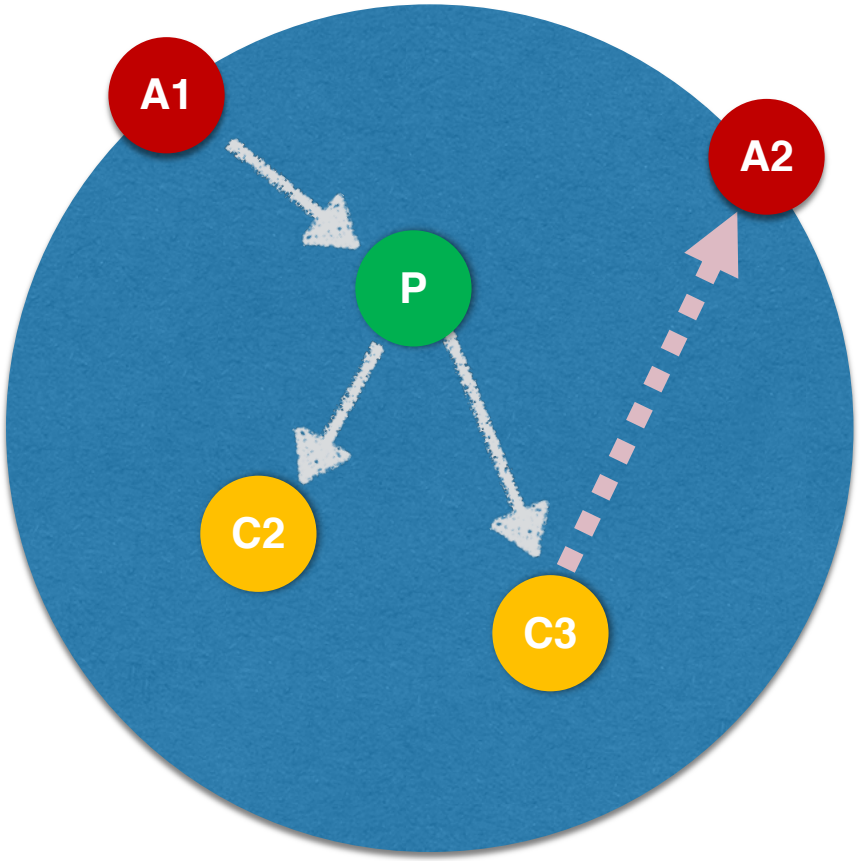
Ethereum

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



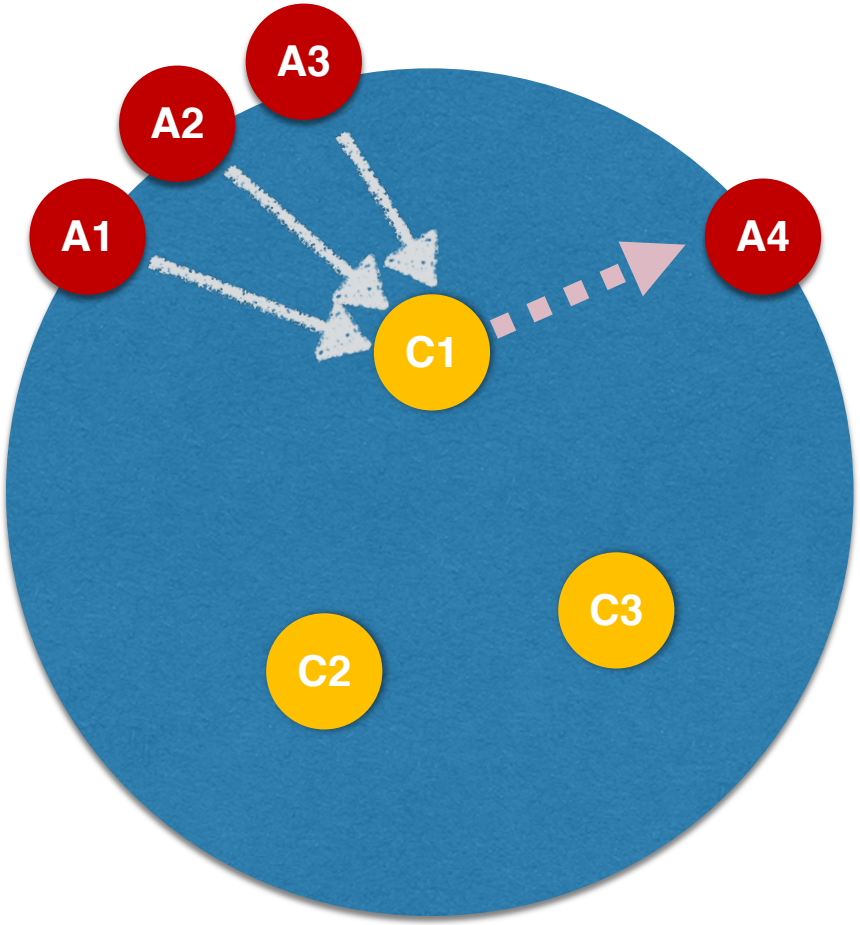
Ethereum

- 連鎖反應 (Proxy, Controller)
- 替換合約



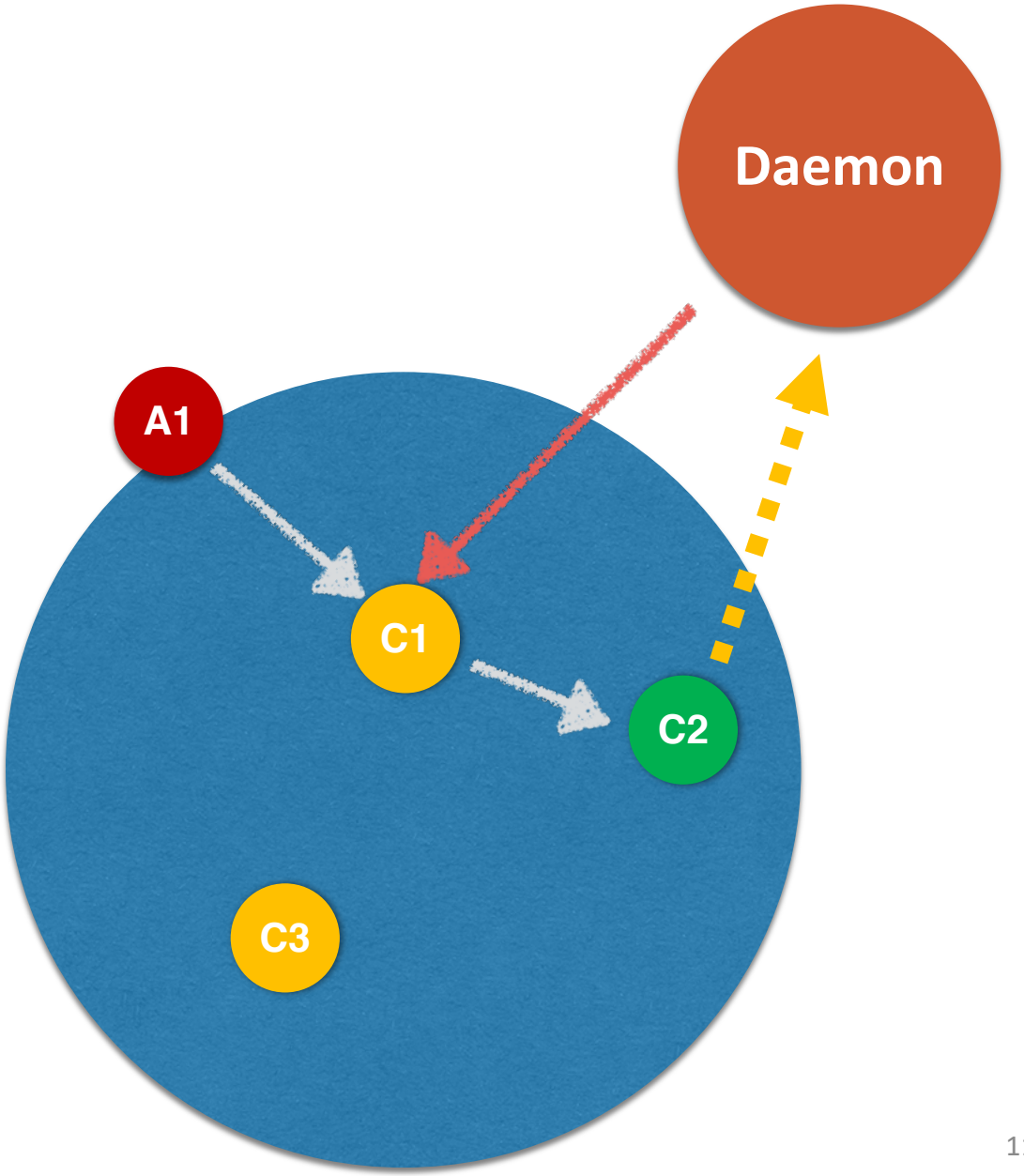
Ethereum

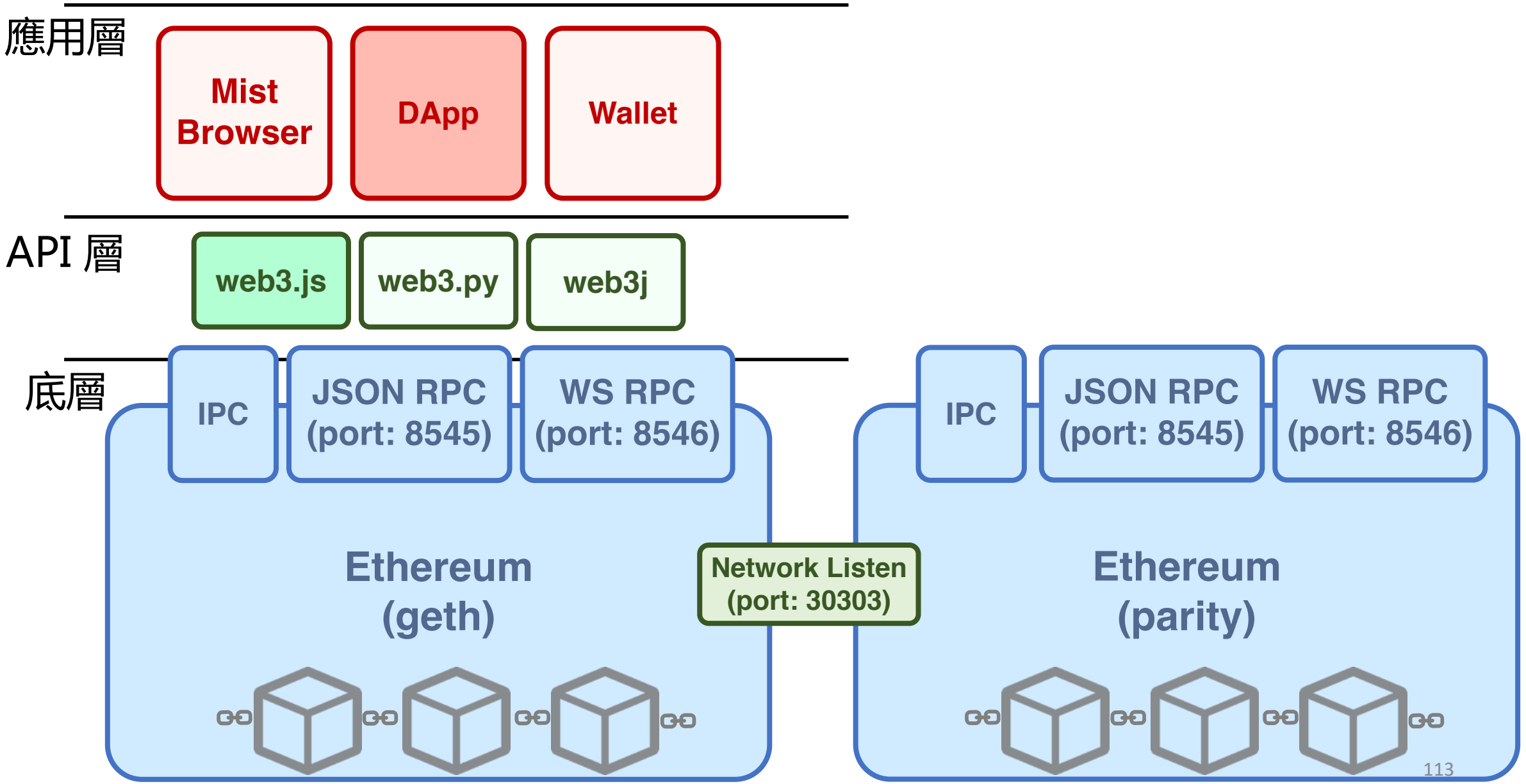
- 多簽 (multi-sig) 透過 Proxy 合約

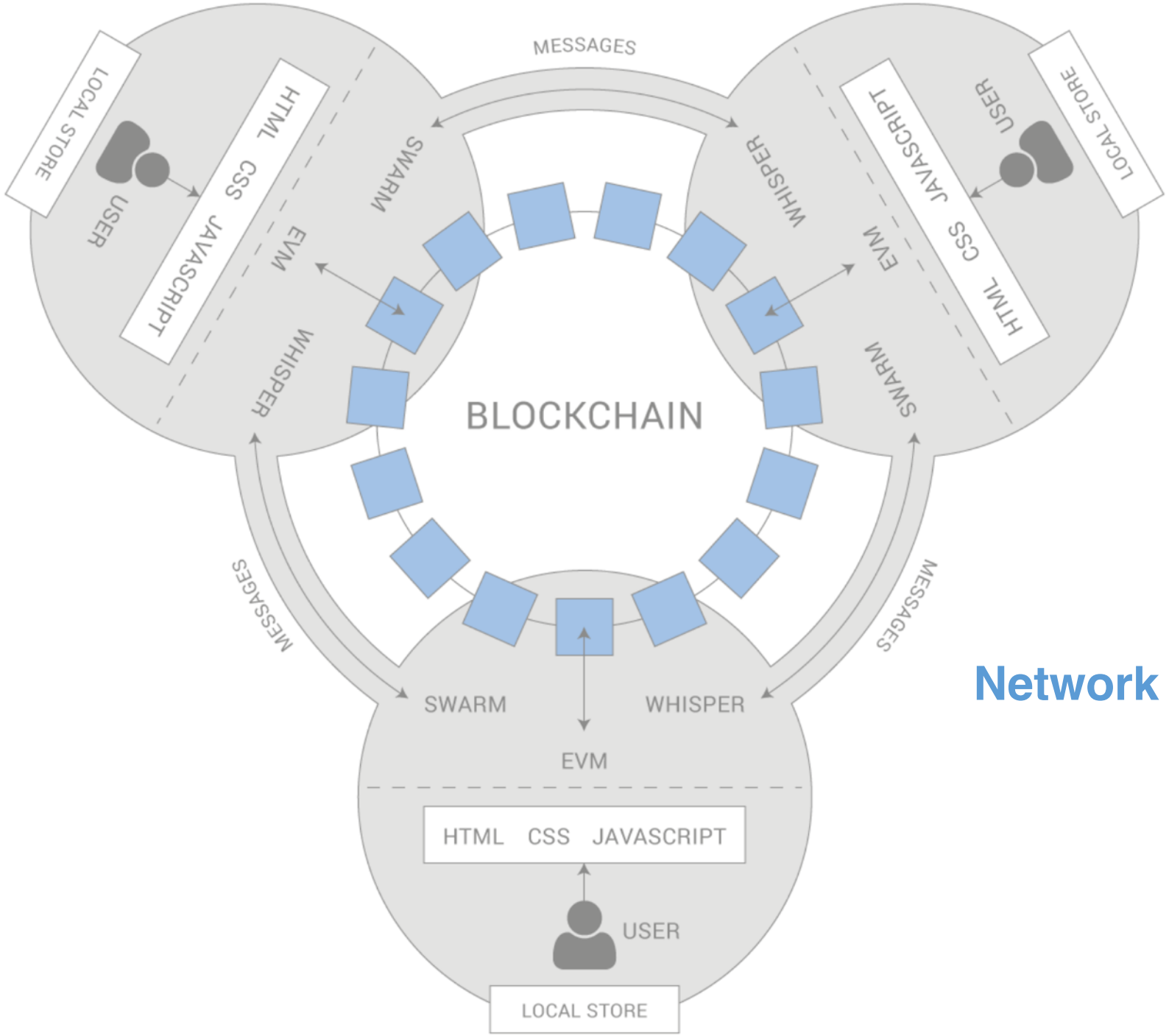


Ethereum

- Oracle
- Feed Data & Proof

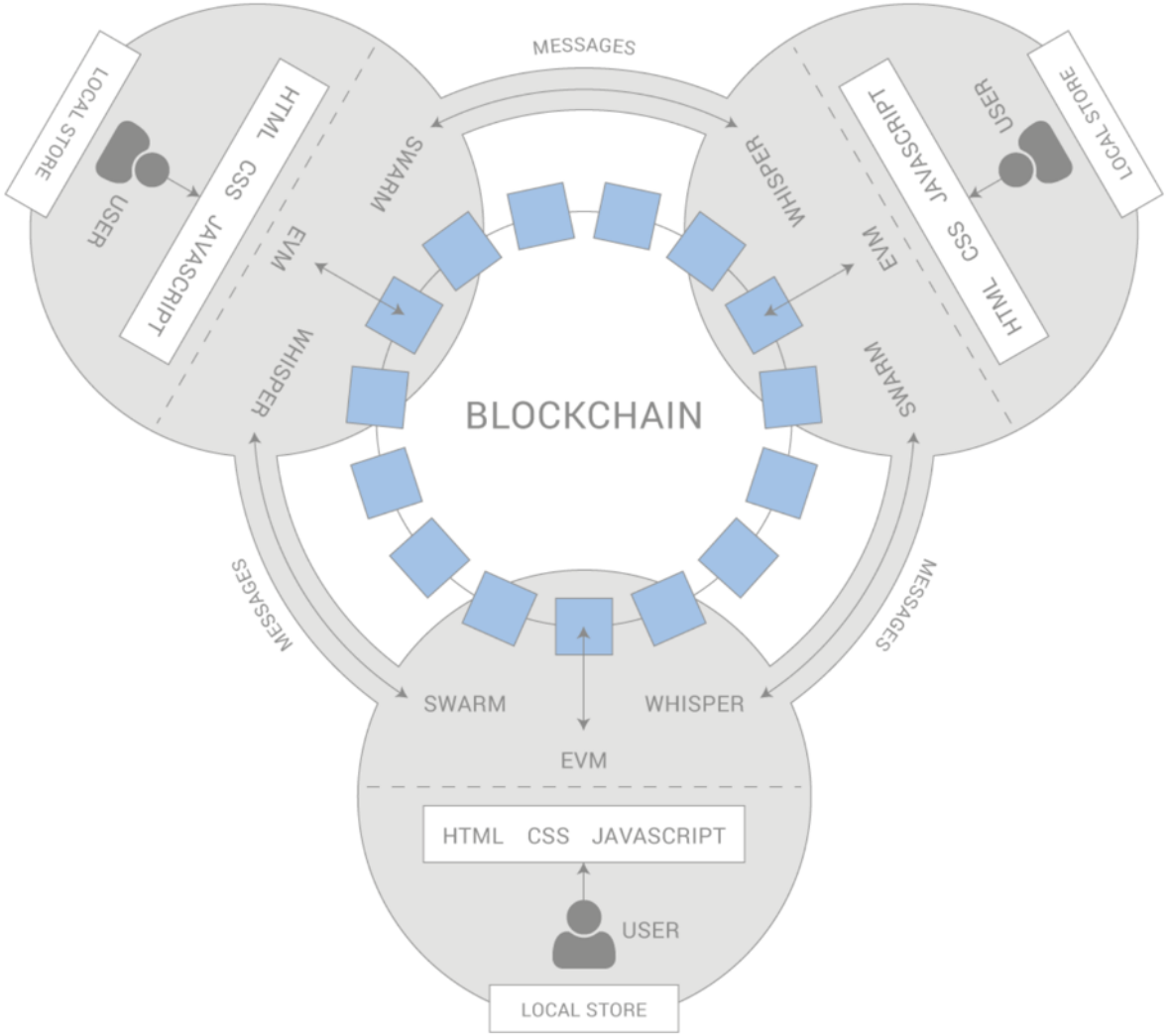



應用層
Mist Browser
DApp
Wallet
API 層
web3.js
web3.py
web3j
底層
IPC
JSON RPC (port: 8545)
WS RPC (port: 8546)
Ethereum (geth)
Network Listen (port: 30303)
IPC
JSON RPC (port: 8545)
WS RPC (port: 8546)
Ethereum (parity)

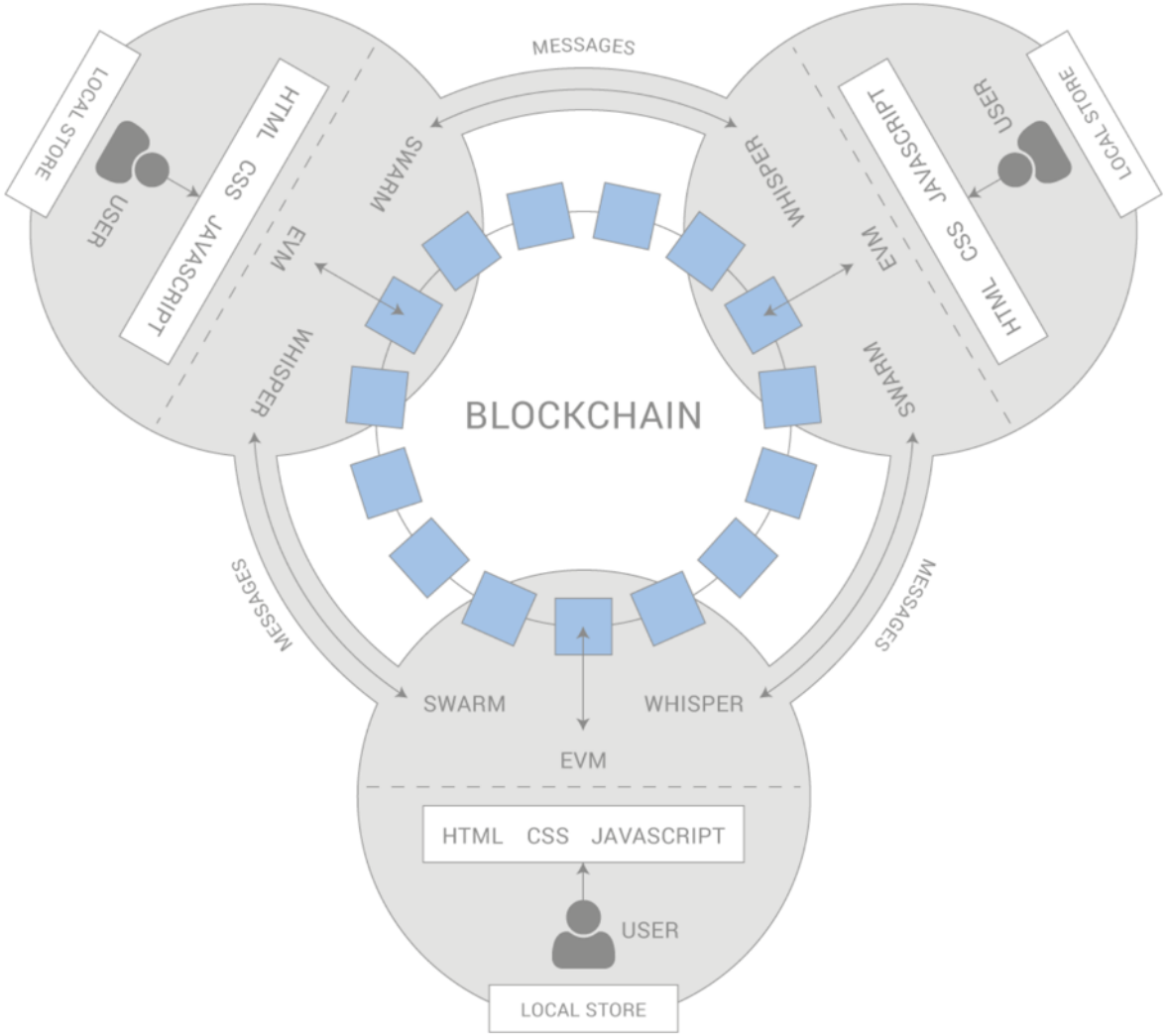

MESSAGES
LOCAL STORE
USER
HTML CSS JAVASCRIPT
EVM
SWARM
WHISPER
BLOCKCHAIN
WHISPER
SWARM
EVM
HTML CSS JAVASCRIPT
USER
LOCAL STORE
MESSAGES
MESSAGES
SWARM
WHISPER
EVM
HTML CSS JAVASCRIPT
USER
LOCAL STORE

Network ID

Mainnet

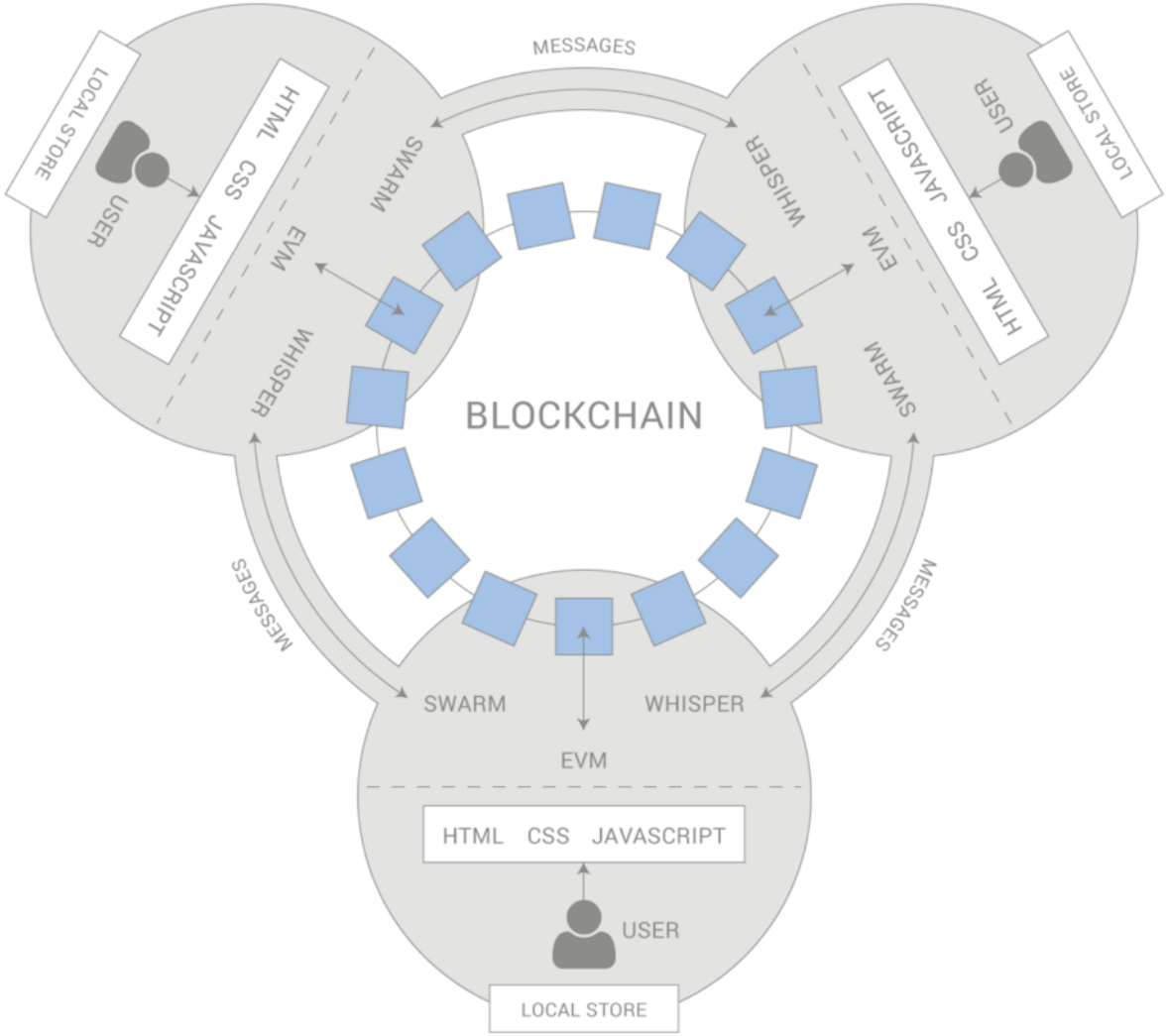


Testnet

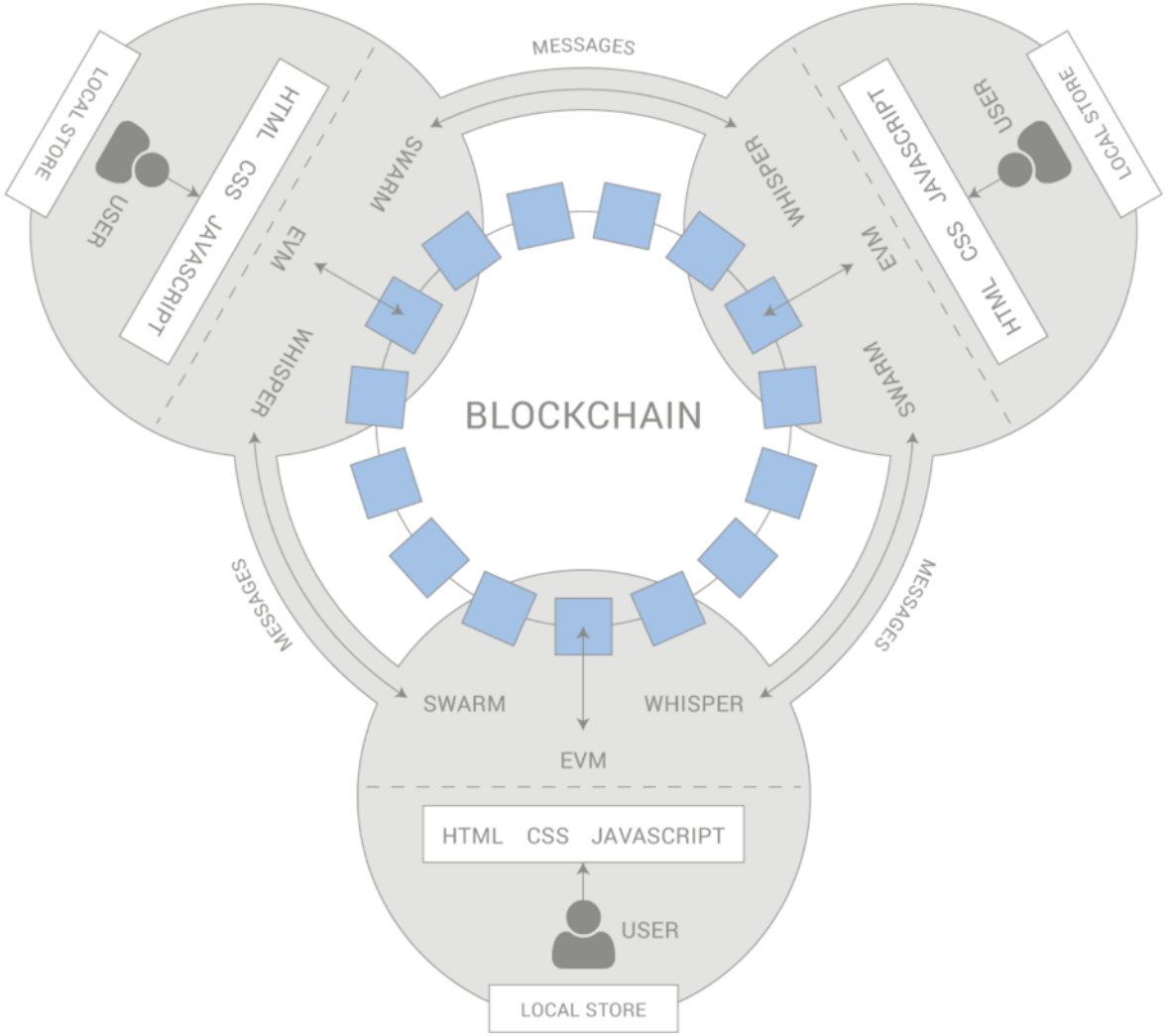


公網上所使用的鏈

Dev chain



Private chain



開發上所使用的鏈，方便開發者測試

合約開發工具介紹

- 一般開發
  - 安裝 Ethereum-client (geth, parity etc)
  - 用 Solidity 撰寫合約
  - 透過 Solidity compiler 進行編譯
  - 以 web3.js 將合約送出佈署
- 需要有測試鏈或以太鏈(測試昂貴)
- 步驟繁雜

Ethereum Dev (1/2)

- 搭建 Ethereum 私有鏈
- 運行兩個 node 讓彼此互連
- 熟悉 geth JavaScript Console
- 建立其他帳號, 建立交易, 觀察 block
- 運行私有鏈的 miner
- RPC protocol 使用與串接
- Ethereum Block explorer
- Smart Contract on Ethereum
- Mist
- Solidity
- DApp programming

Ethereum Dev (2/2)

- Track 1
  - Introduction to Smart Contracts
  - Setting up the development environment with truffle and testrpc
  - Reading documentation and reference resources
  - Watching tutorials on building Smart Contracts
  - Smart contract security and testing
- Track 2
  - Setting up a development environment with ReactJS and Firebase
  - ReactJS and firebase Hello World
  - Research on Ethereum and Dapp
  - Development of a non-trivial Smart Contract Application
  - Building user-centric interfaces
  - Should have a keen interest on building beautiful yet simple UI

Appendix

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]

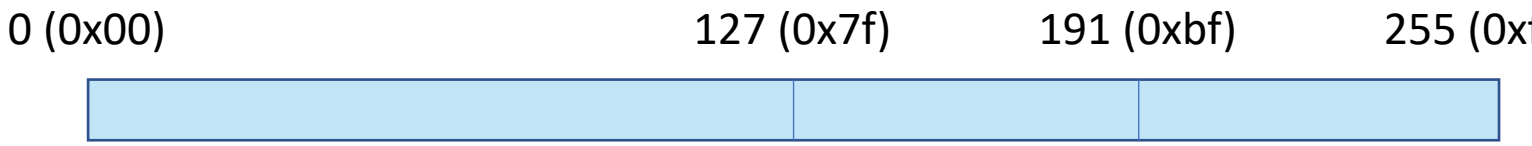
https://github.com/ethereum/wiki/wiki/RLP

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



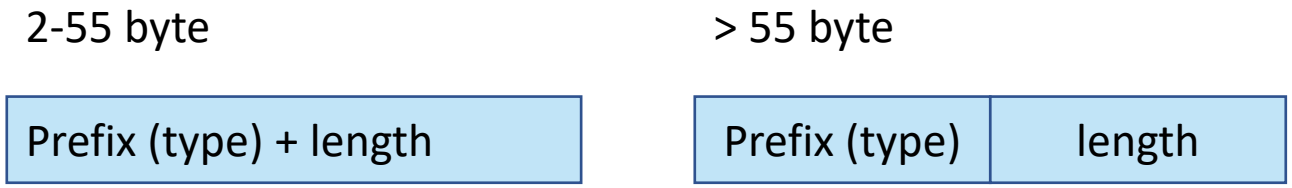
- [0x00, 0x7f]: byte
- [0x80, 0xbf]: string
- [0xc0, 0xff]: list

- 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]

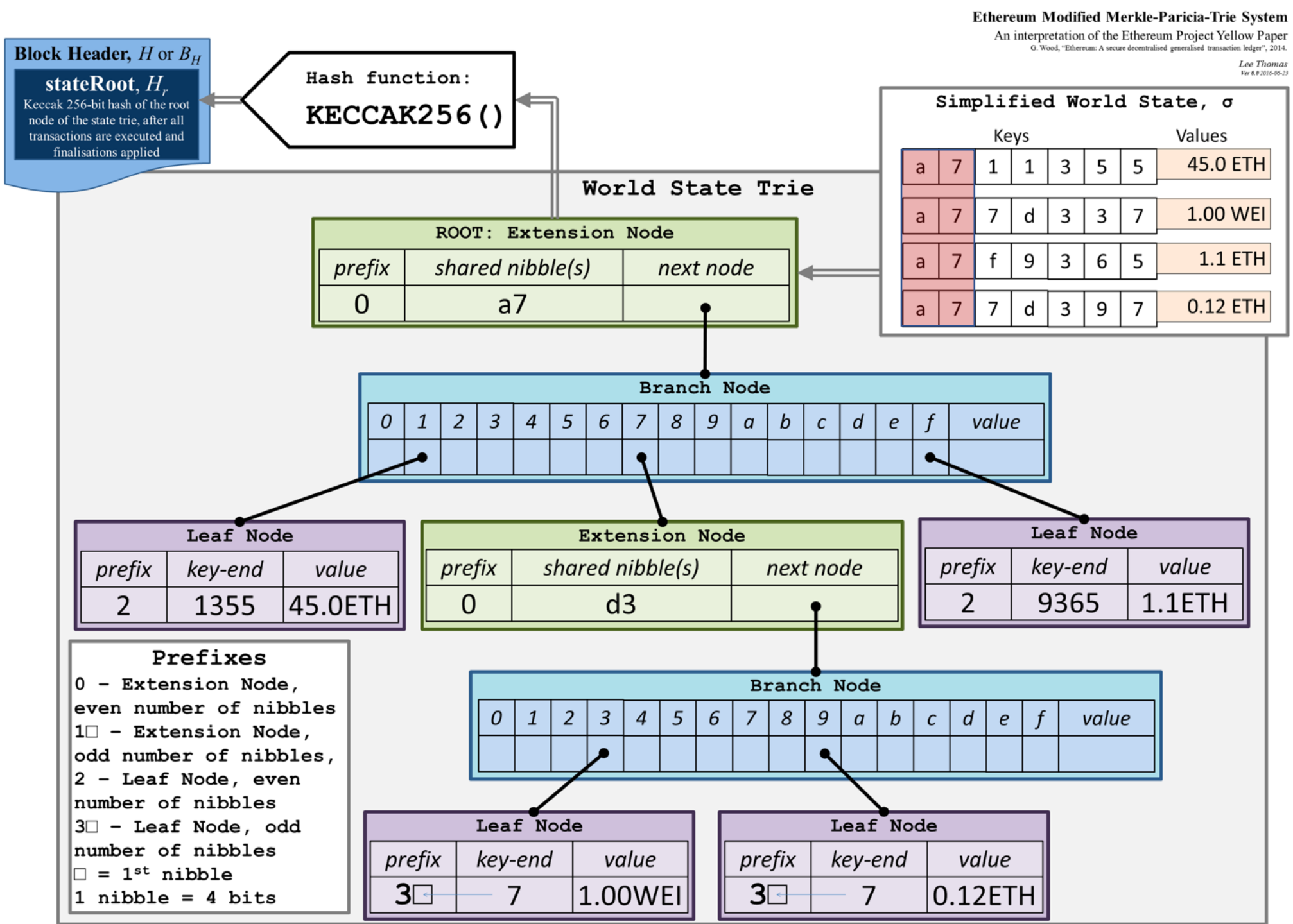
https://github.com/ethereum/wiki/wiki/RLP

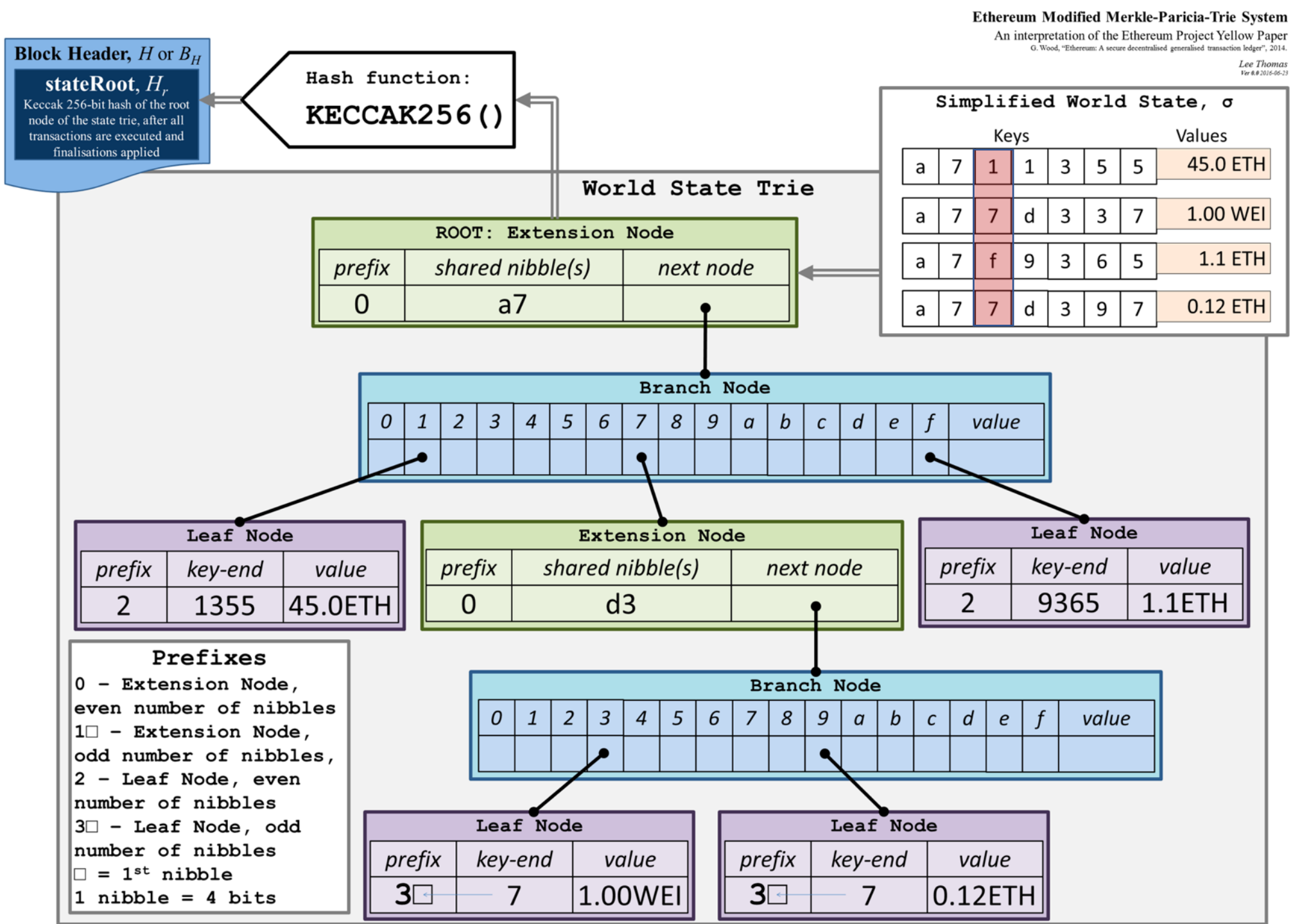
RLP

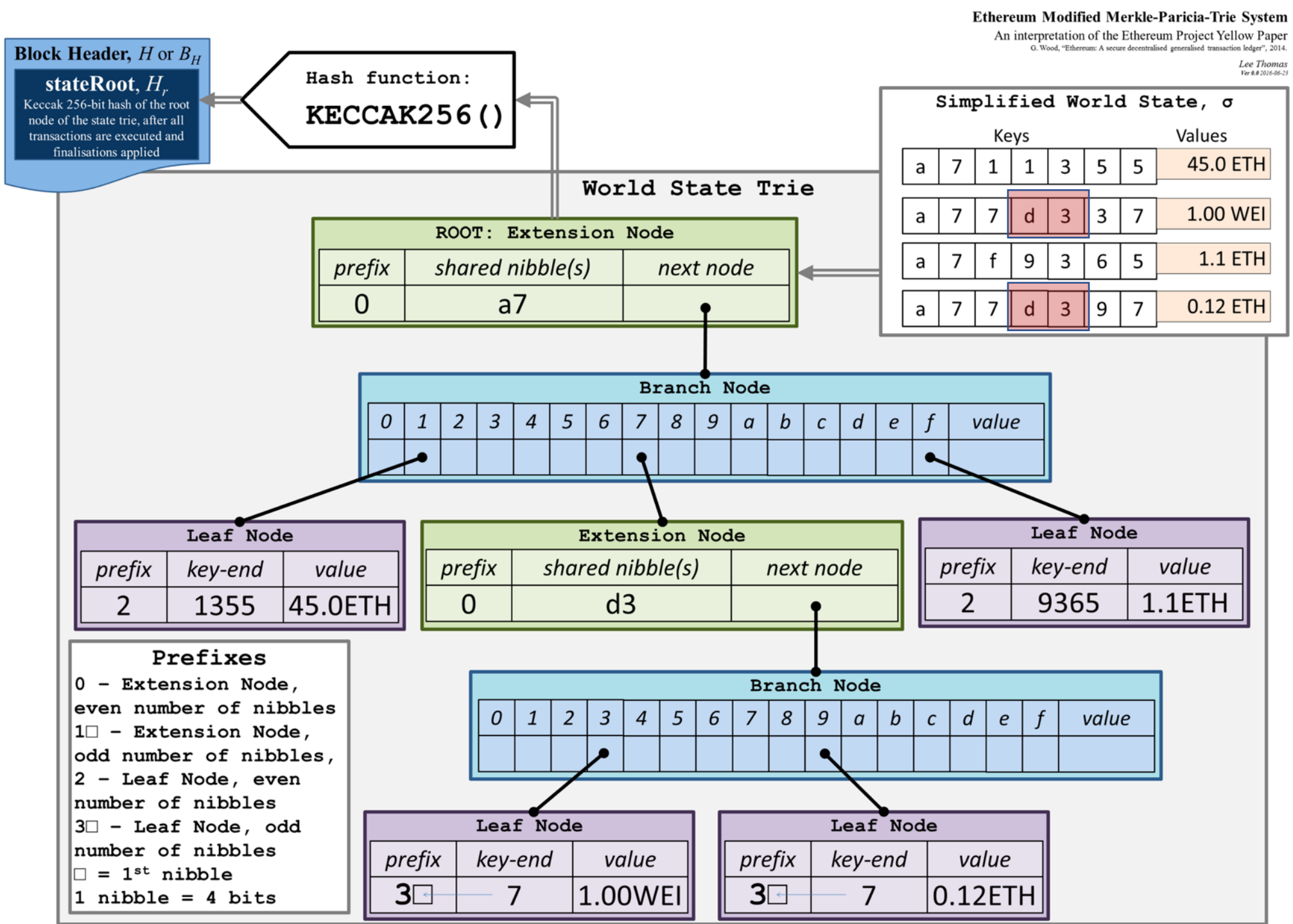
- [0x00, 0x7f]: byte
- [0x80, 0xbf]: string
- [0xc0, 0xff]: list

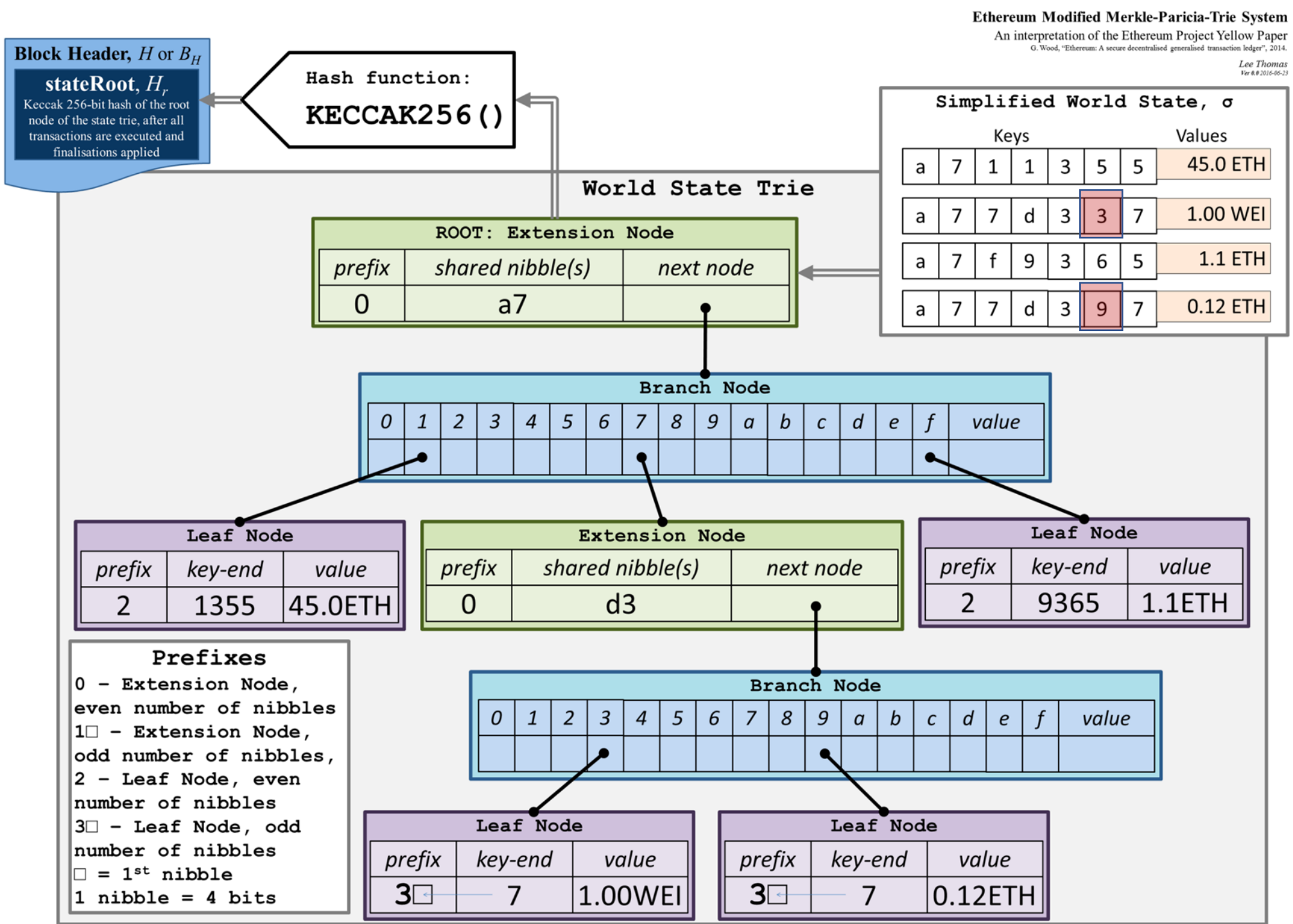


- The string “taiwan” 0x86 = 0x80 + 0x06
  - [0x86, 0x74, 0x61, 0x69, 0x77, 0x61, 0x6e]
- The list ['nccu', 'taiwan’] 0xcc = 0xc0 + 0x0c
  - [0xcc, 0x84, 0x6e, 0x63, 0x63, 0x75, 0x86, 0x74, 0x61, 0x69, 0x77, 0x61, 0x6e]
- The string with 256 “a”, “aaaaaaaa….aaaaa”
  - [0xb9, 0x01, 0x00, 0x61, …, 0x61]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