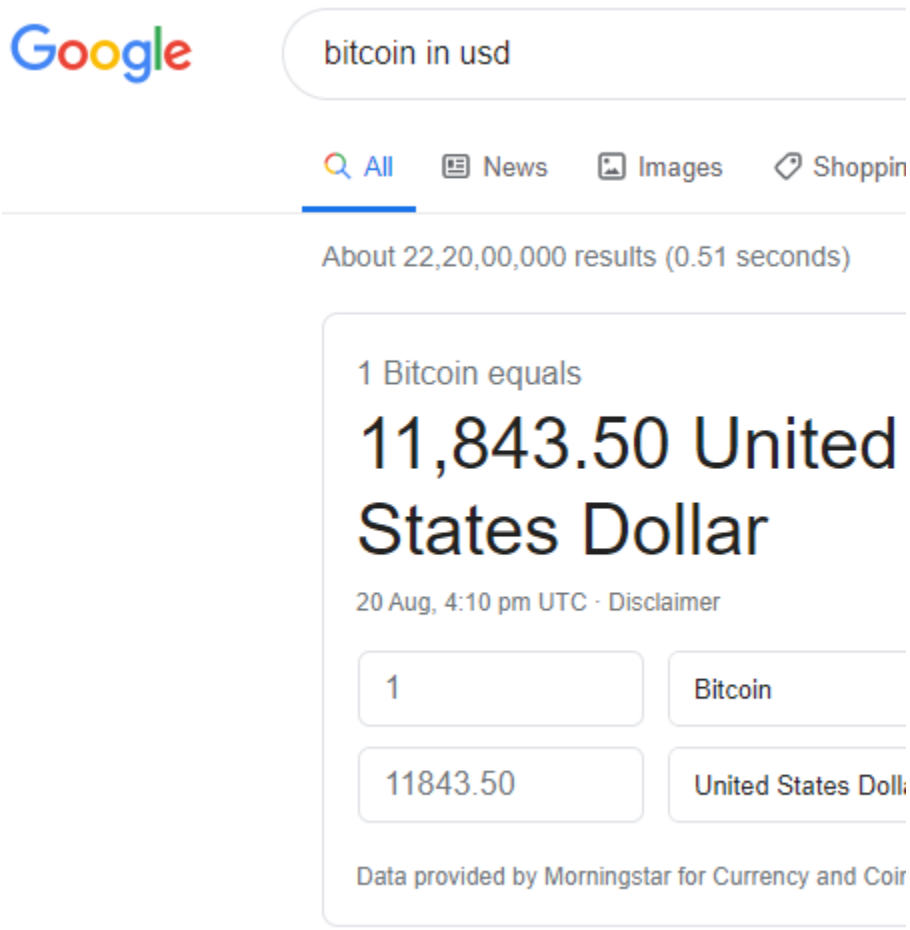
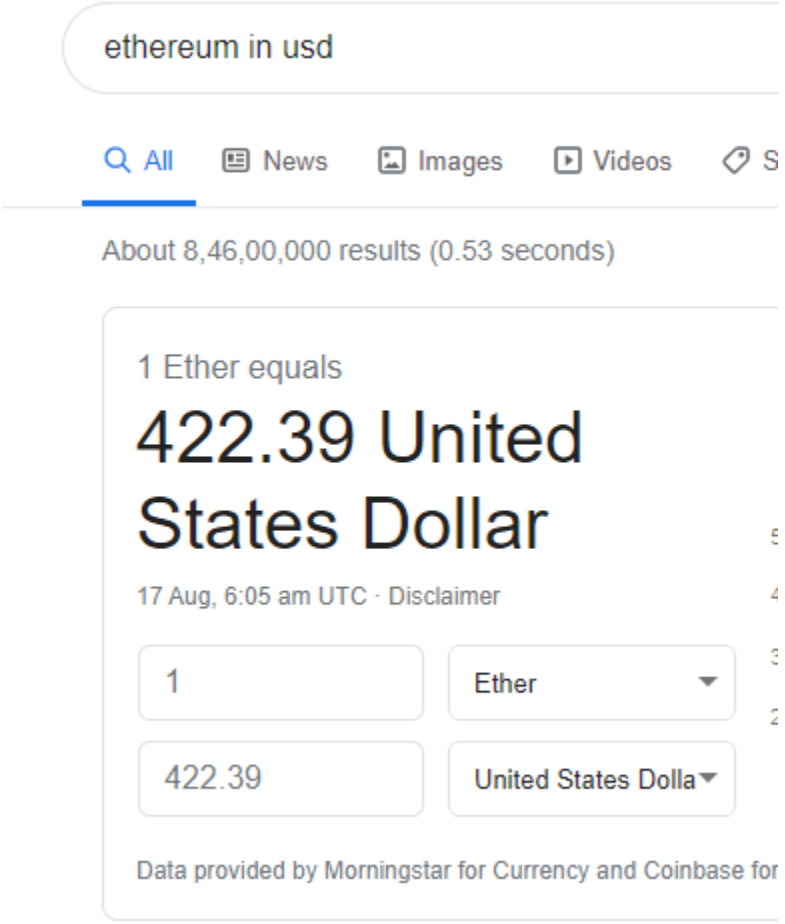
Ethereum: A blockchain-based smart contract platform

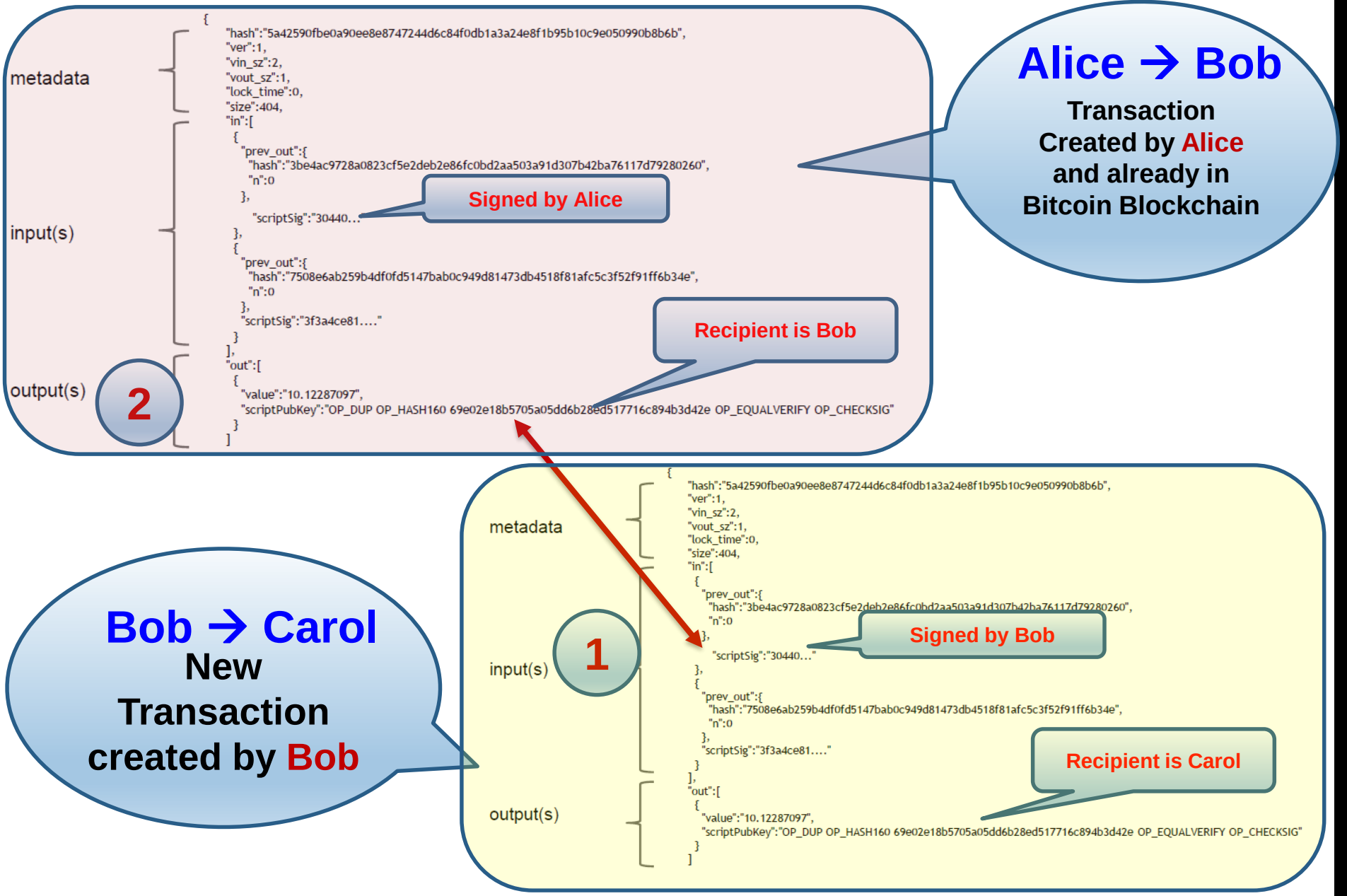
Dr. Raju Halder,
Dept. of Comp. Sc. & Engg., IIT Patna
halder@iitp.ac.in

Ethereum: A blockchain-based smart contract platform





Bitcoin Vs. Ethereum




```
{
"hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
"ver":1,
"vin_sz":2,
"vout_sz":1,
"lock_time":0,
"size":404,
"in":[
 {
  "prev_out":{
   "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
   "n":0
  },
   "scriptSig":"30440…"
 },
 {
  "prev_out":{
   "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
   "n":0
  },
  "scriptSig":"3f3a4ce81…."
 }
],
"out":[
 {
  "value":"10.12287097",
  "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
 }
]
```

Bitcoin Vs. Ethereum

- Existing blockchain protocols were designed with script language

Less Expressive
Not Turing Complete

Bitcoin Vs. Ethereum

- Why not make a protocols like this?

OR
THIS

OR
THIS

Bitcoin Vs. Ethereum

Vitalik Buterin

Ethereum

- It's *the world's programmable blockchain.*
- Blockchain with expressive programming language
  - Programming language makes it ideal for **smart contracts**
  - Smart contracts enable much more applications

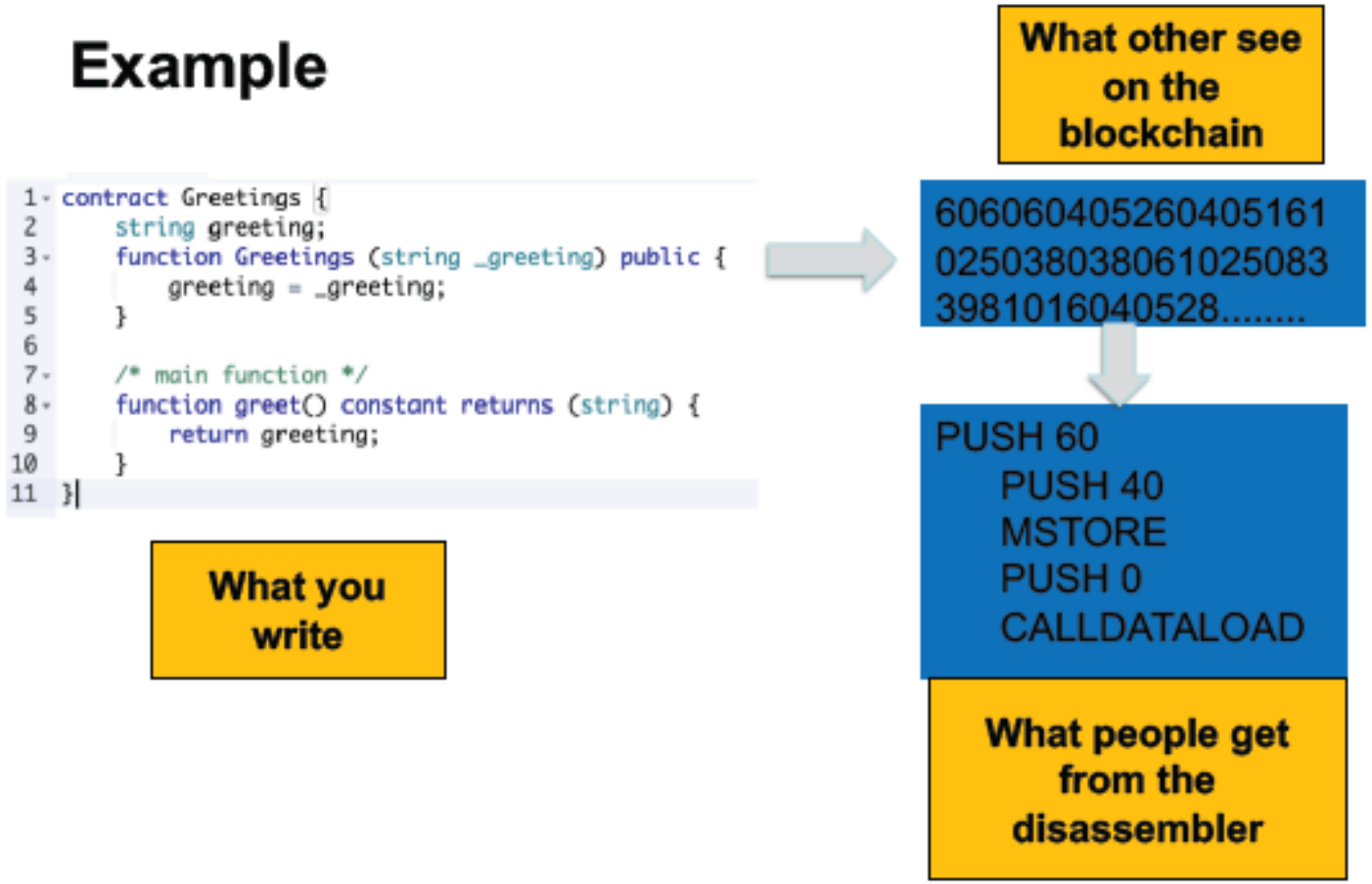
Turing Complete Language
(e.g., Solidity Language)

Bitcoin Vs. Ethereum

A **smart contract** is a computer program executed in a **secure environment** that directly controls **digital assets**

Blockchain is widely used for other applications

What you can see?



Ethereum World State

- A blockchain is a globally shared, transactional database.

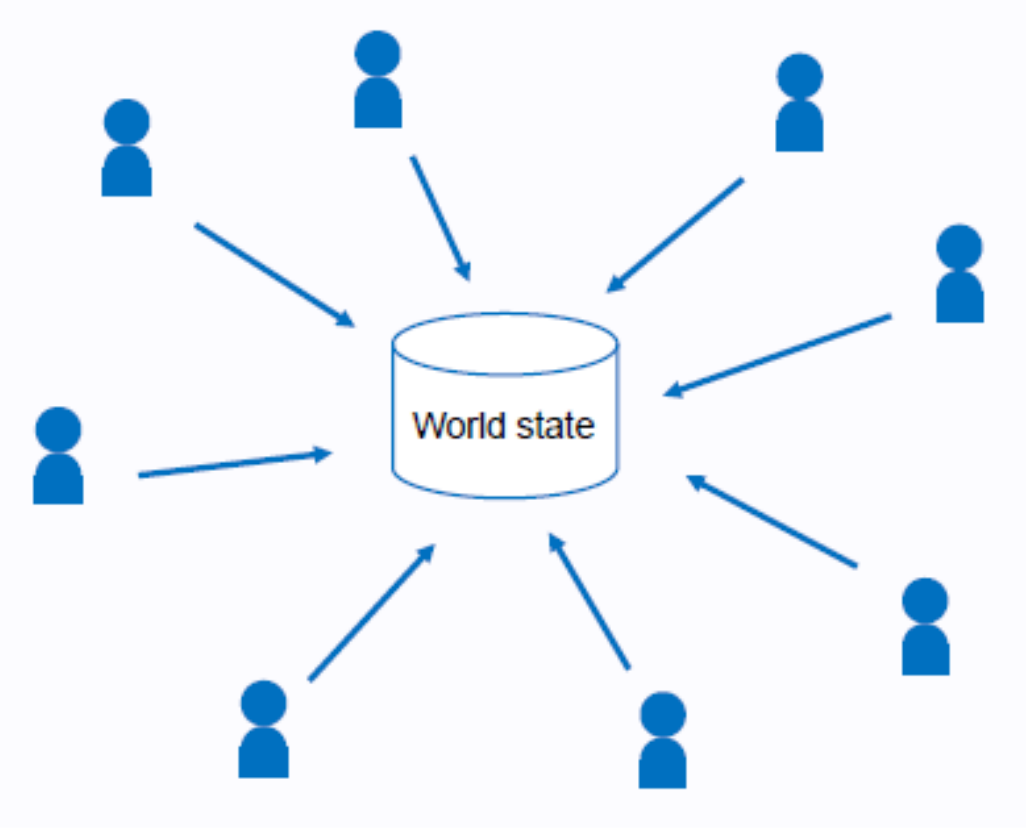


Image credit: https://github.com/takenobu-hs/ethereum-evm-illustrated

Ethereum World State

- A blockchain is a globally shared, **decentralised**, transactional database.

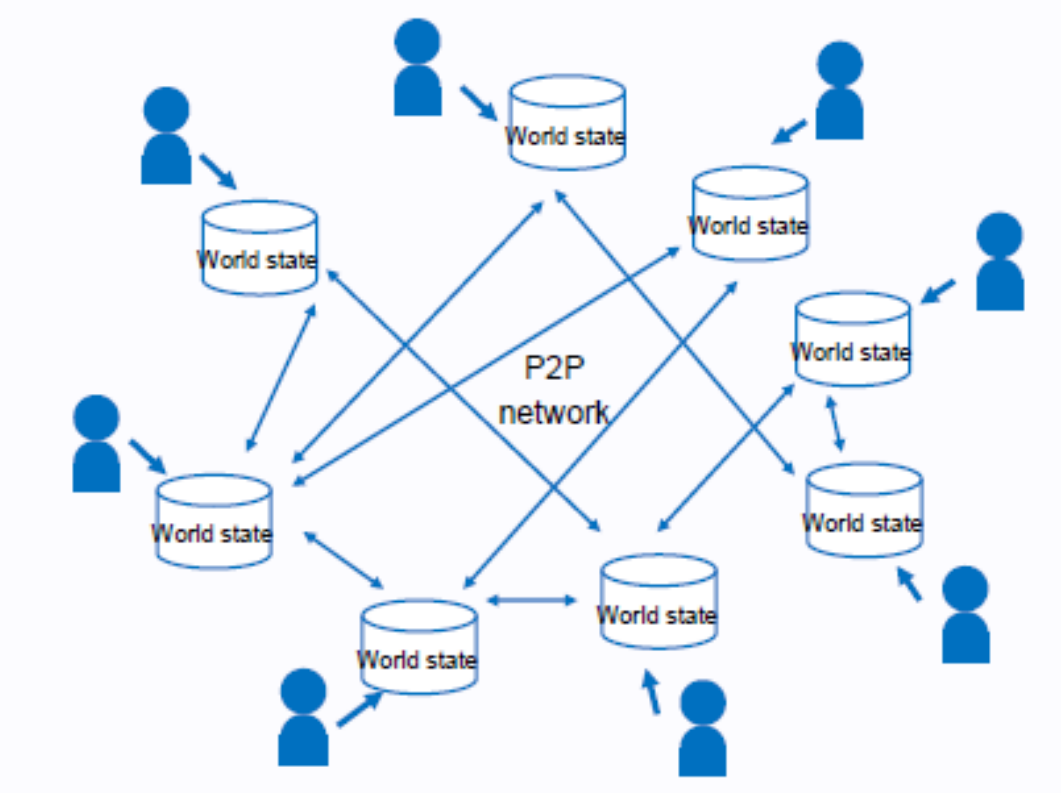
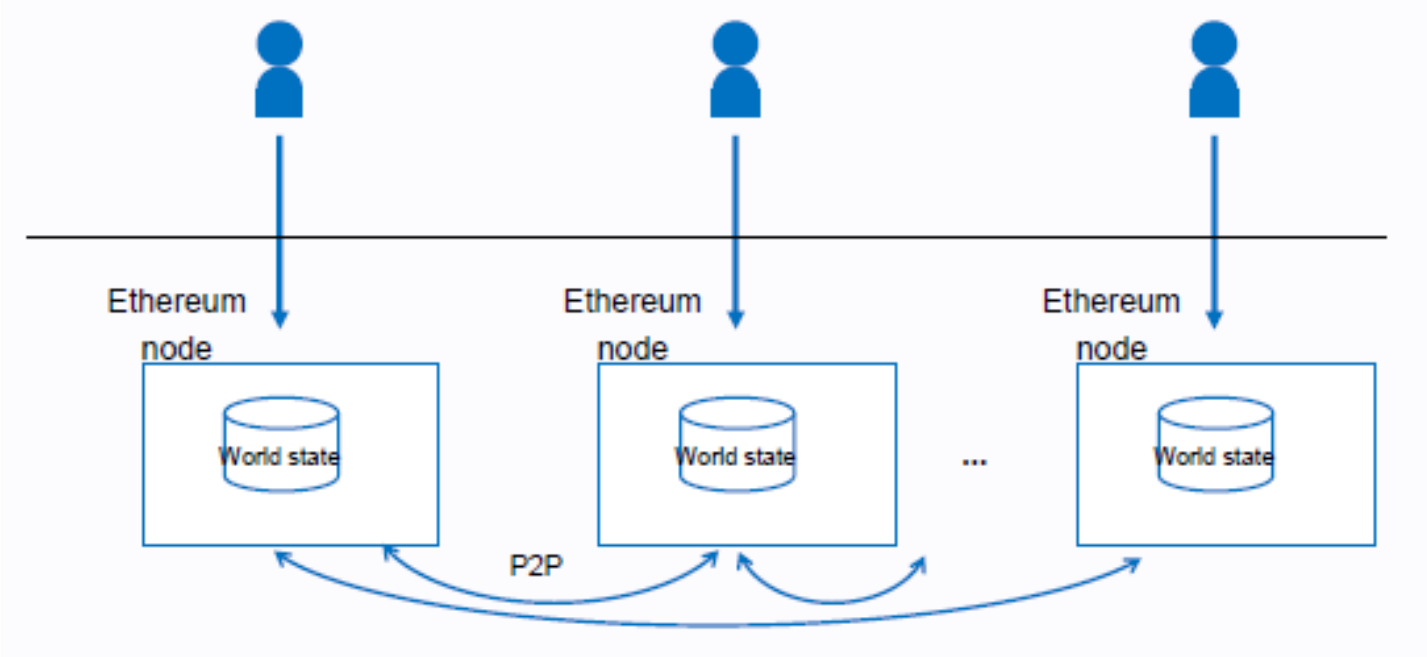


Image credit: https://github.com/takenobu-hs/ethereum-evm-illustrated

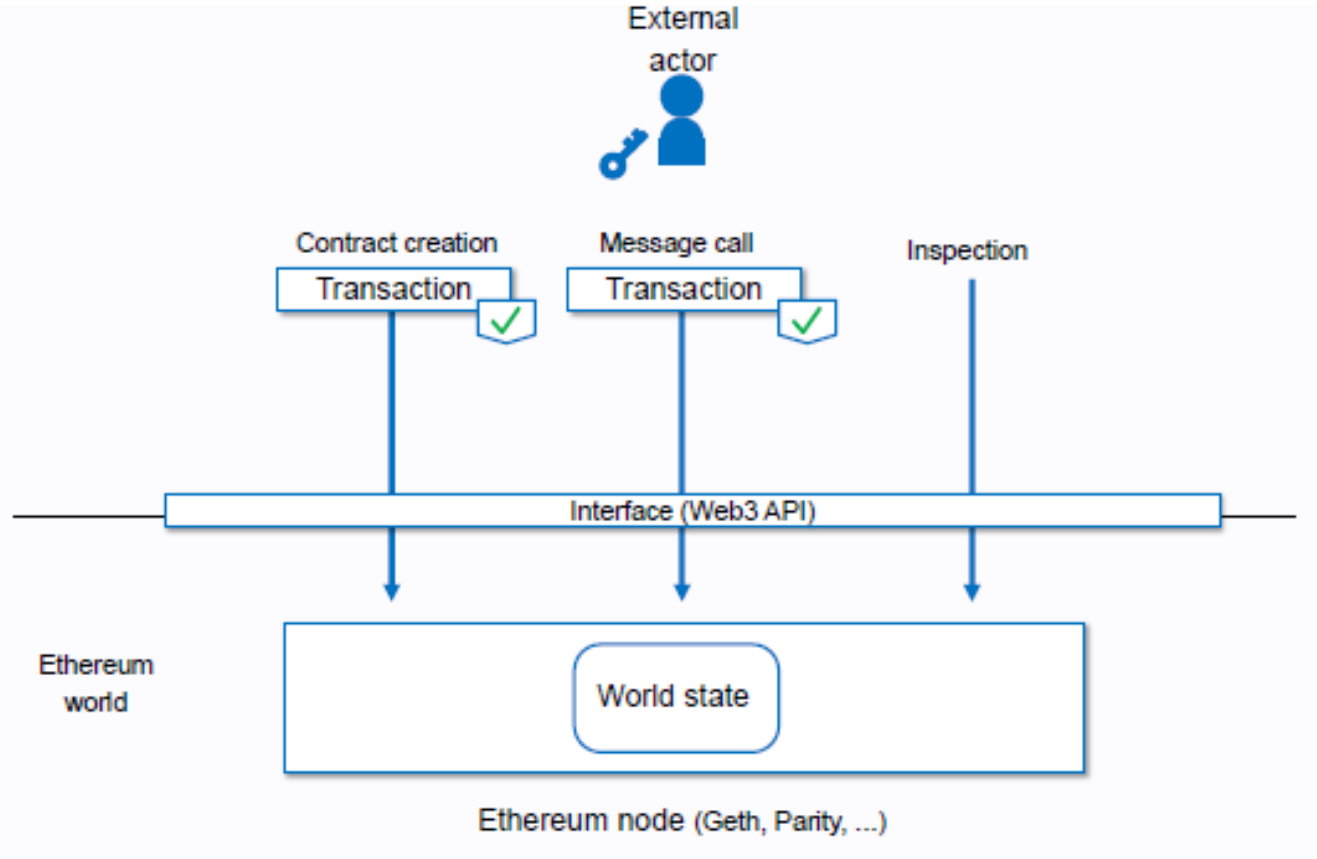
Ethereum World State

- Decentralised nodes constitute Ethereum P2P network.



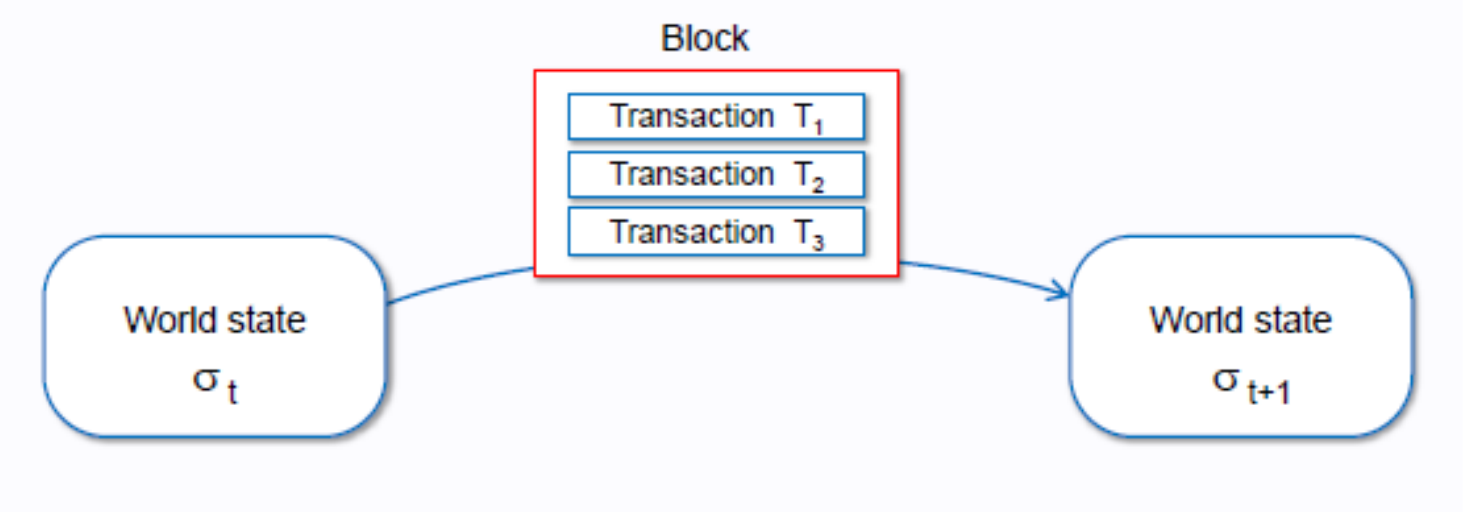
Interface to Nodes

- External actors access the Ethereum world through Ethereum nodes.



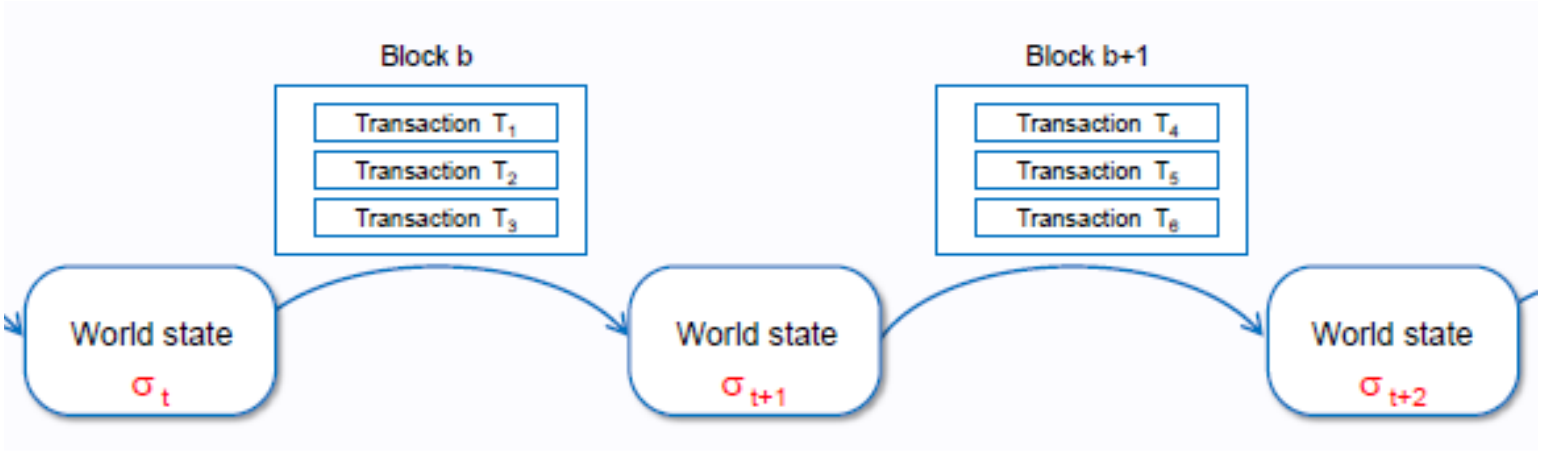
Ethereum as State Machine

- Ethereum can be viewed as a transaction-based state machine.
- Transactions are collated into blocks. A block is a package of data.

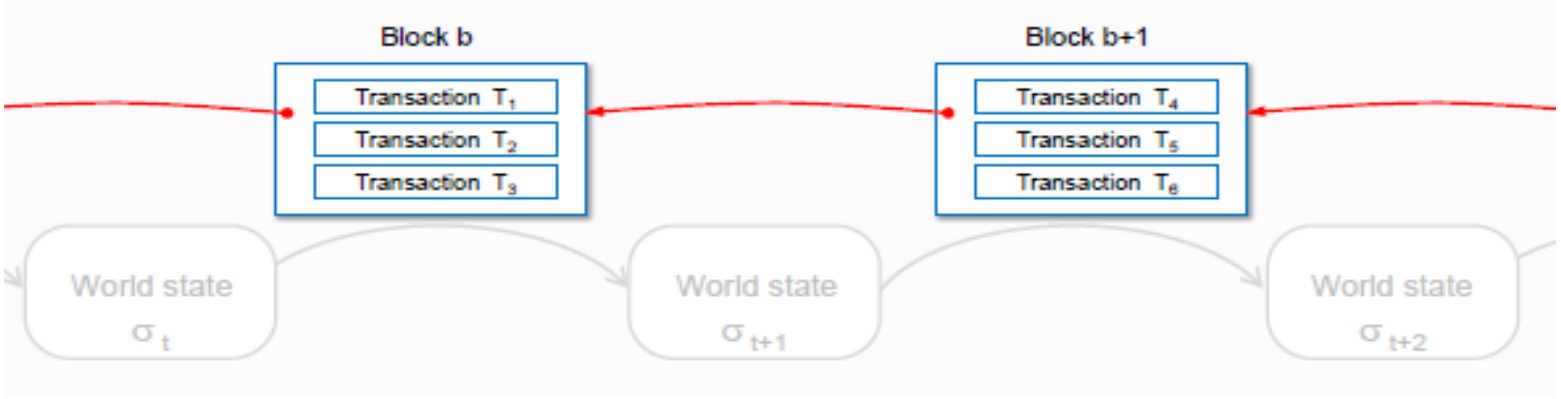


Statechain Vs. Blockchain

- From the viewpoint of the states, Ethereum can be seen as a statechain.

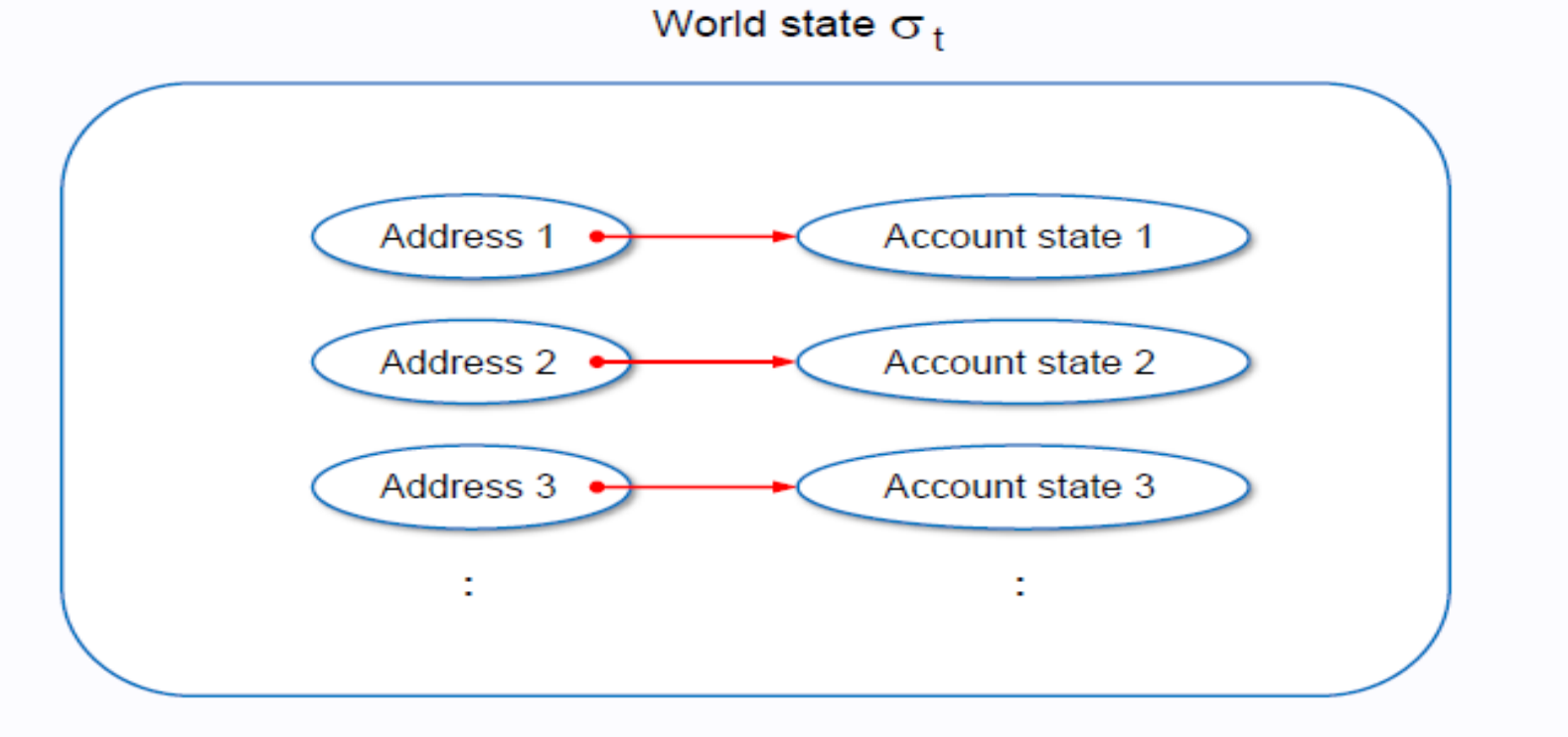


- From the viewpoint of the implementation, Ethereum can also be seen as a chain of blocks, so it is `BLOCKCHAIN`.

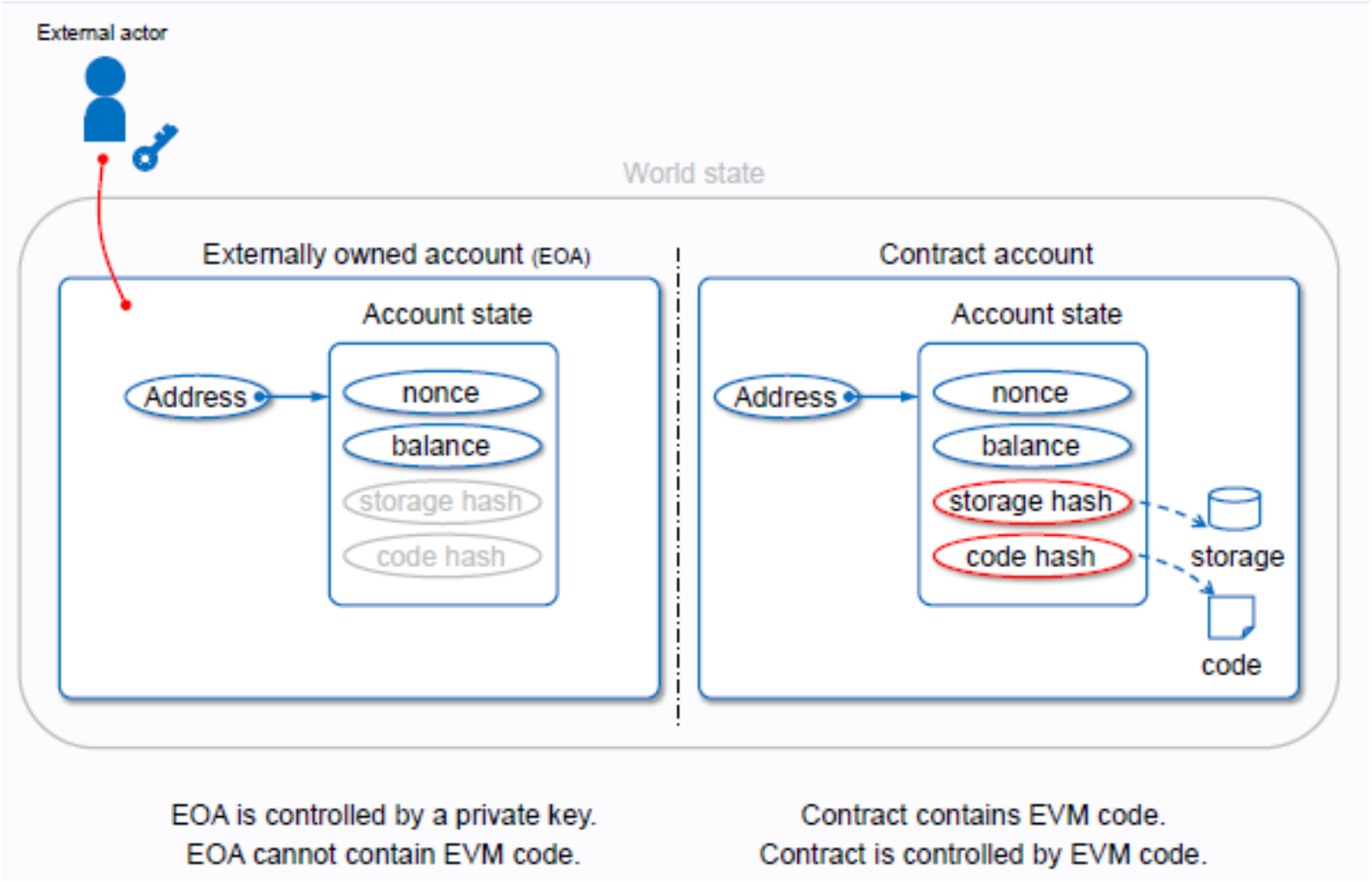


World State

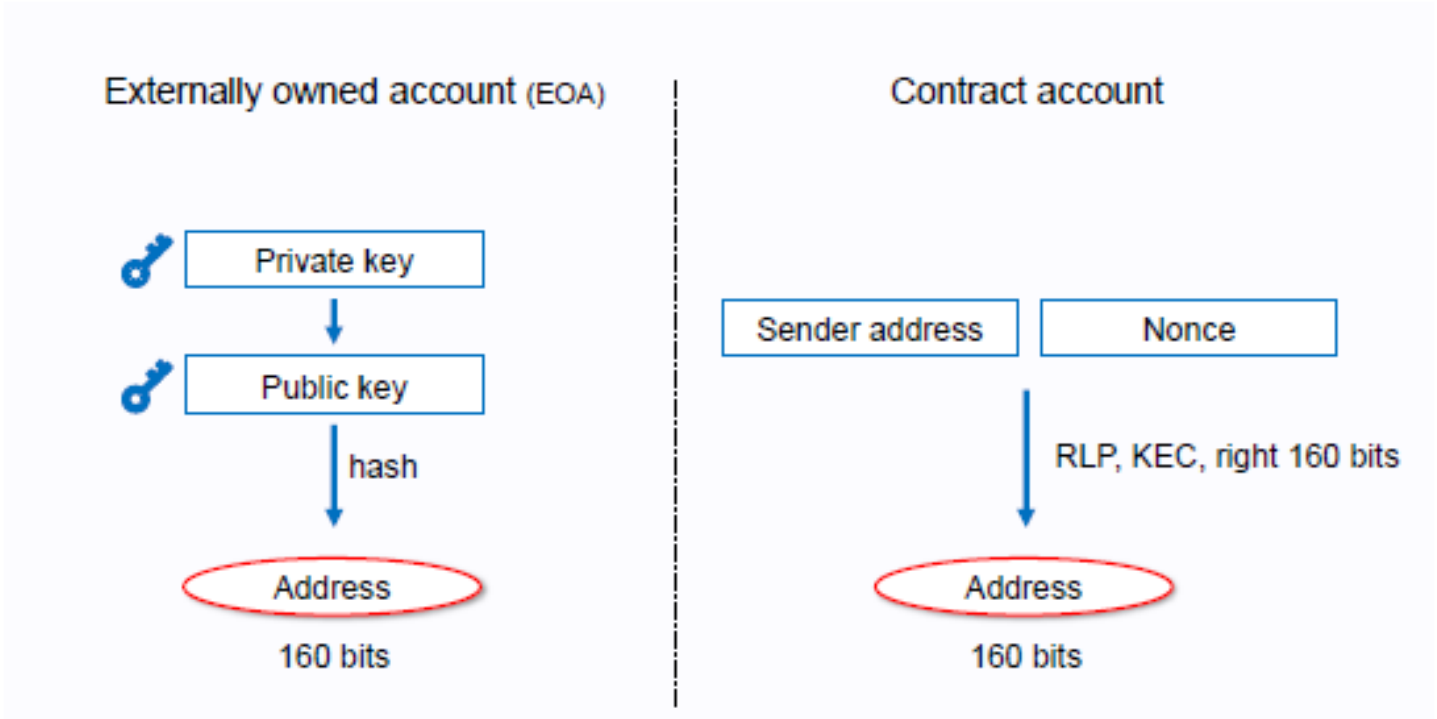
- The world state is a mapping between address and account state.



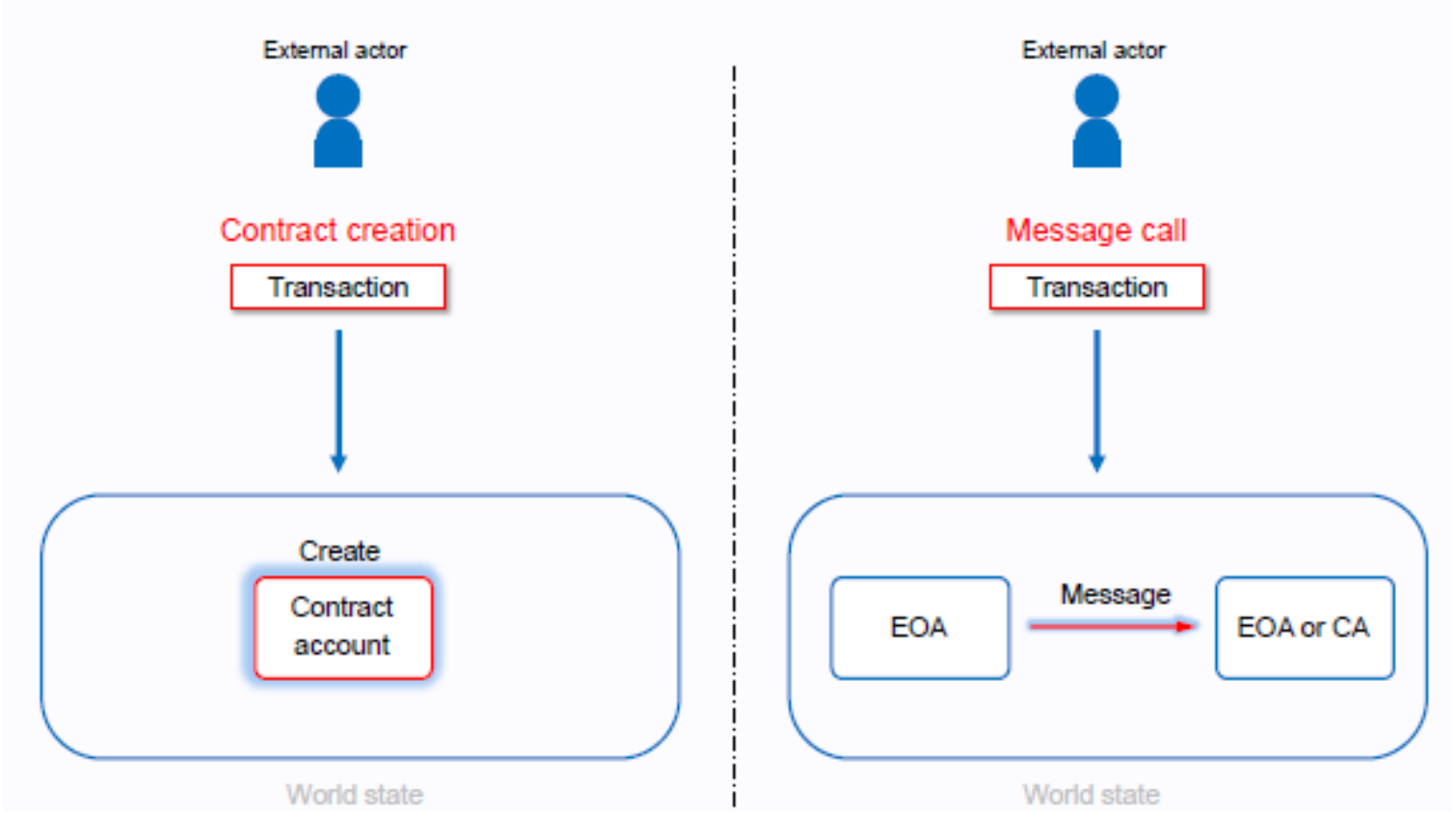
Two Types of Accounts



Account Address

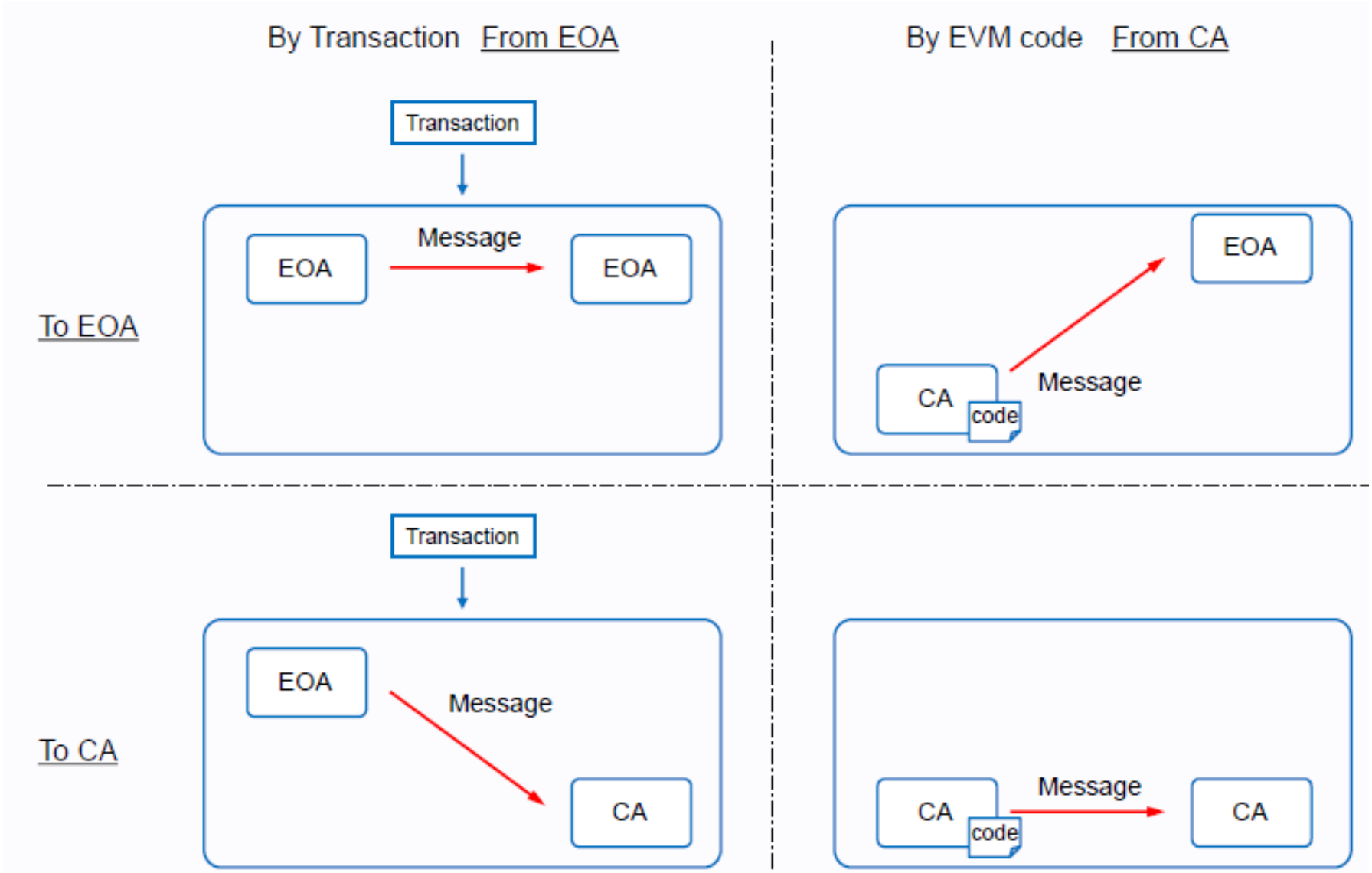


Two Types of Transactions

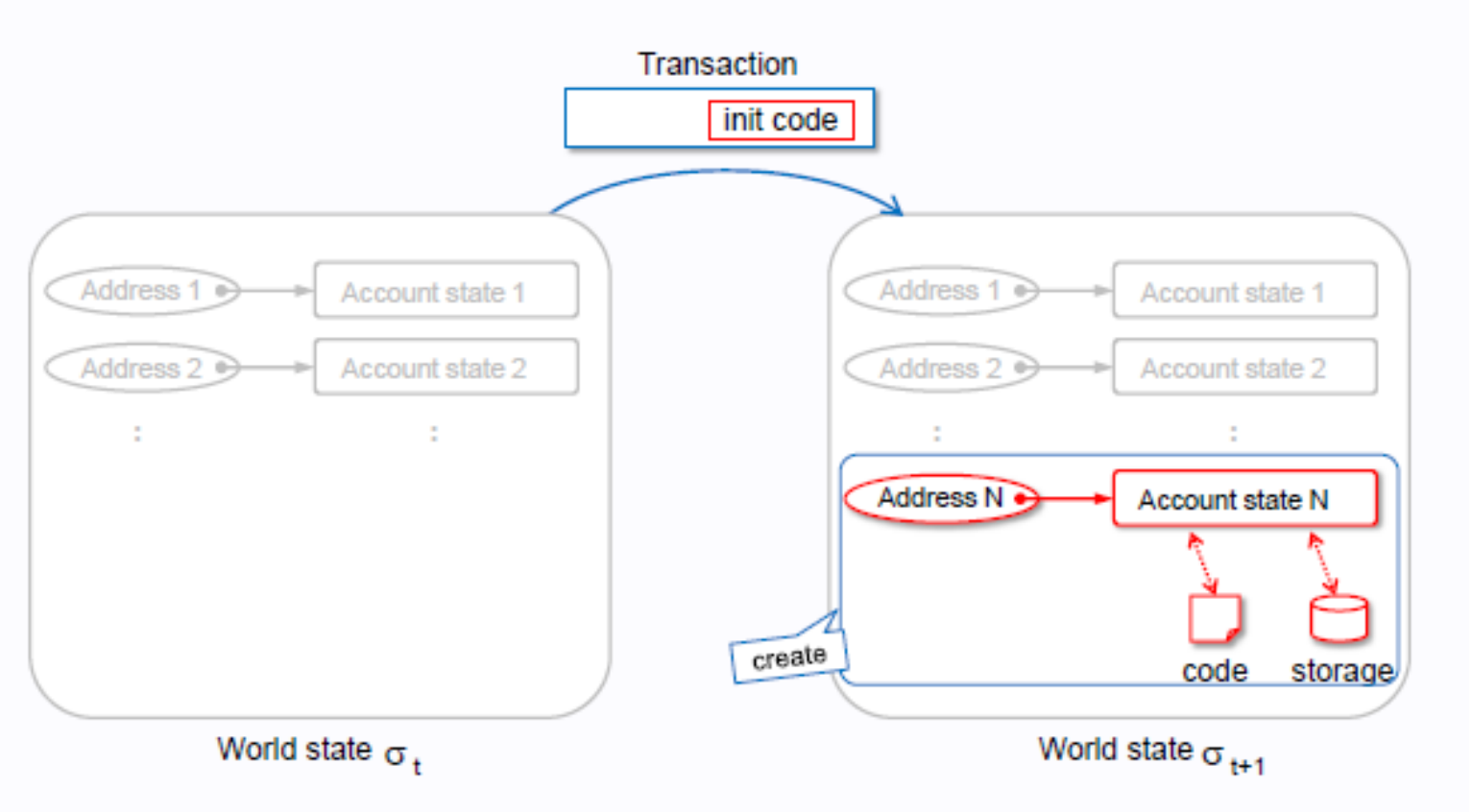


Message is Data (as a set of bytes) and Value (specified as Ether) .

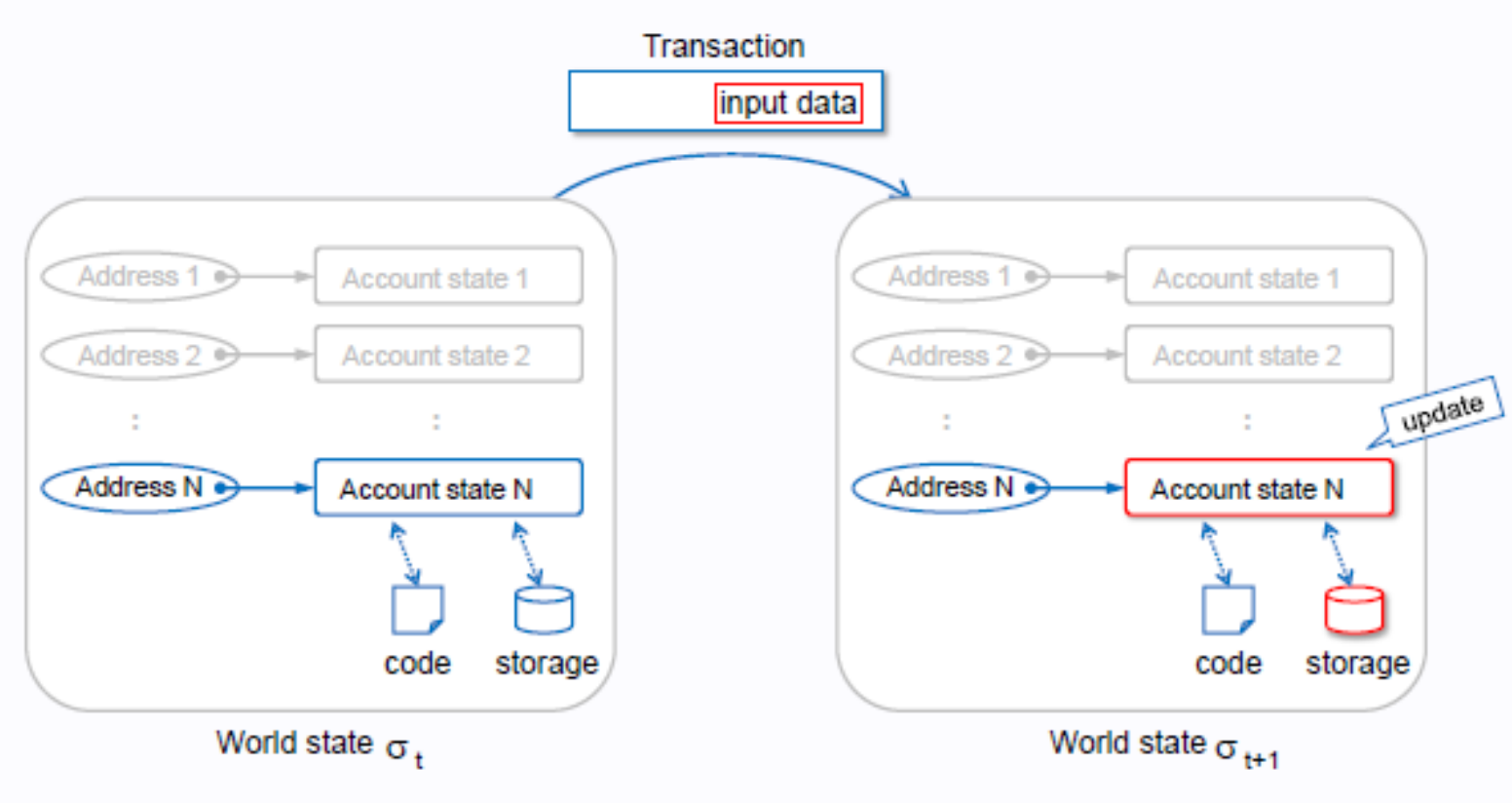
Four Cases of Messages



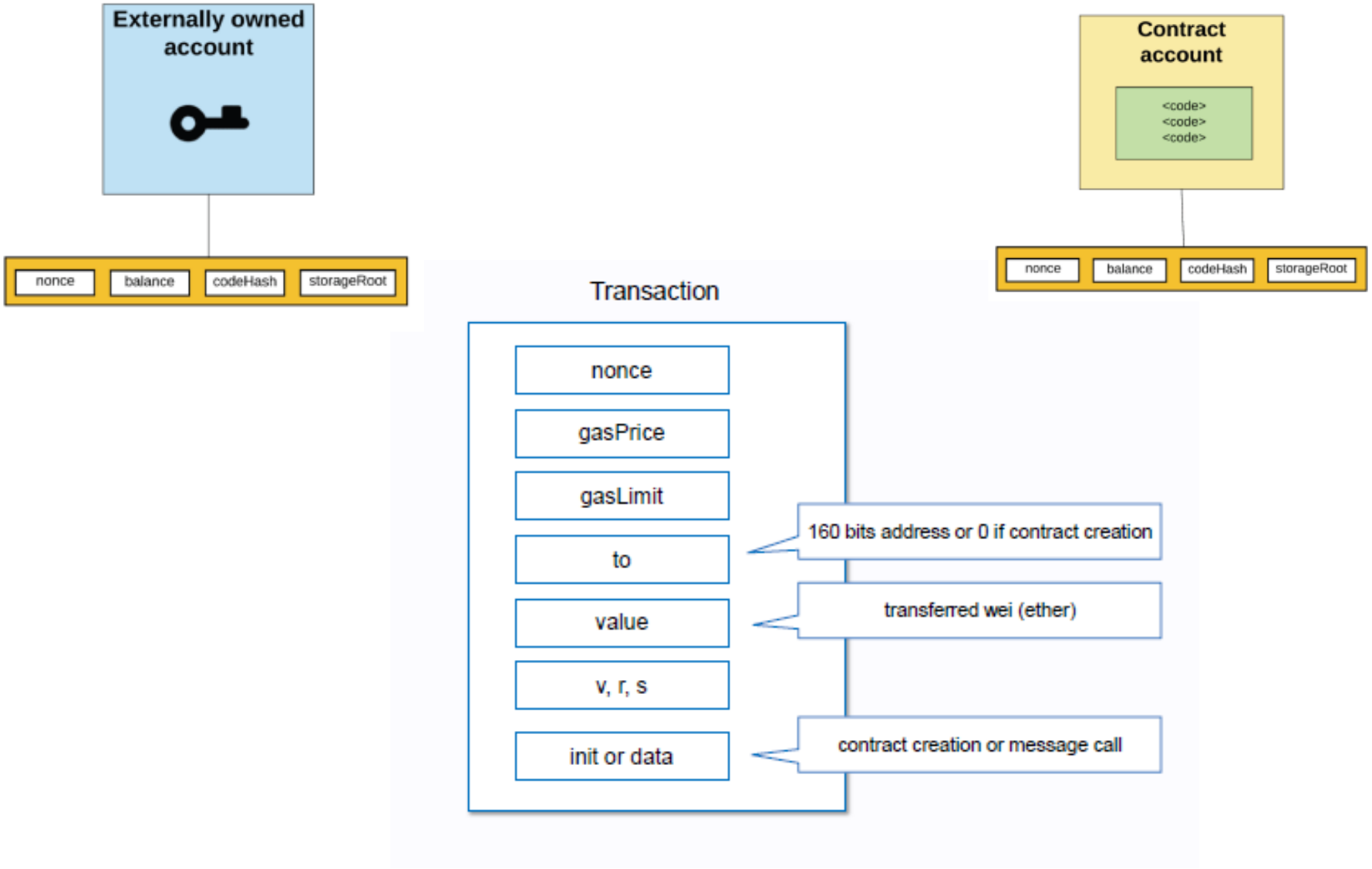
Contract Creation Transaction



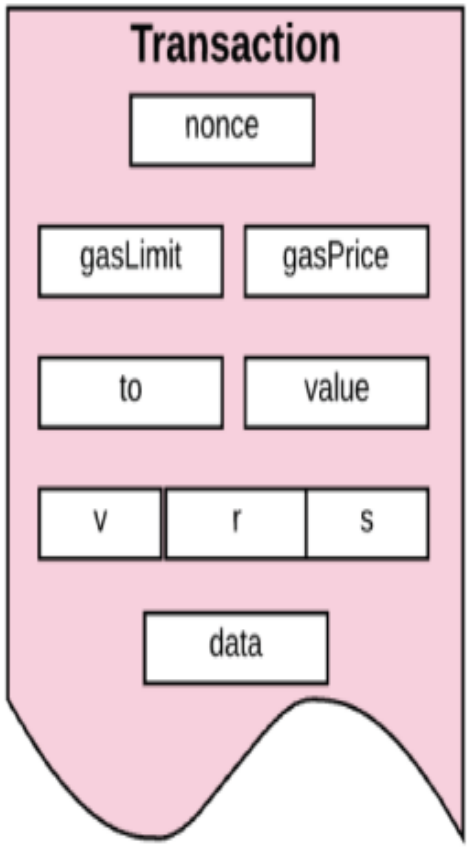
Message Call Transaction



Transaction Structure



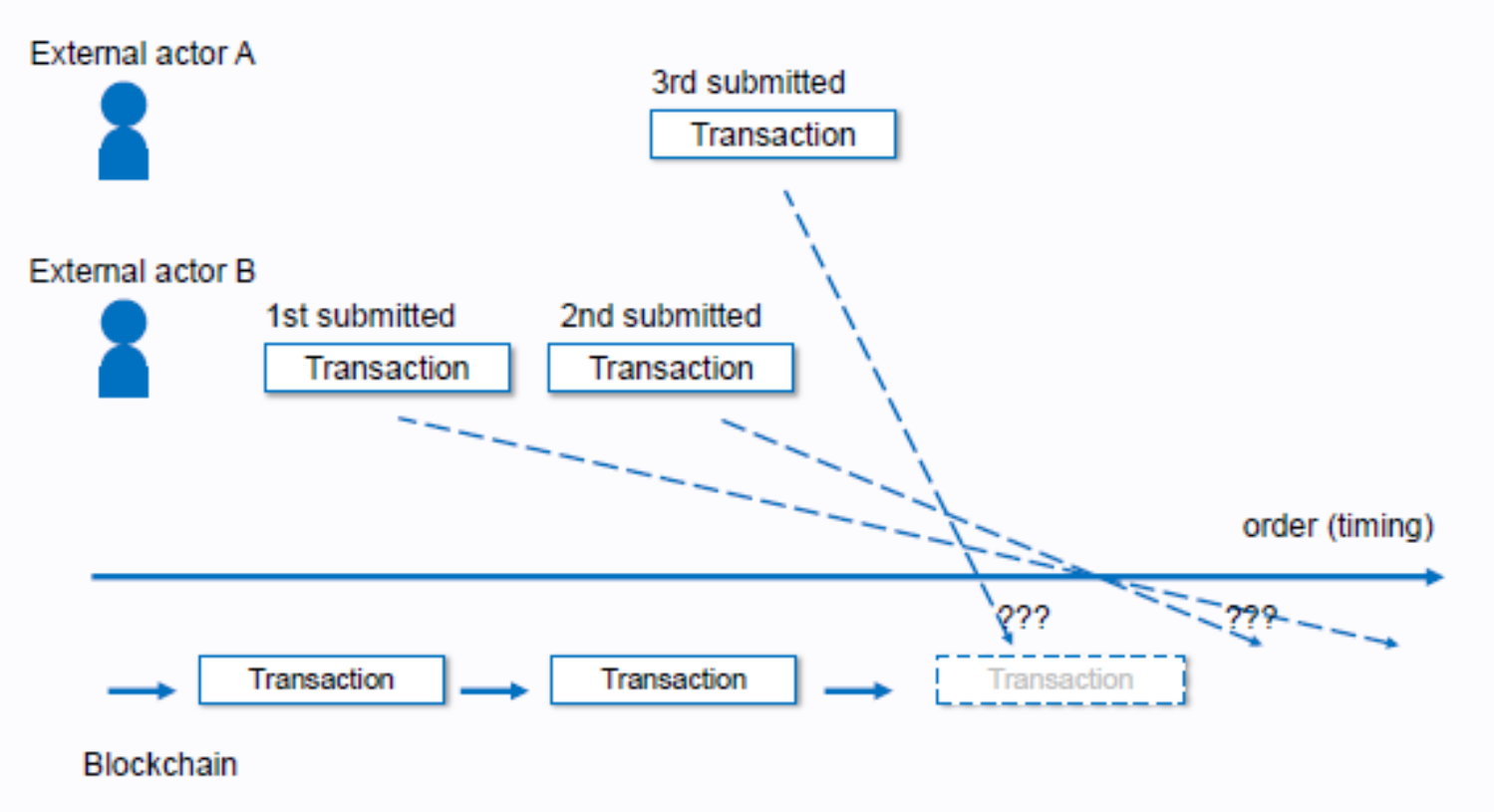
Transaction Structure



- nonce: A count of the number of transactions sent by the sender.
- gasPrice
- gasLimit
- to: Recepient's address
- value: Amount of Wei Transferred from sender to recipient.
- v,r,s: Used to generate the signature that identifies the sender of the transaction.
- init: EVM code used to initialize the new contract account.
- data: Optional field that only exists for message calls.

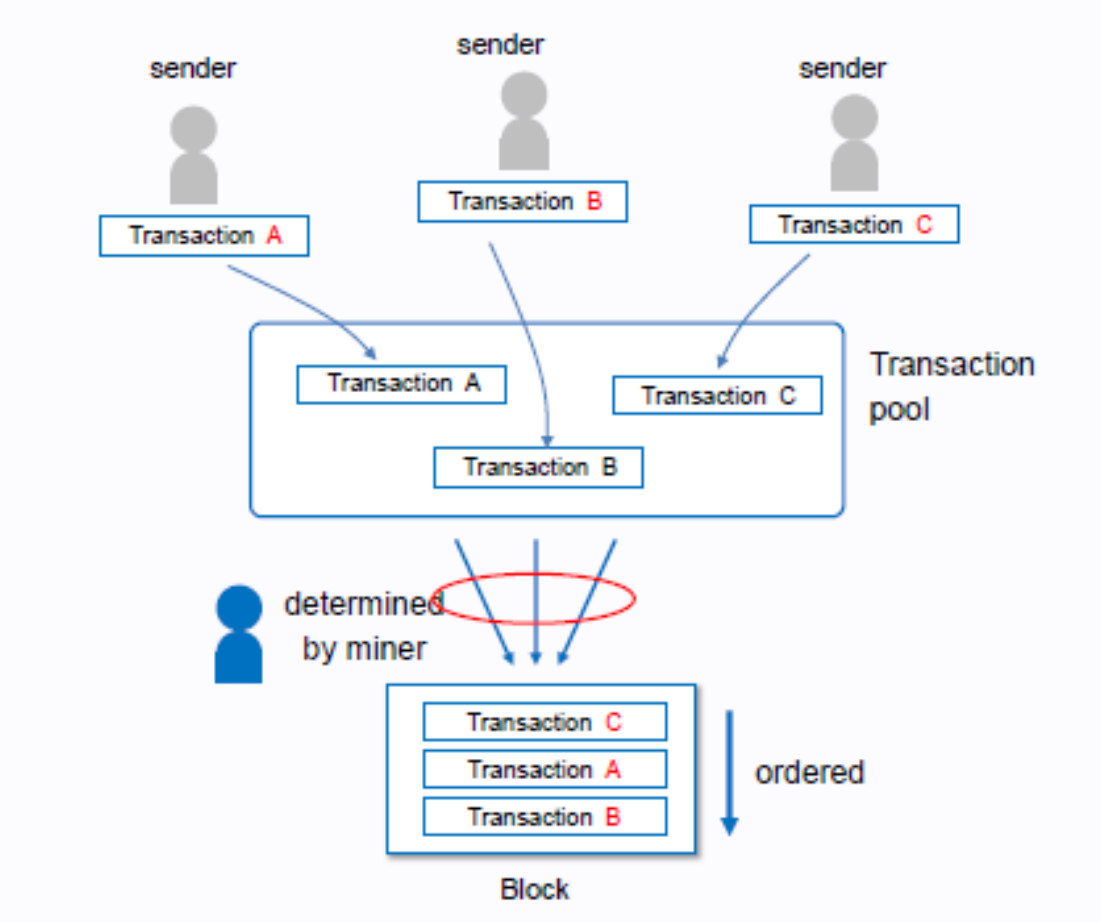
Order of Transactions

- Transaction order is not guaranteed



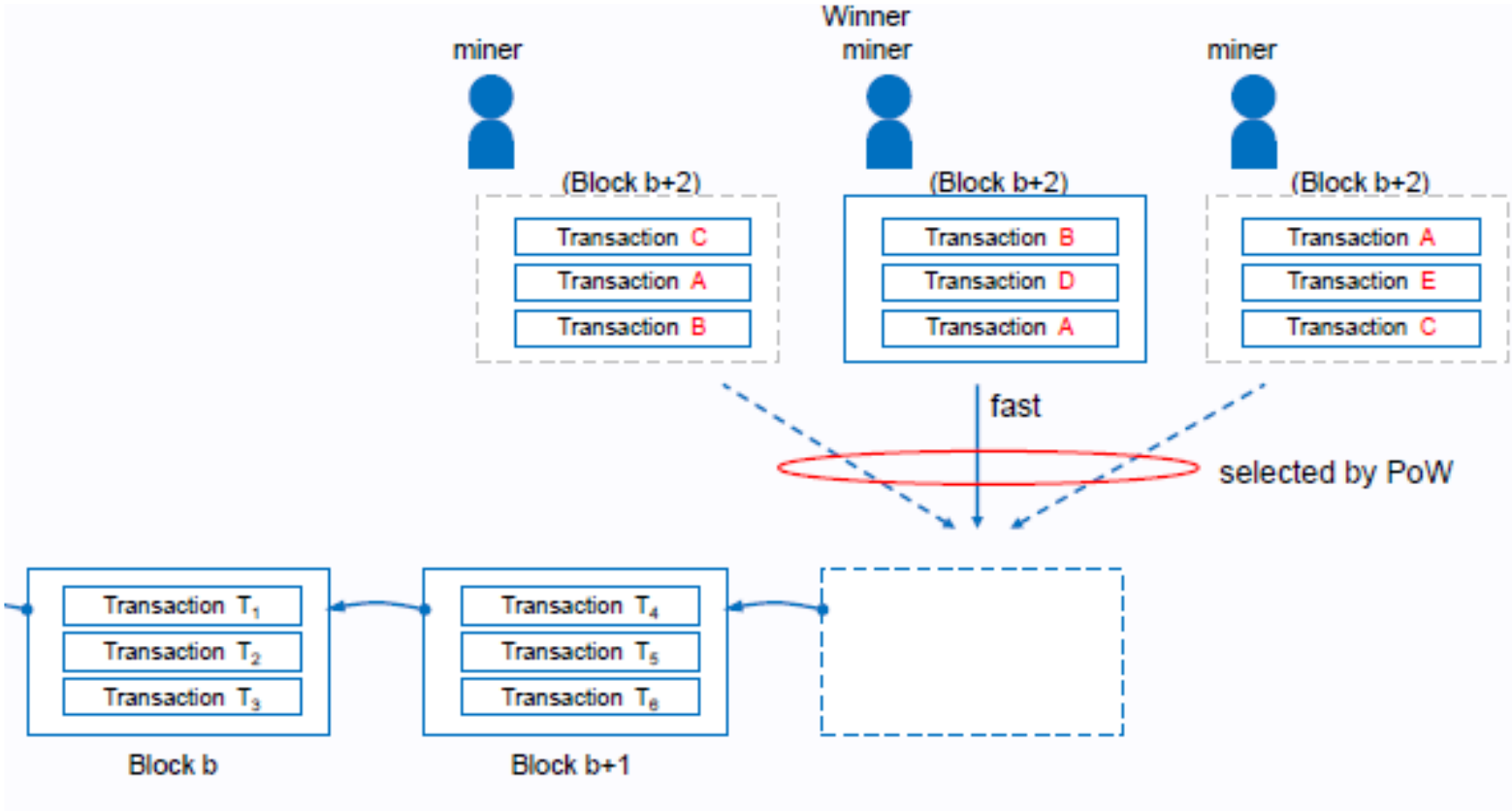
Order of Transactions

- Miner can determine the order of transactions in a block.

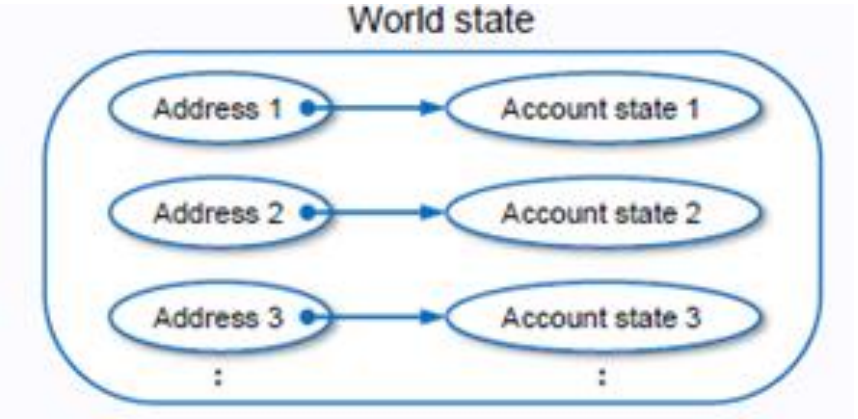


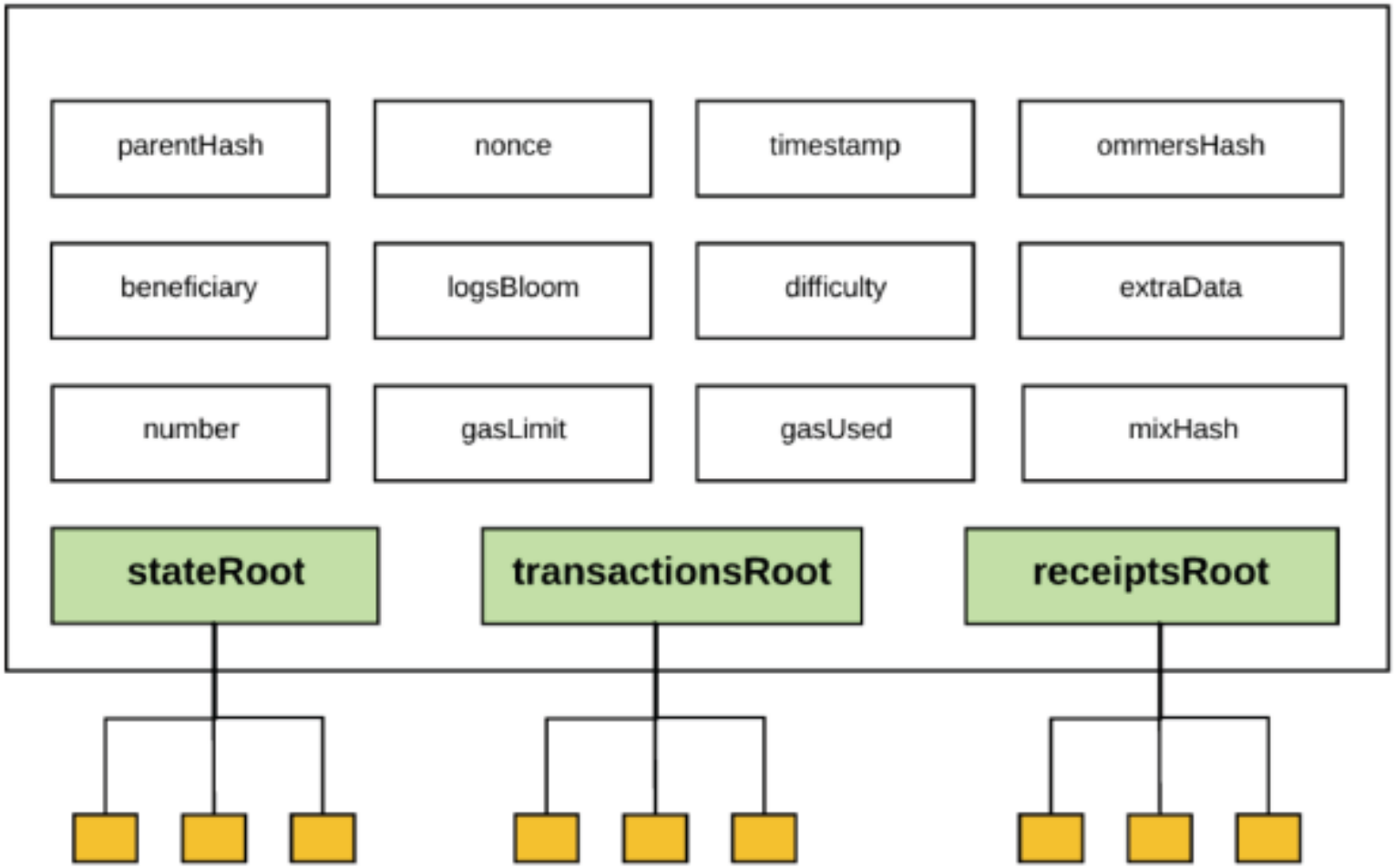
Order of Transactions

- The order between blocks is determined by a consensus algorithm such as PoW.



Block Header





Merkle Patricia Trie

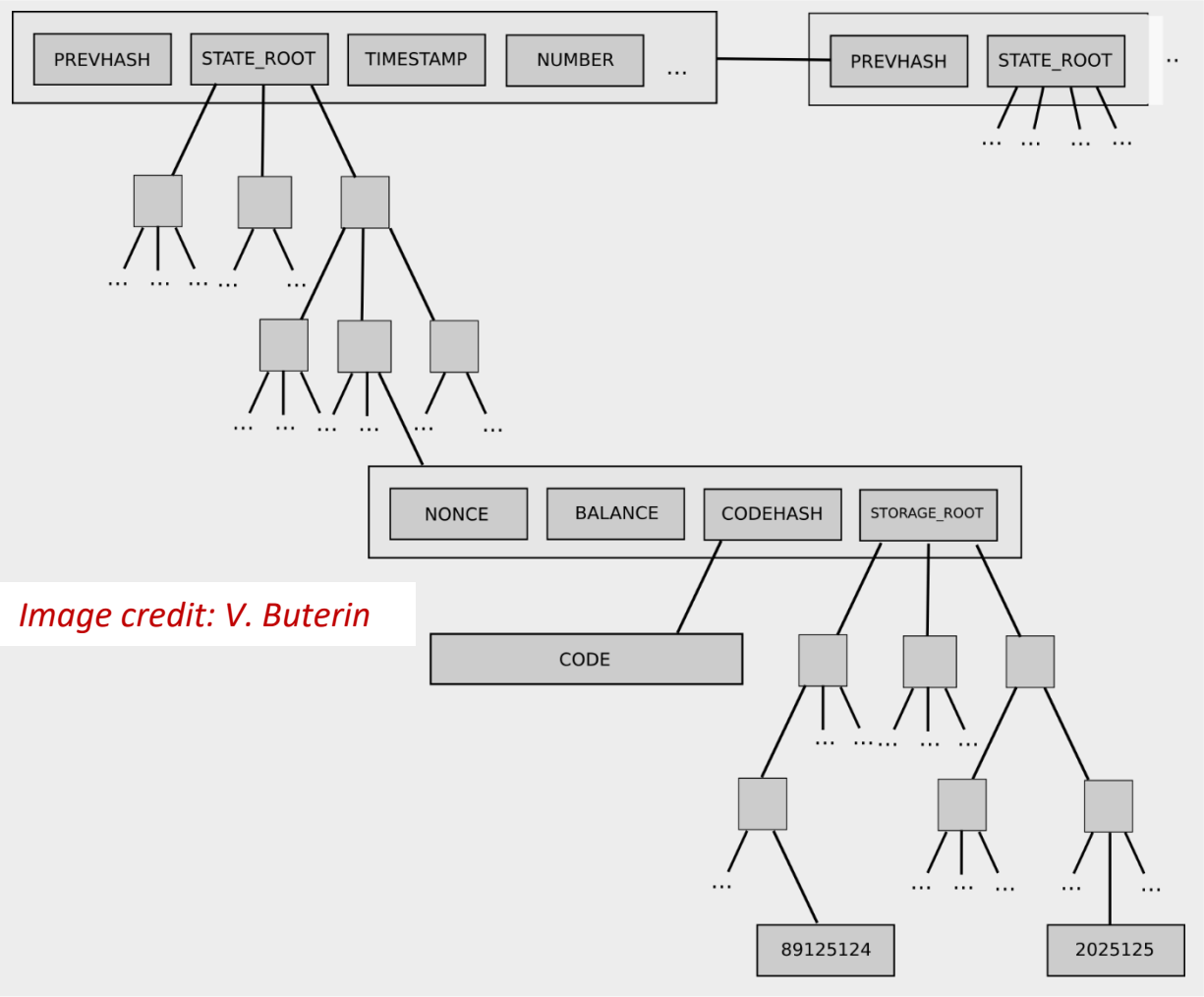
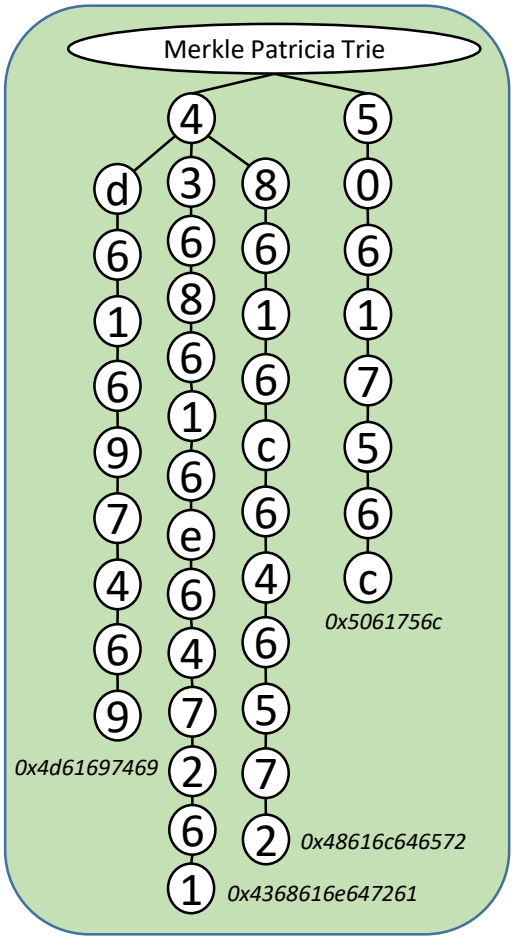
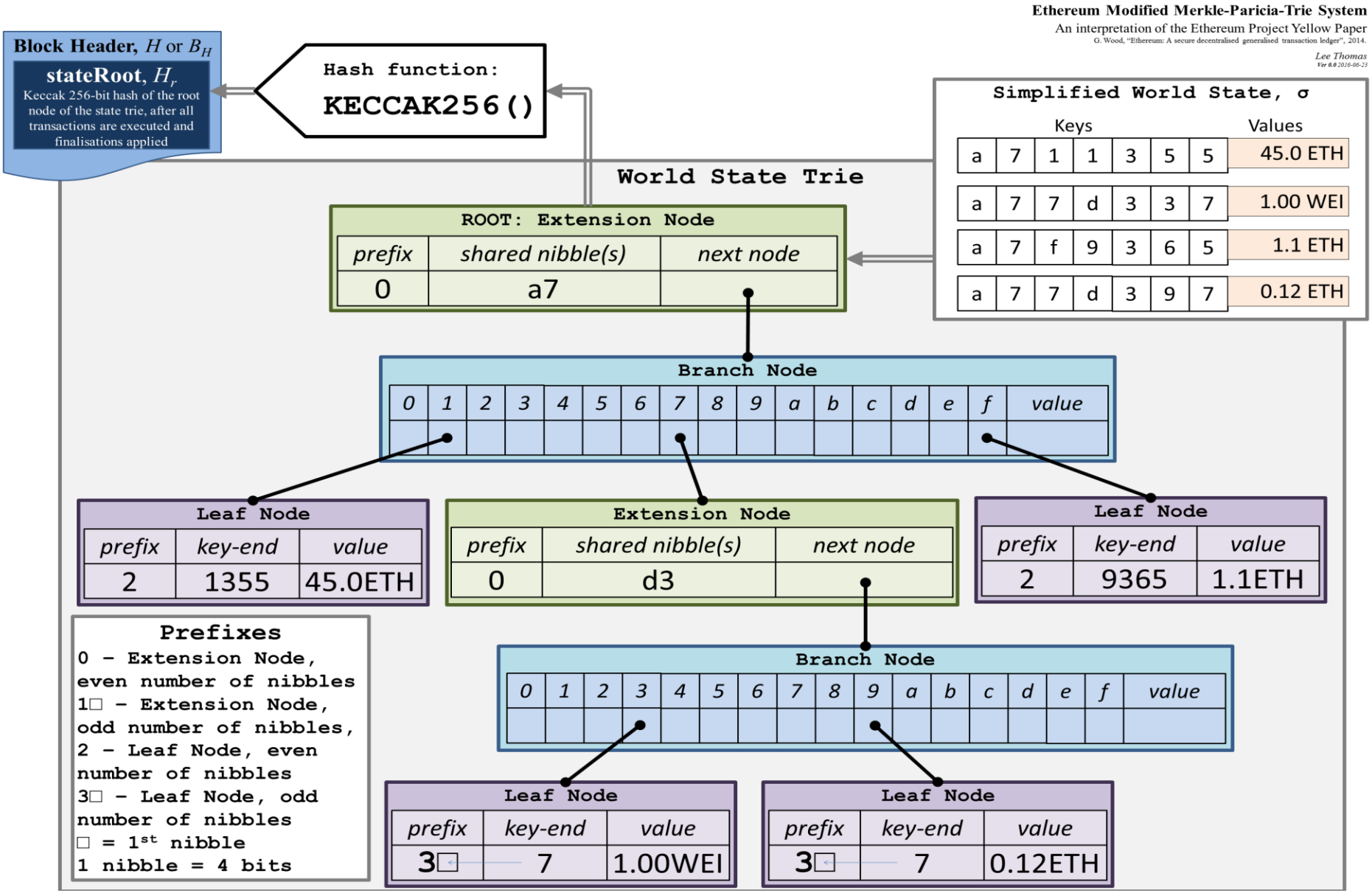


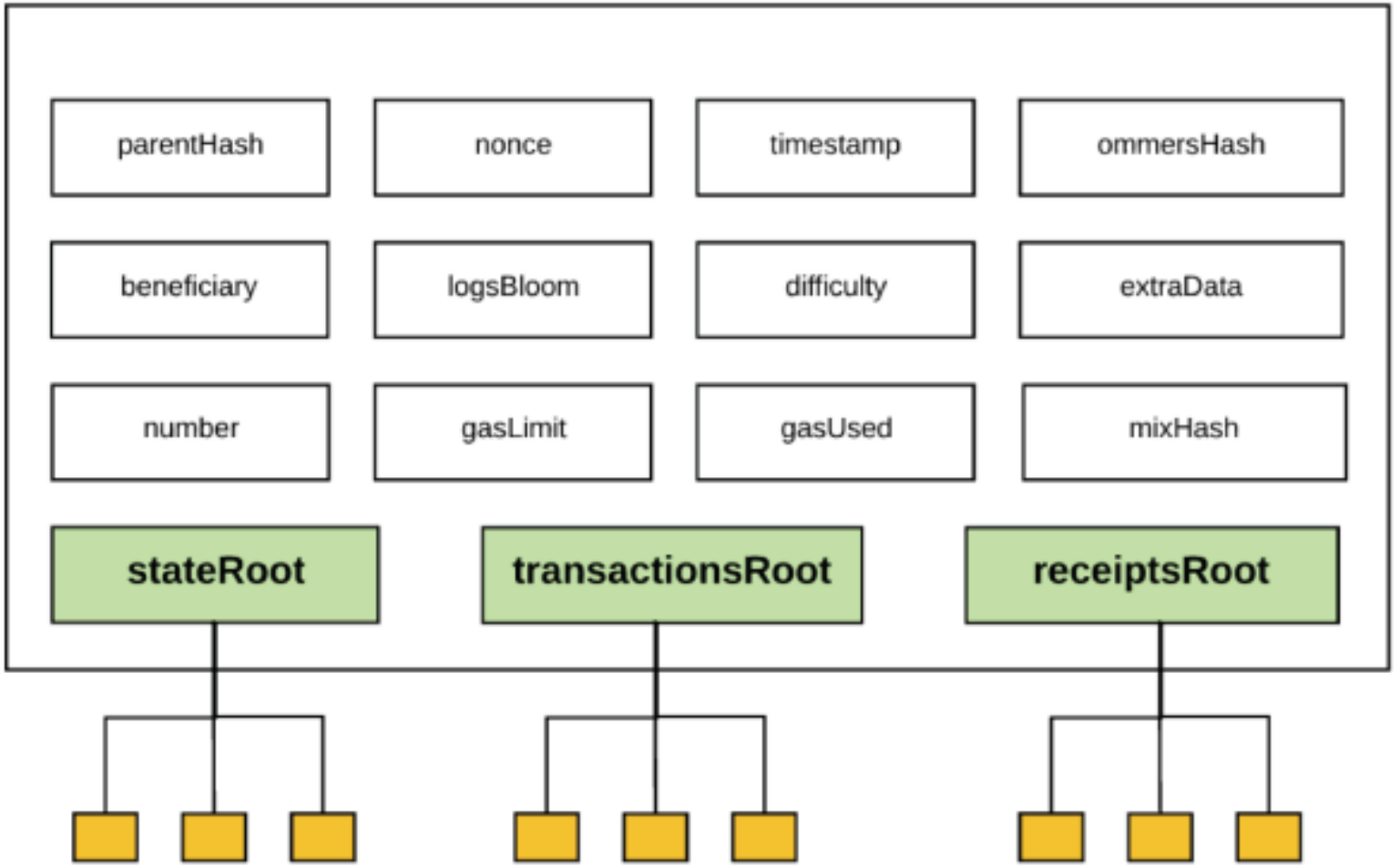
Image credit: V. Buterin



Merkle Patricia Trie

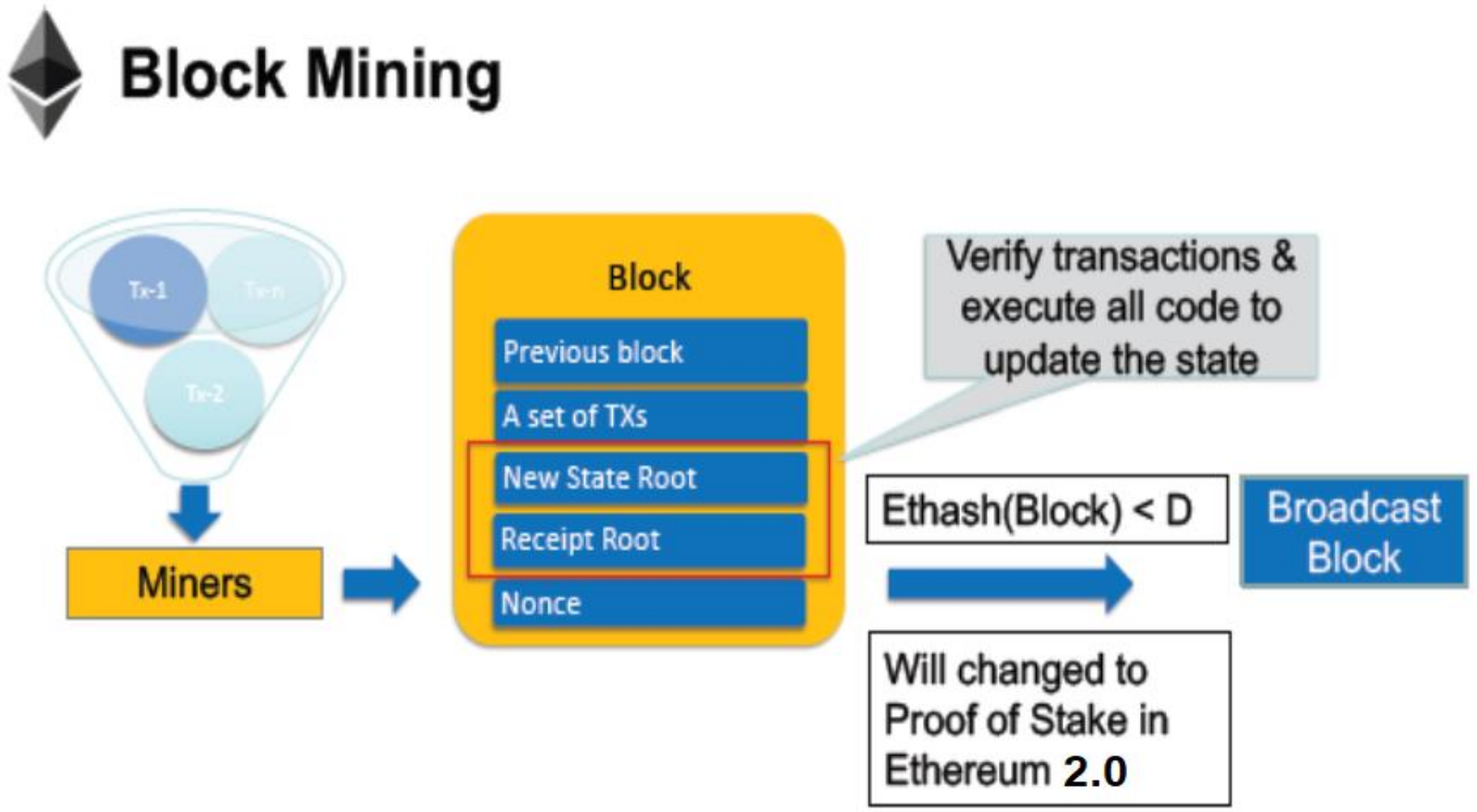


Block Header



Ethereum's PoW

- Follows Ethash hashing algorithm



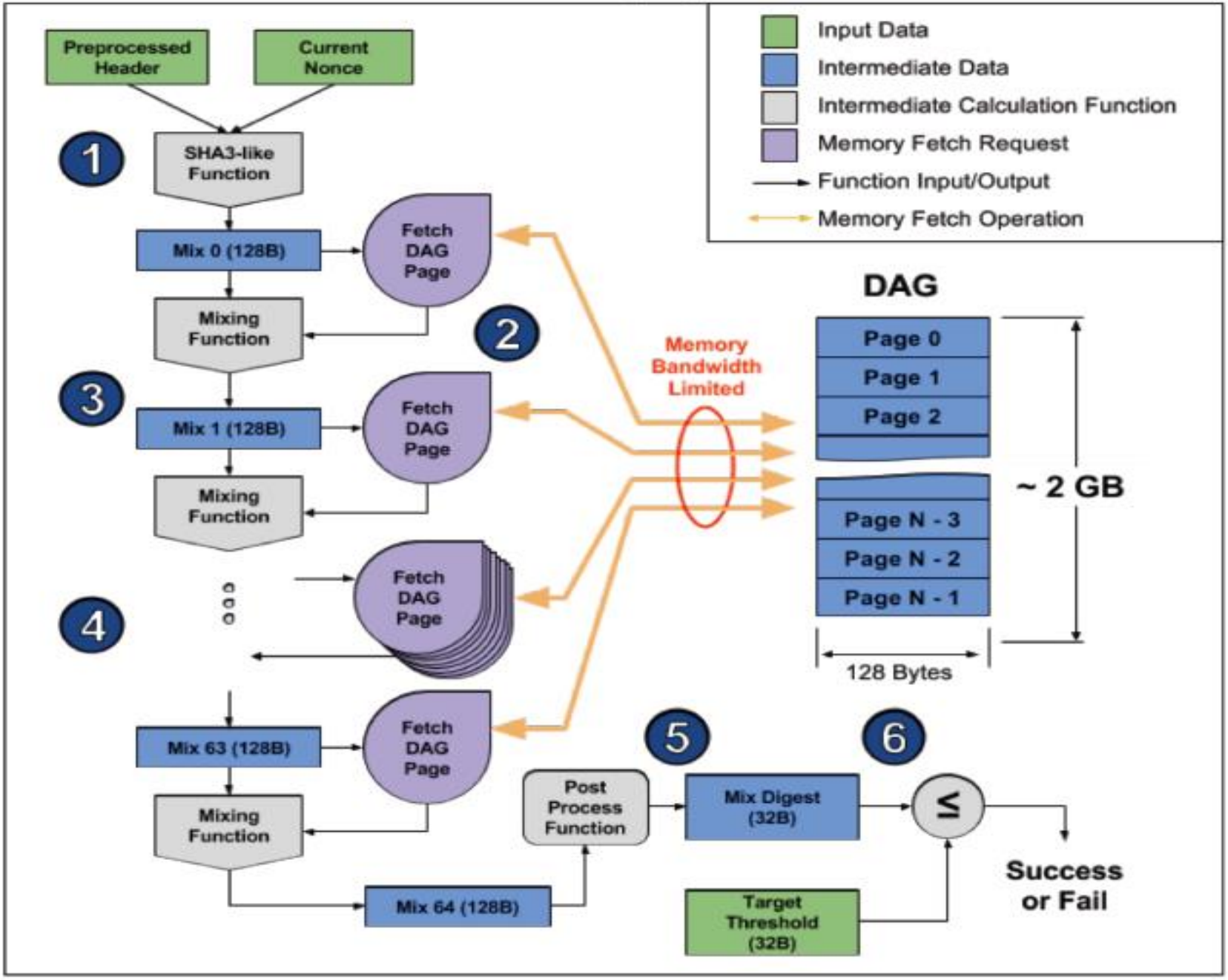
Ethereum's PoW

- Multilevel DAG Construction

- Light nodes only need to store the cache for verification.
- They can efficiently verify a transaction without storing the entire blockchain dataset.

- Each item in the dataset depends on only a small number of items from the cache.
- The dataset DAG grows linearly with time.
- Miners need to store this entire dataset DAG.

Ethash Hashing Algorithm

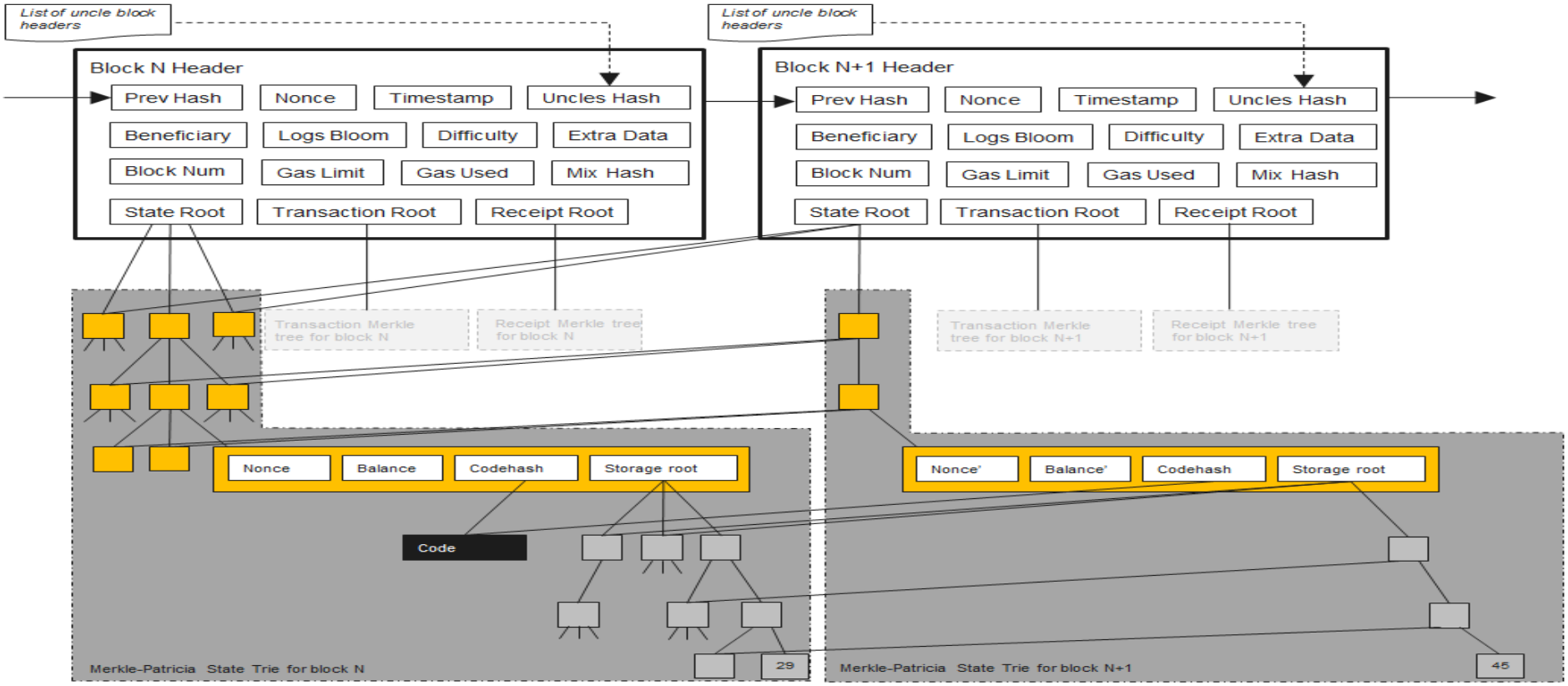


Computation Vs. Memory Access

Why is Ethash Memory Hard?

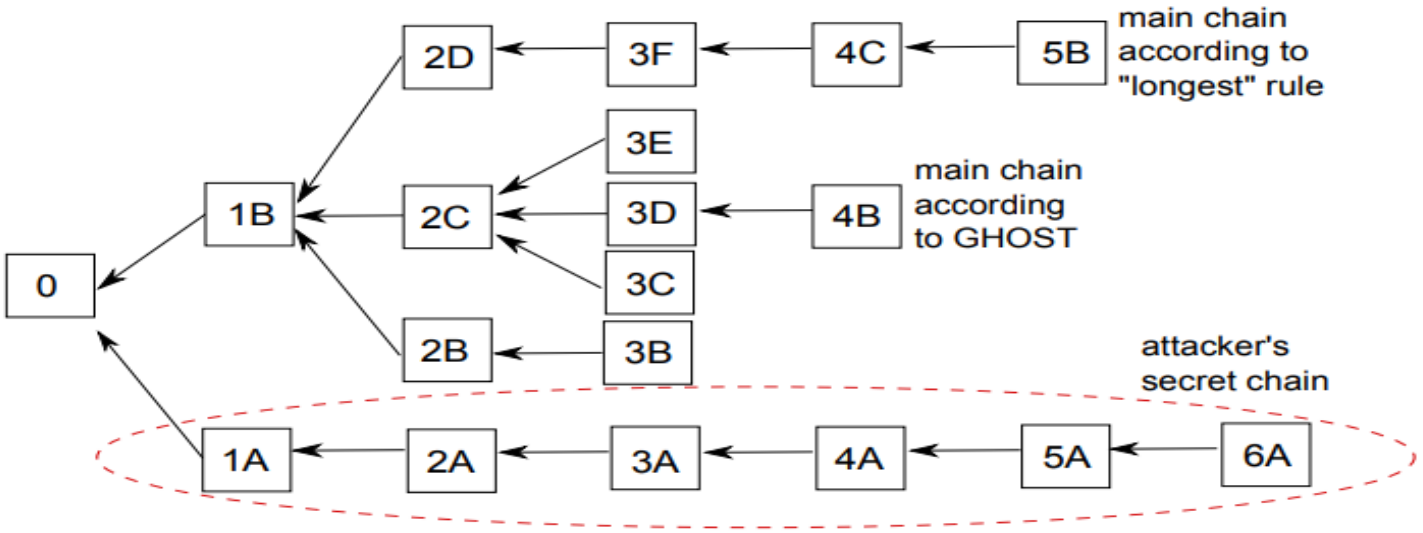
- Every mixing operation requires a 128 byte read from the DAG.
- Hashing a single nonce requires 64 mixes, resulting in (128 Bytes x 64) = 8 KB of memory read.
- The reads are random access, so putting a small chunk of the DAG in an L1 or L2 cache isn't going to help much.
- Fetching the DAG pages from memory is much slower than the mixing computation
- The best way to speed up the ethash hashing algorithm is to speed up the 128 byte DAG page fetches from memory.
- Thus, we consider the ethash algorithm to be memory hard.

Ethereum Blockchain



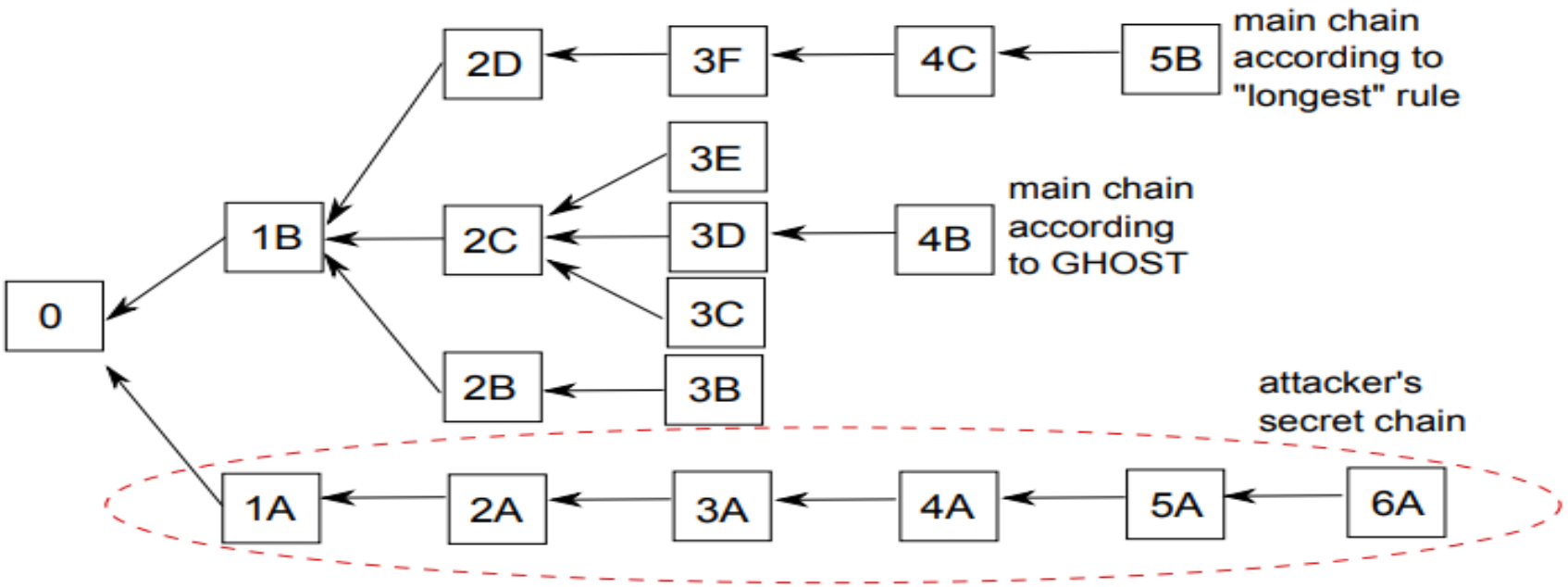
Ethereum's PoW

- Greedy Heaviest Observed Subtree (GHOST) by Yonatan and Aviv (Dec 2013)
- Fork creation is inevitable; How to deal with it?
- Bitcoin considers LONGEST CHAIN. What about Ethereum?



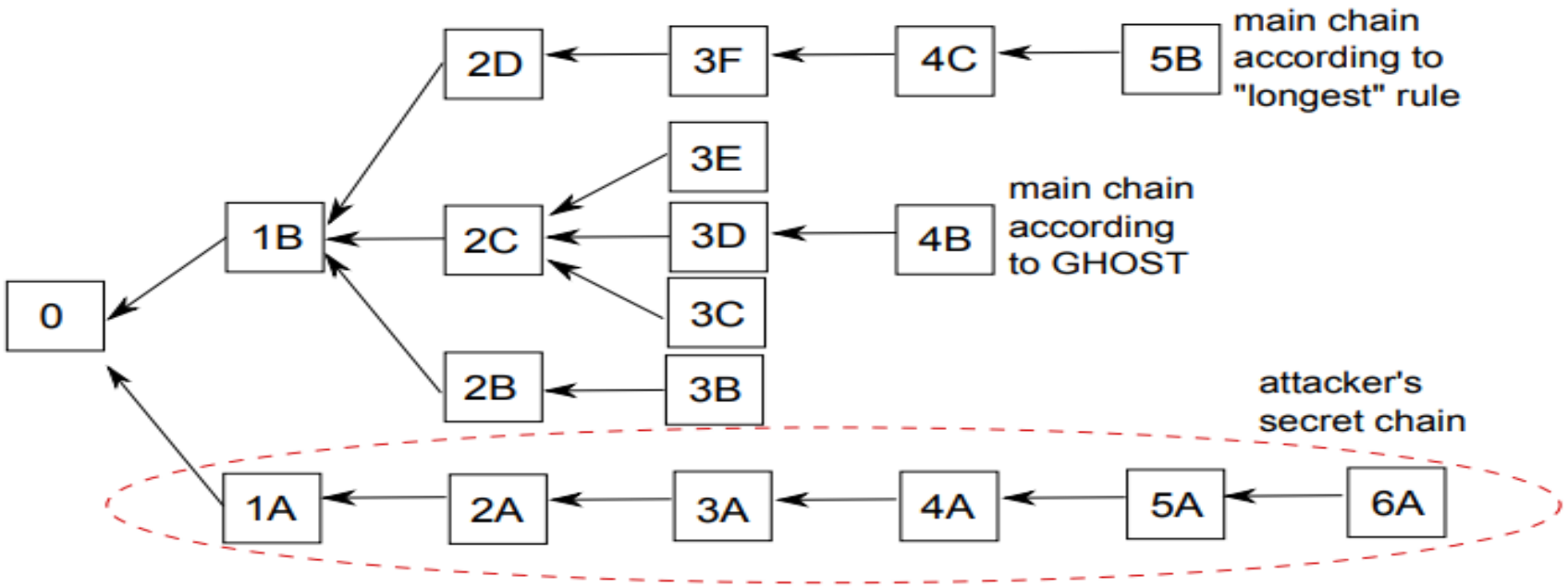
Ethereum's PoW

- Blockgap is 15 sec (much less than bitcoin which is 10 min)
  [Source: https://etherscan.io/chart/blocktime]
- More competing Blocks → More no. of orphaned blocks
- Aim to prevent orphaned blocks (blocks discovered by miners with less computing power)



Ethereum's PoW

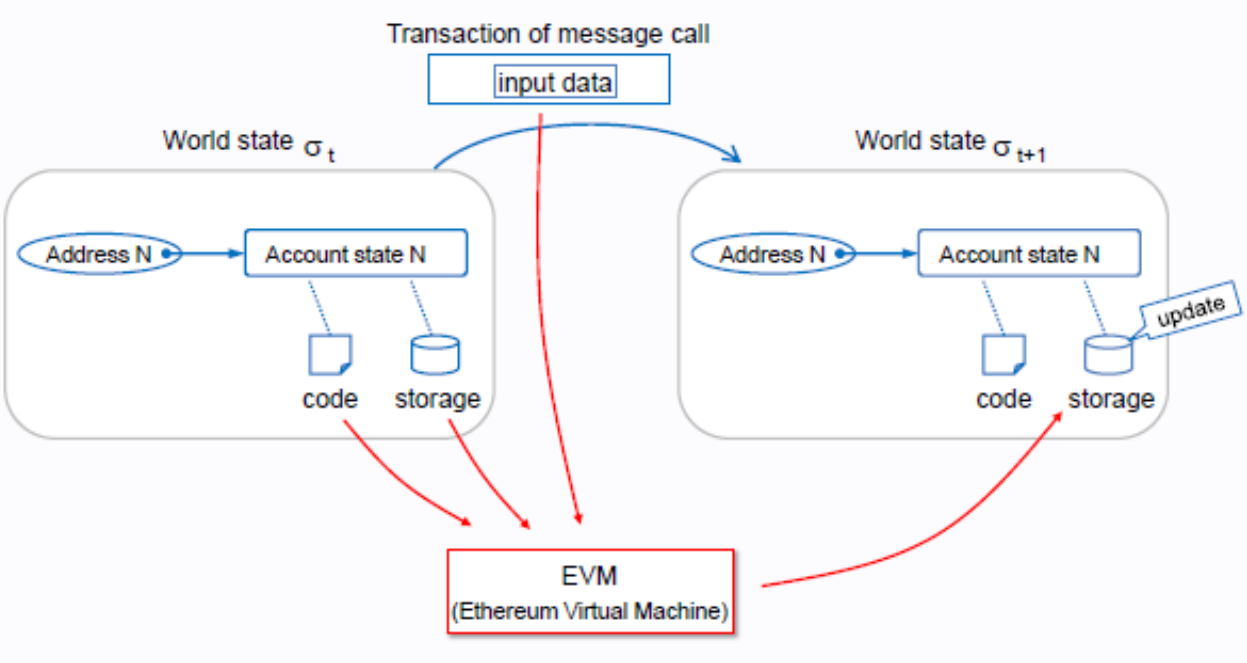
- Uncle Blocks (or Ommer): whose parent is equal to current block's parent's parent
- Consider Tree structure instead of chain
- Choose the Heaviest path as main chain, where weight depends on how dense the subtree is



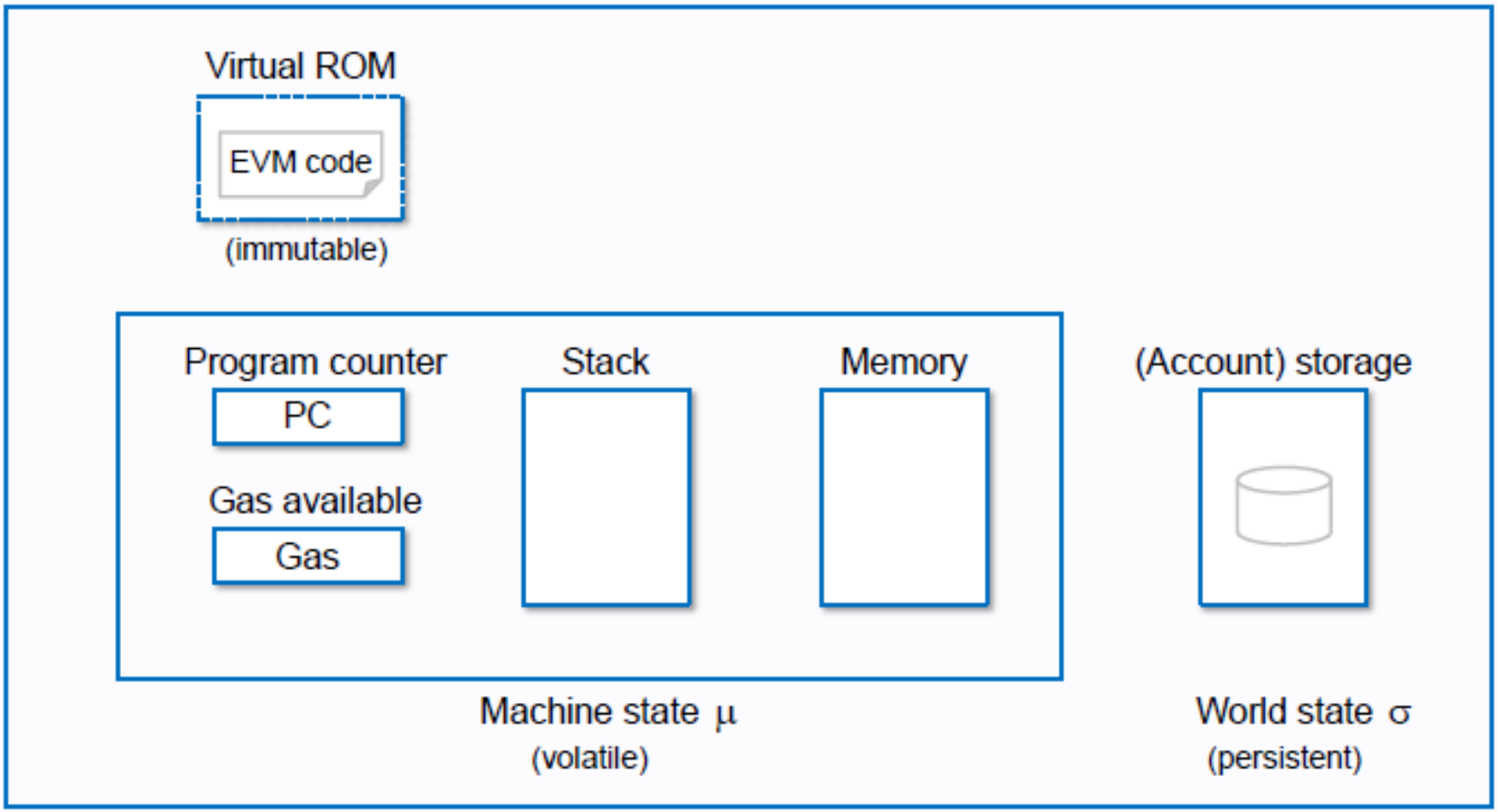
Ethereum Virtual machine (EVM)
&
Smart Contract Execution

Ethereum Virtual Machine (EVM)

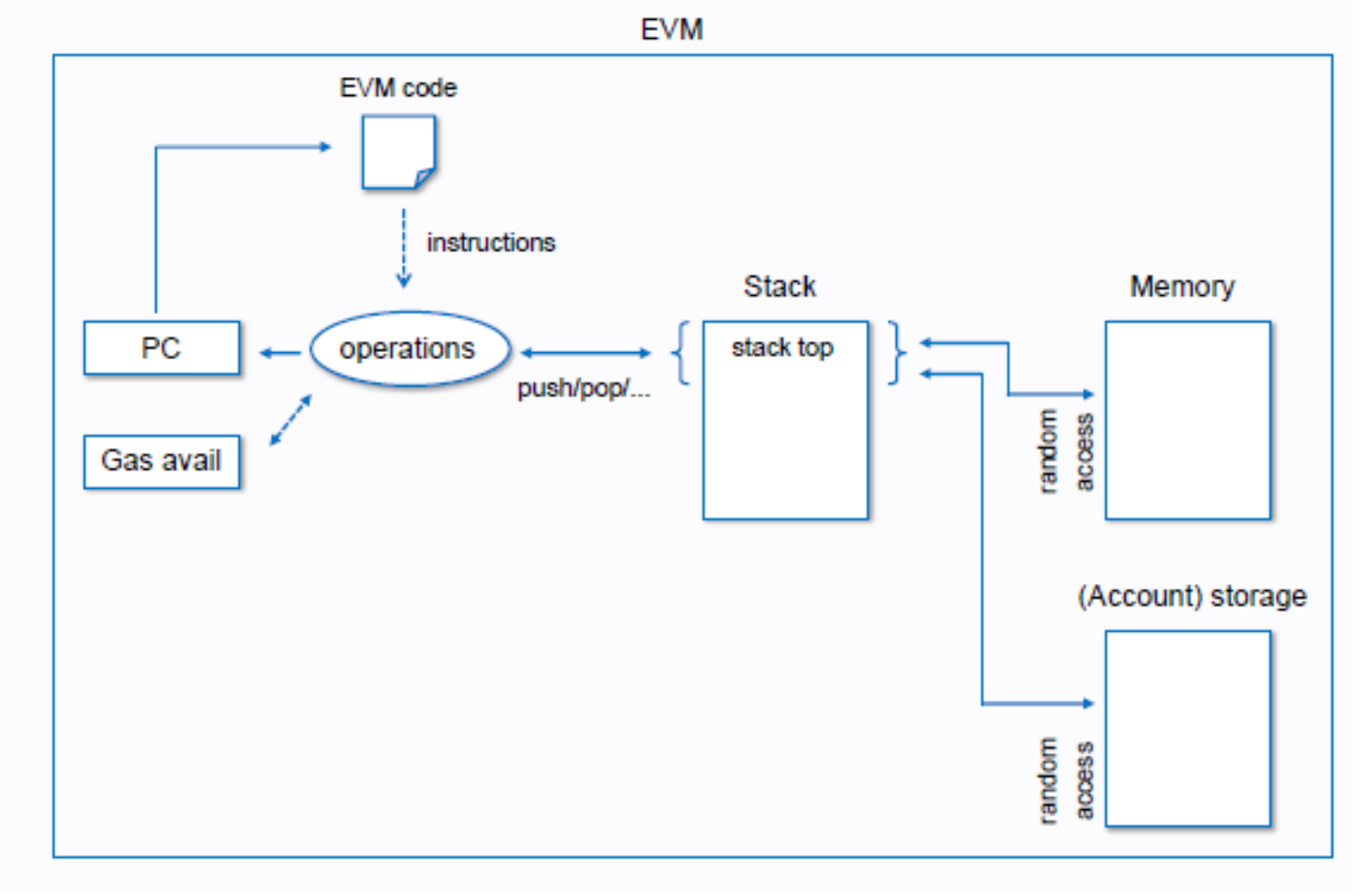
- EVM code is executed on Ethereum Virtual Machine (EVM).
- The Ethereum Virtual Machine is the runtime environment for smart contracts in Ethreum.



EVM: Stack-based Architecture



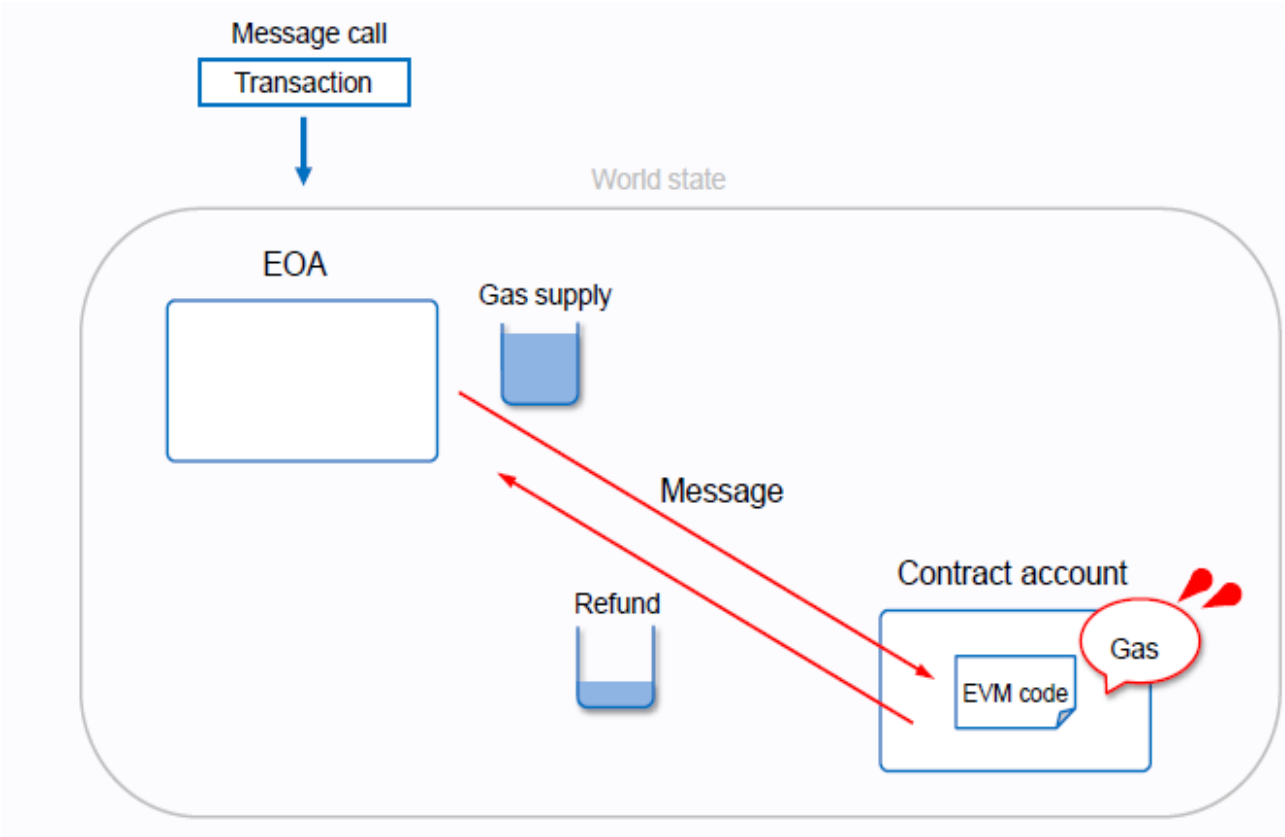
EVM Execution Model

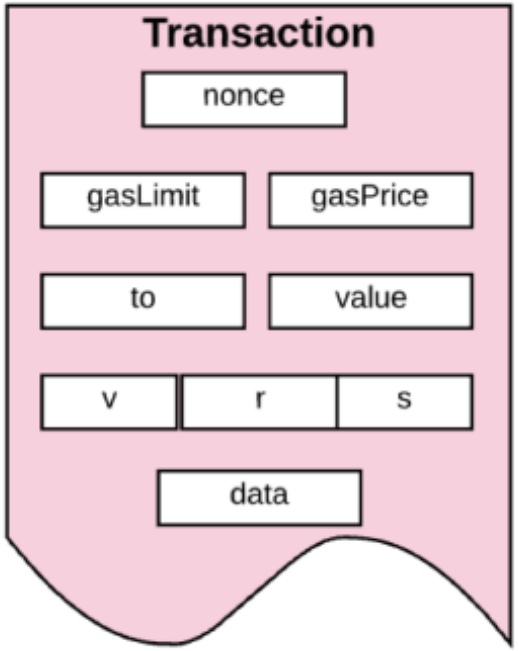


Gas and Fee

- All programmable computation in Ethereum is subject to fees (denominated in gas).

Operation	Gas	GasCost
PUSH1	111741	3
PUSH1	111738	3
MSTORE	111726	12
CALLDATASIZE	111724	2
ISZERO	111721	3
PUSH2	111718	3
JUMPI	111708	10



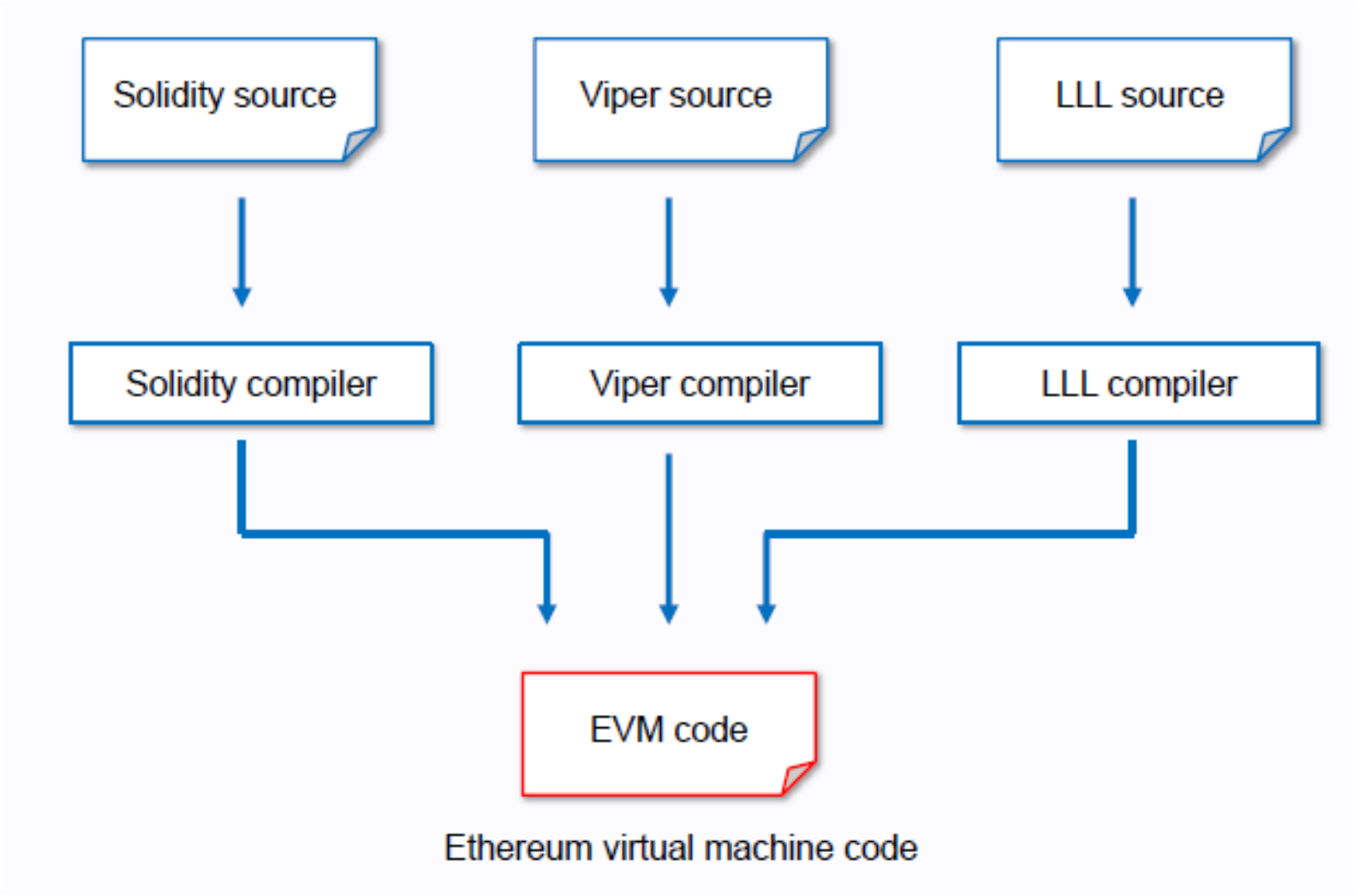


Gas and Fee

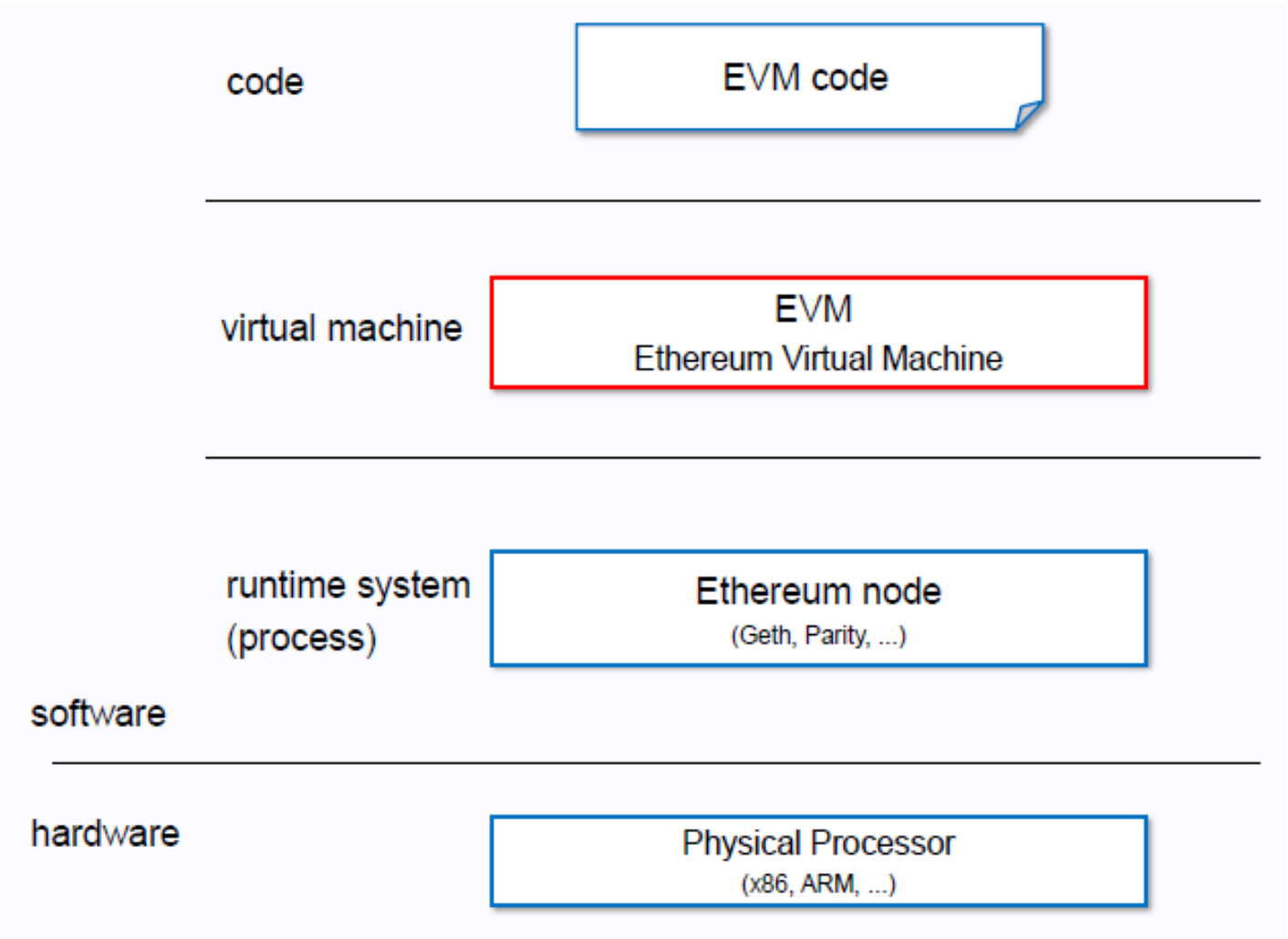
- Gas limit: Max no. of computational steps the transaction is allowed.
- Gas Price: Max fee the sender is willing to pay per computation step.

- Out of Gas Exception, revert the state as if the transaction has never happened
- Sender still pays all the gas
- Note that Block has also a GasLimit (Like Block size in Bitcoin)

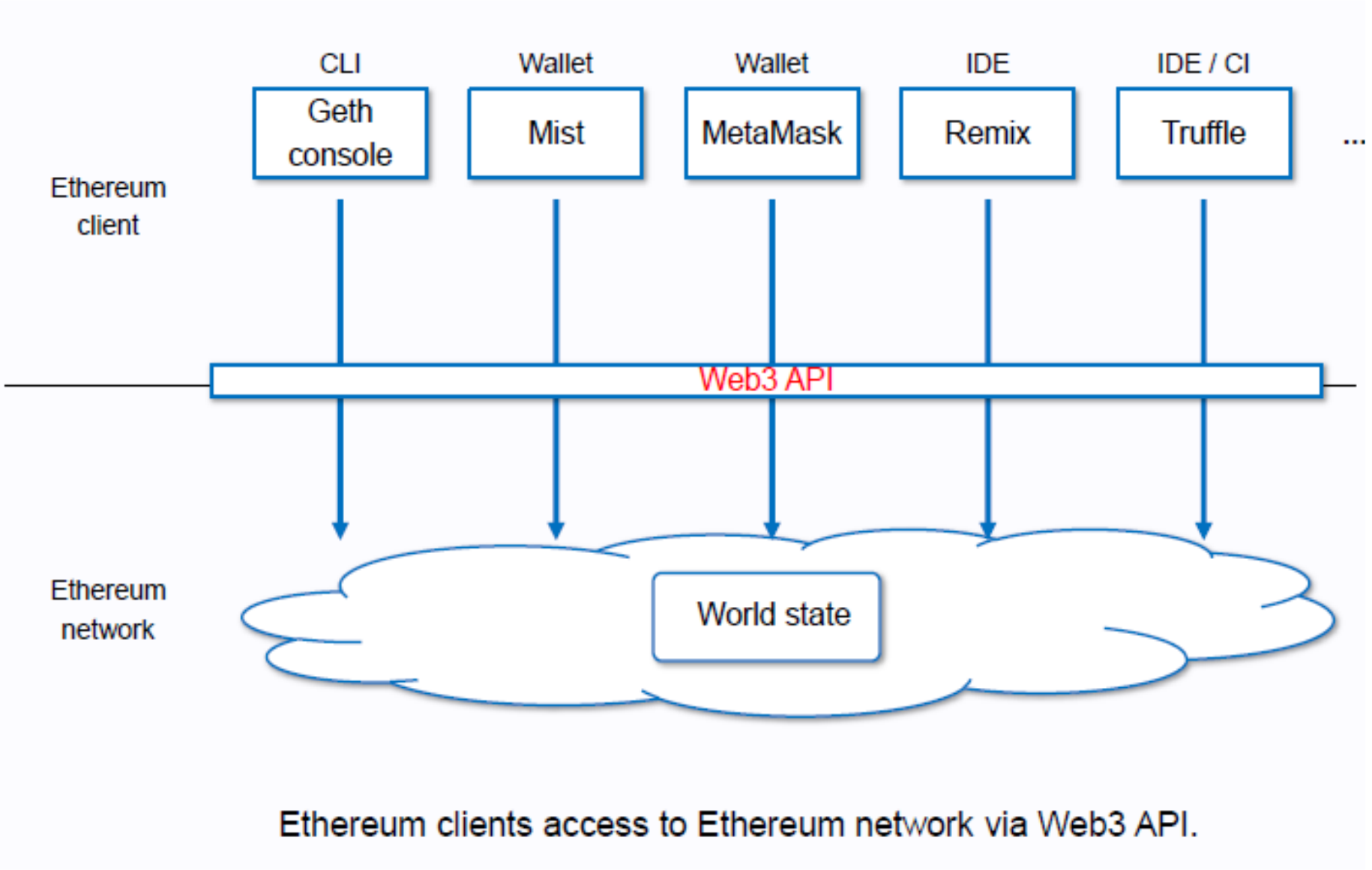
EVM Codes


Solidity source
Viper source
LLL source
Solidity compiler
Viper compiler
LLL compiler
EVM code
Ethereum virtual machine code

EVM Layers

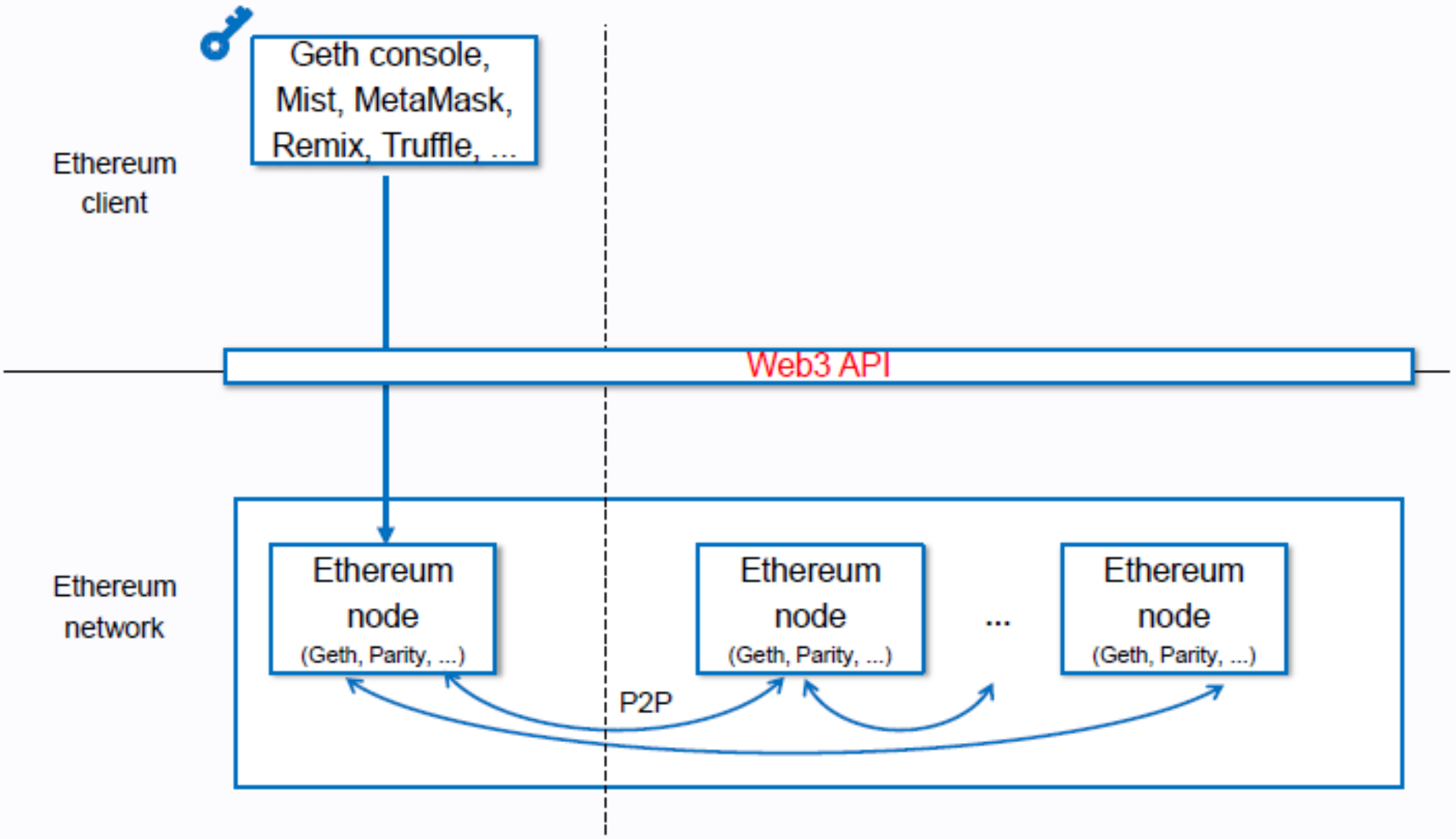

code
EVM code
virtual machine
EVM
Ethereum Virtual Machine
runtime system
(process)
Ethereum node
(Geth, Parity, ...)
software
hardware
Physical Processor
(x86, ARM, ...)

Web3 API and Client



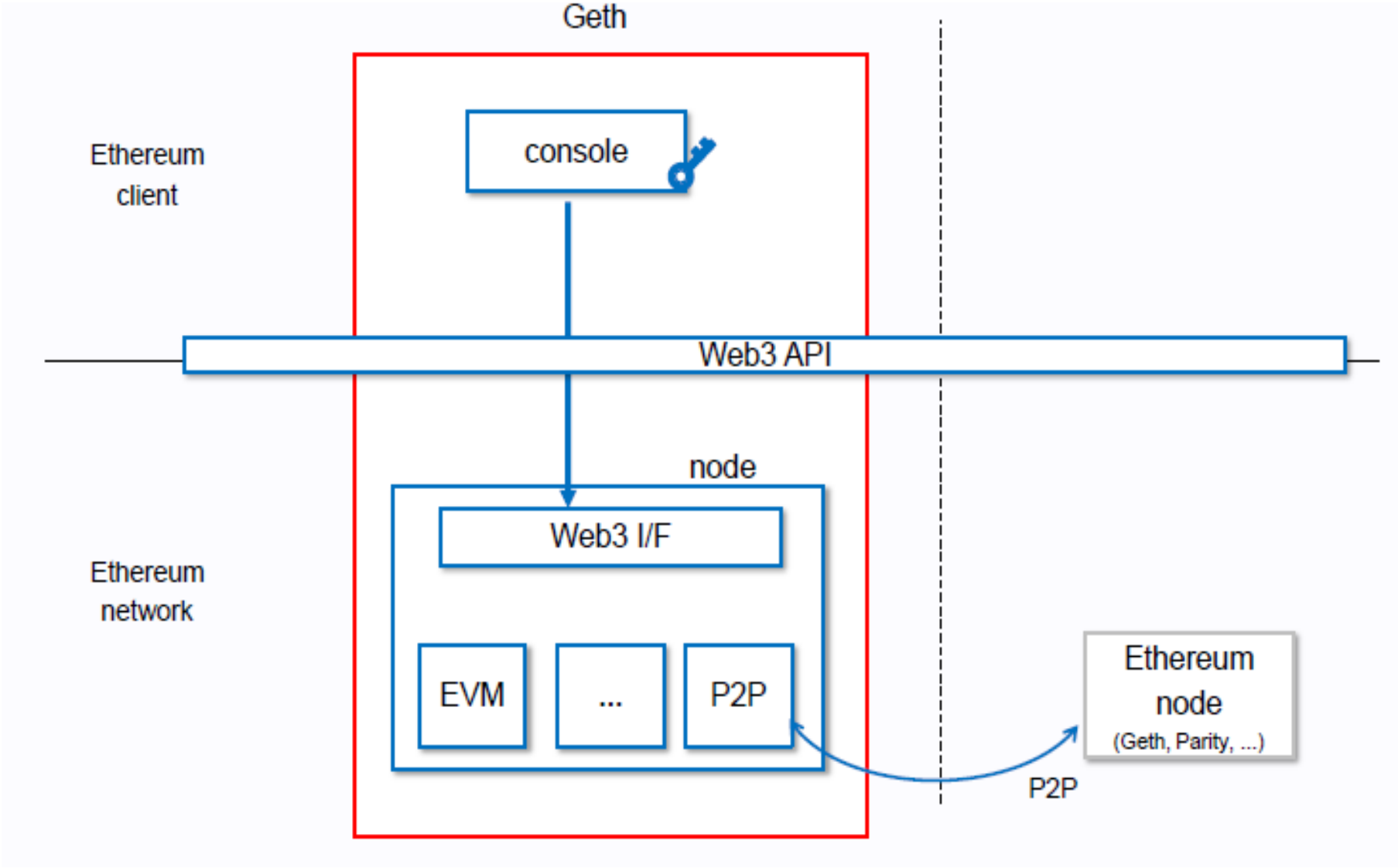
Ethereum clients access to Ethereum network via Web3 API.

Web3 API and Client

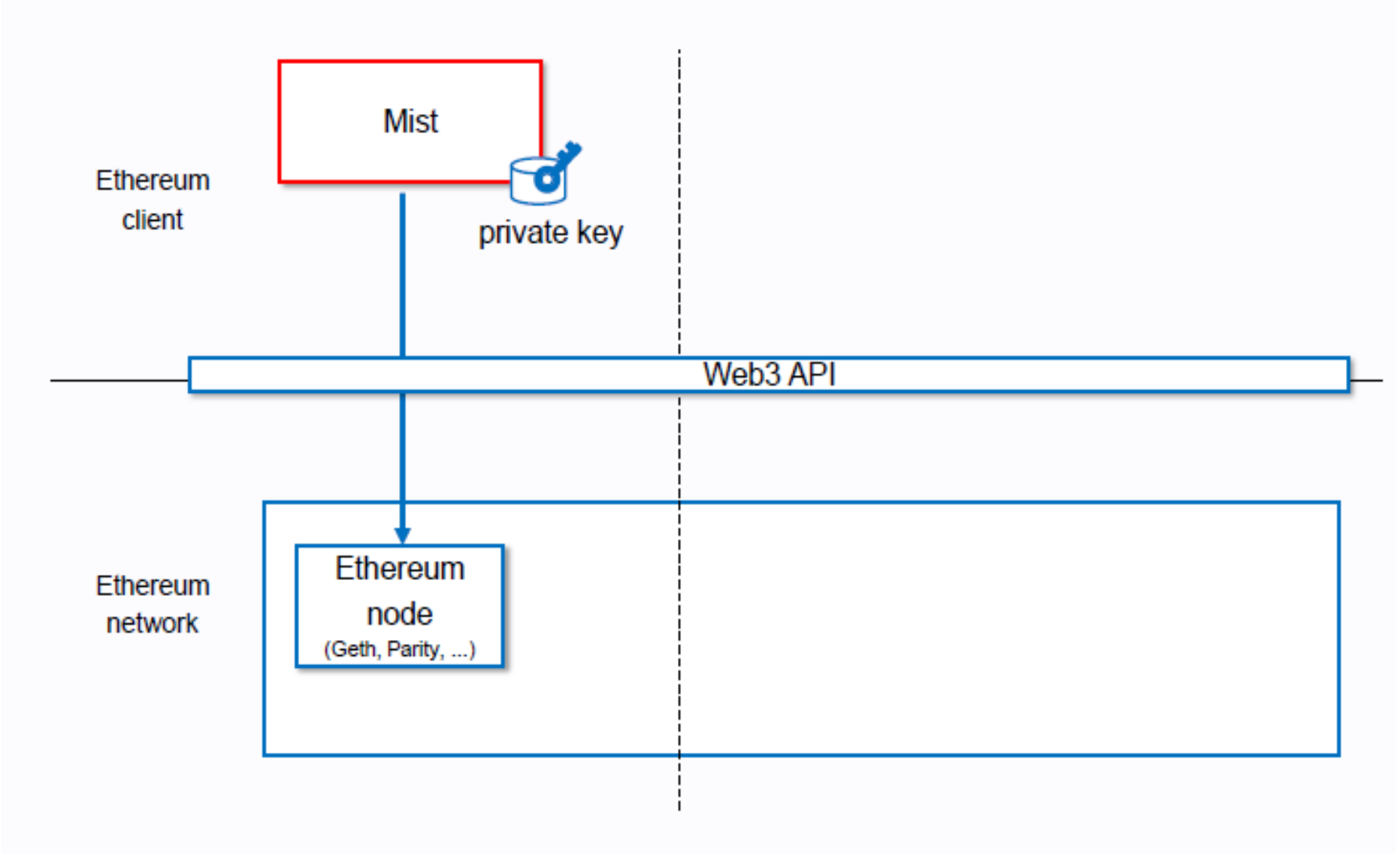


Ethereum clients access to Ethereum network via Web3 API.

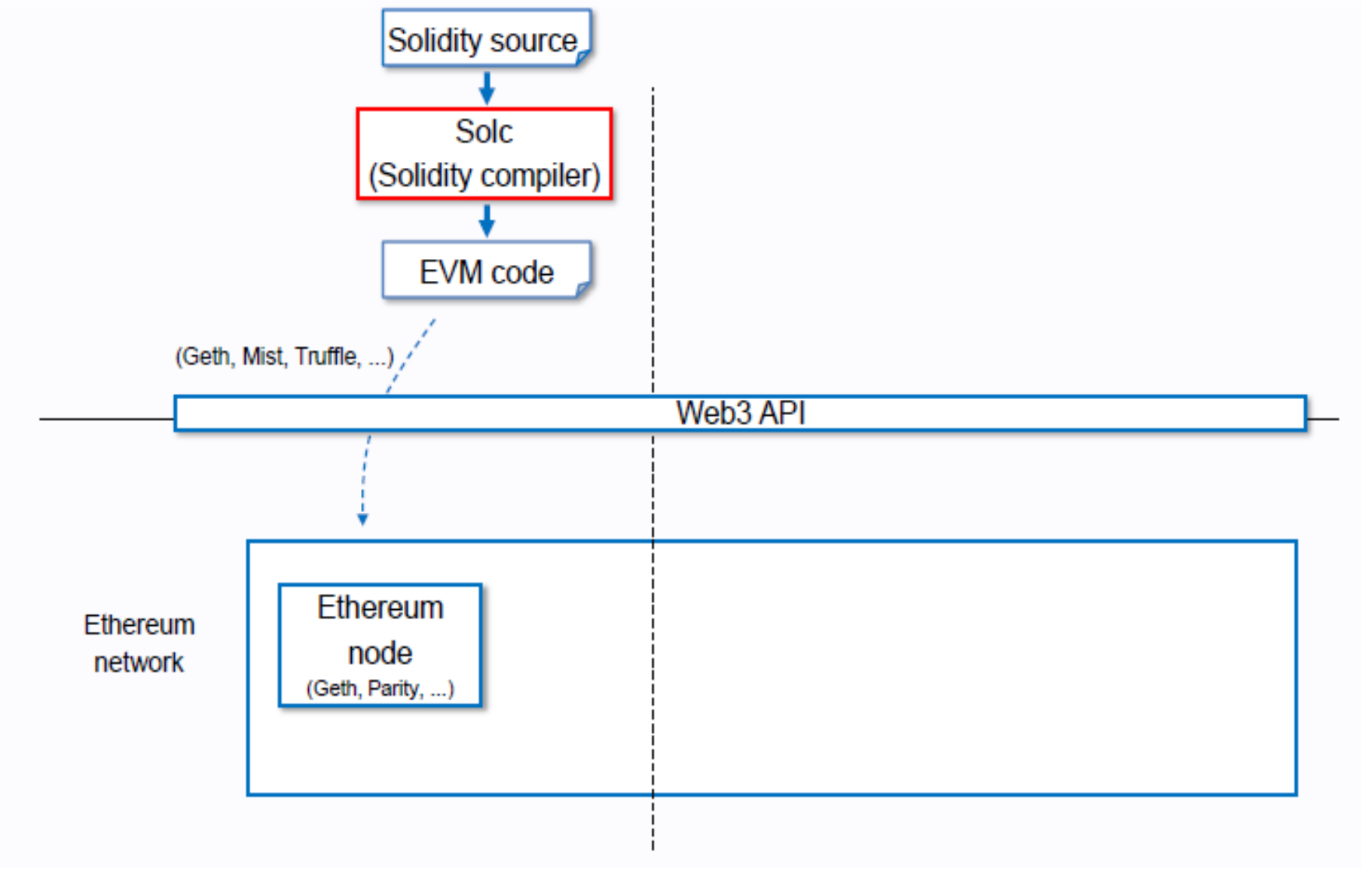
Geth


Geth
Ethereum client
console
Web3 API
node
Web3 I/F
Ethereum network
EVM
...
P2P
Ethereum node
(Geth, Parity, ...)
P2P

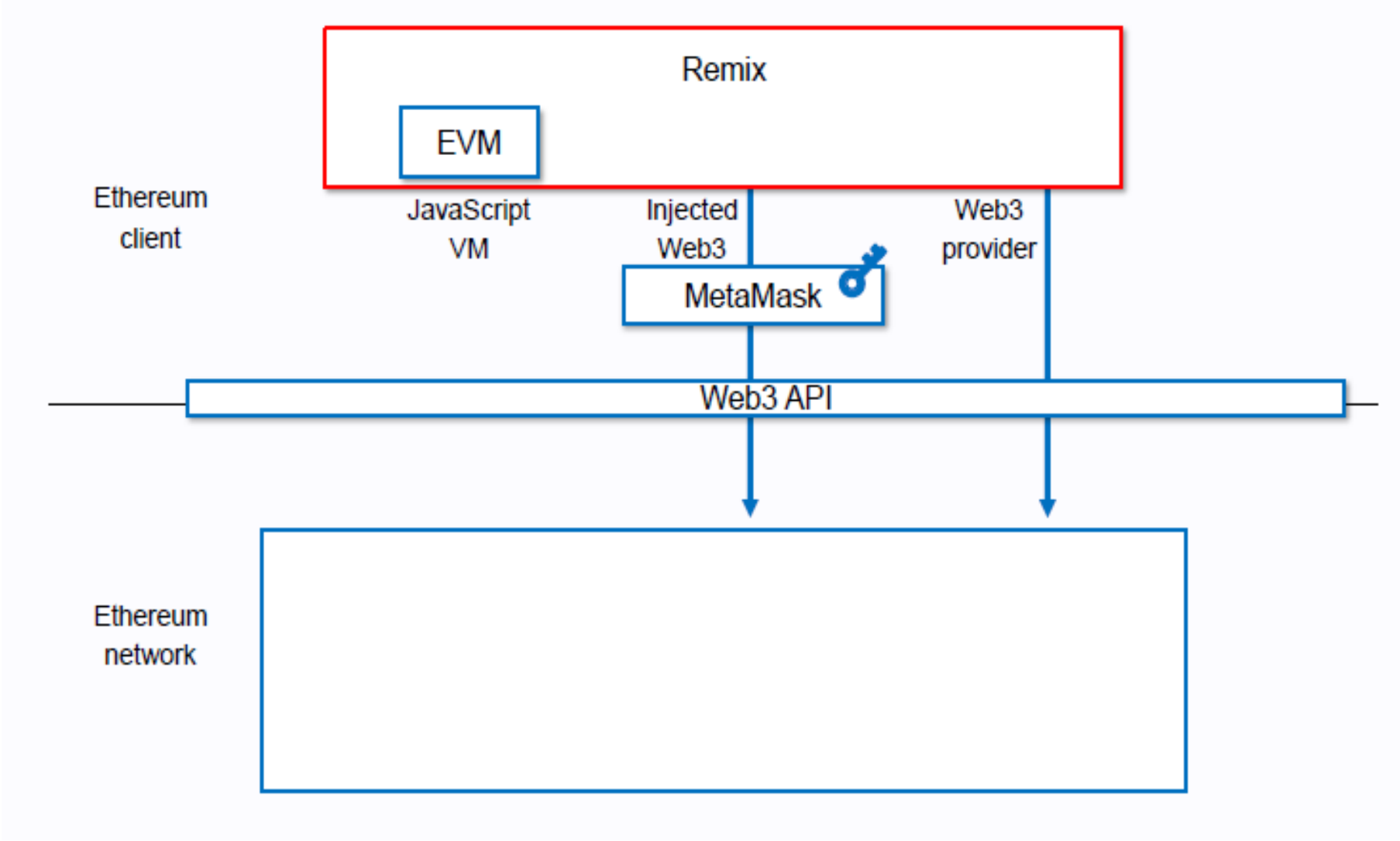
Mist


Mist
private key
Ethereum client
Web3 API
Ethereum node
(Geth, Parity, ...)
Ethereum network

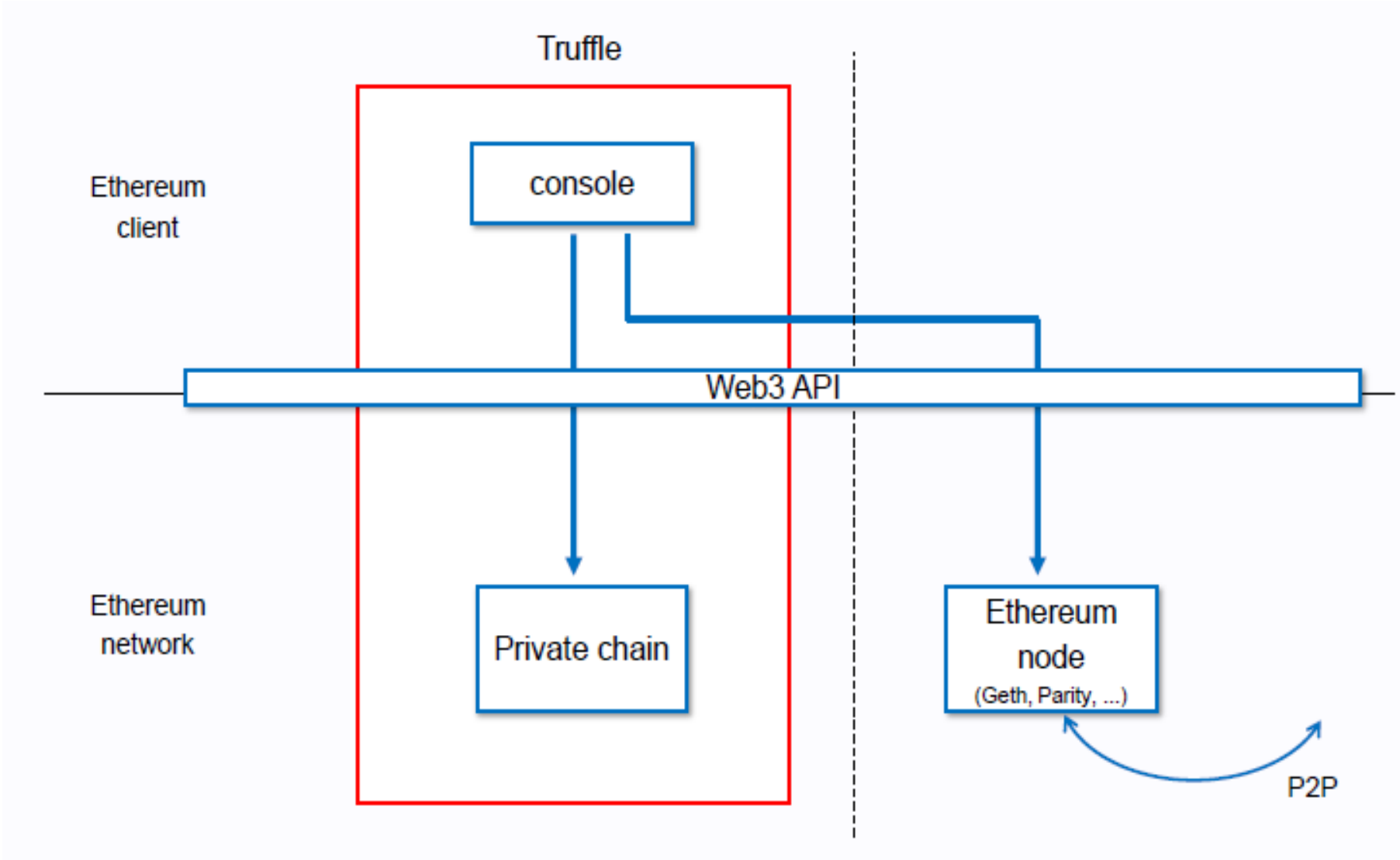
Solc


Solidity source
Solc
(Solidity compiler)
EVM code
(Geth, Mist, Truffle, ...)
Web3 API
Ethereum
network
Ethereum
node
(Geth, Parity, ...)

Remix


Remix
EVM
Ethereum client
JavaScript VM
Injected Web3
Web3 provider
MetaMask
Web3 API
Ethereum network

Truffle


Truffle
Ethereum client
console
Web3 API
Ethereum network
Private chain
Ethereum node
(Geth, Parity, ...)
P2P

Thank You!