Blockchain and Digital Currencies

Lecture 9

PHBS 2024 M3

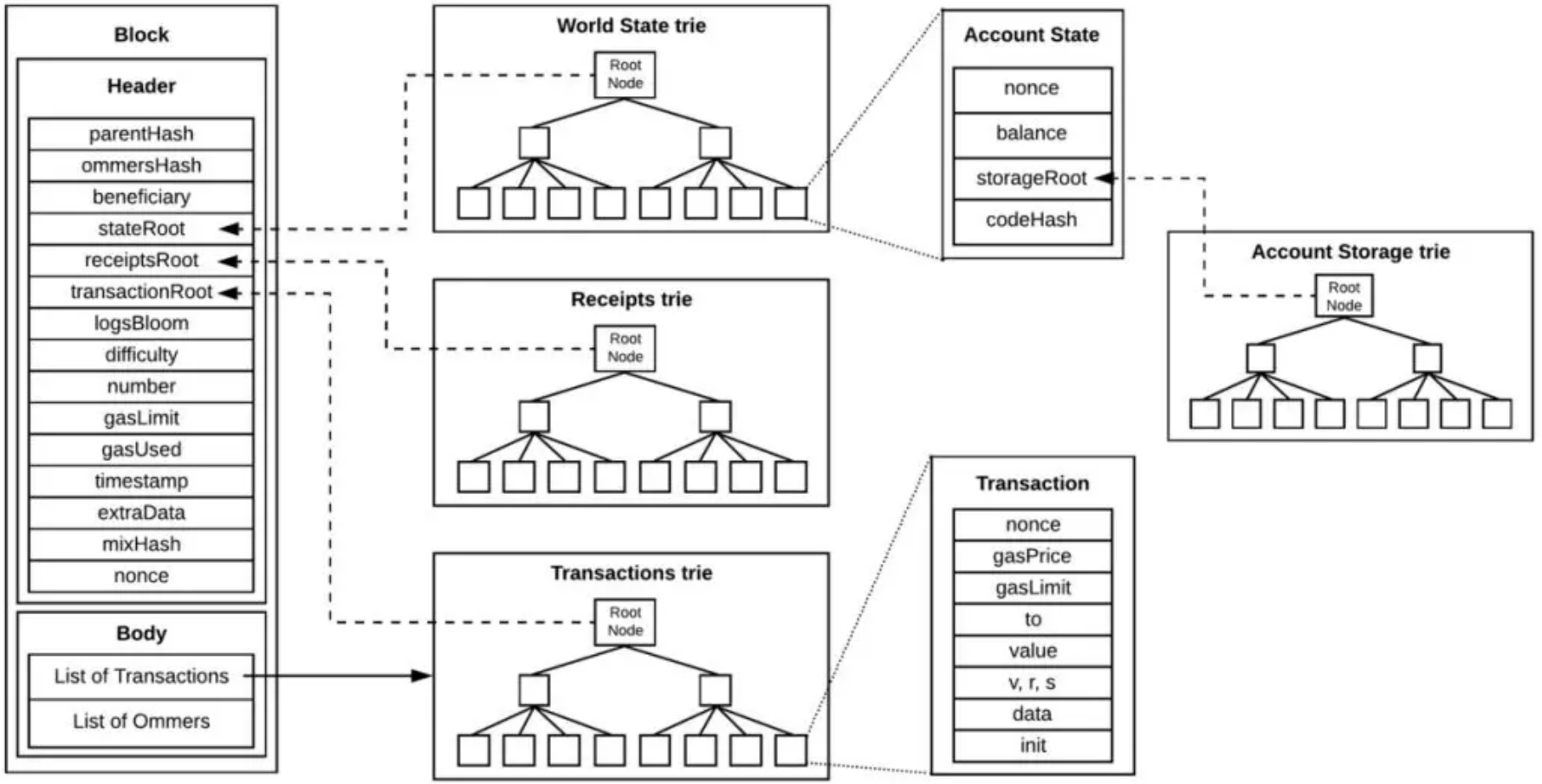
Agenda

- Ethereum Tries
- Reorganized and renamed from Lecture 8

Ethereum Tries

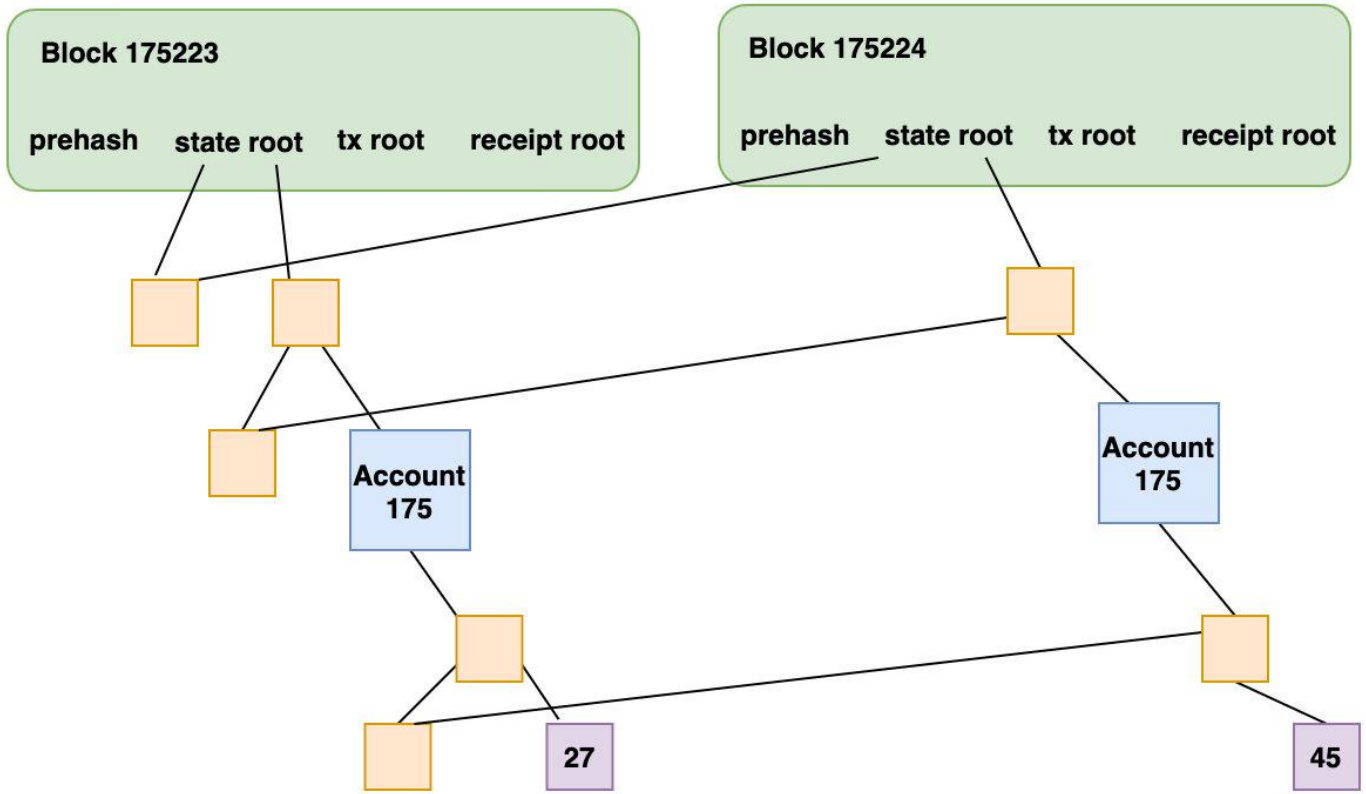
- State root
- Transaction root
- Receipt root
- Storage root

Summary of Tries



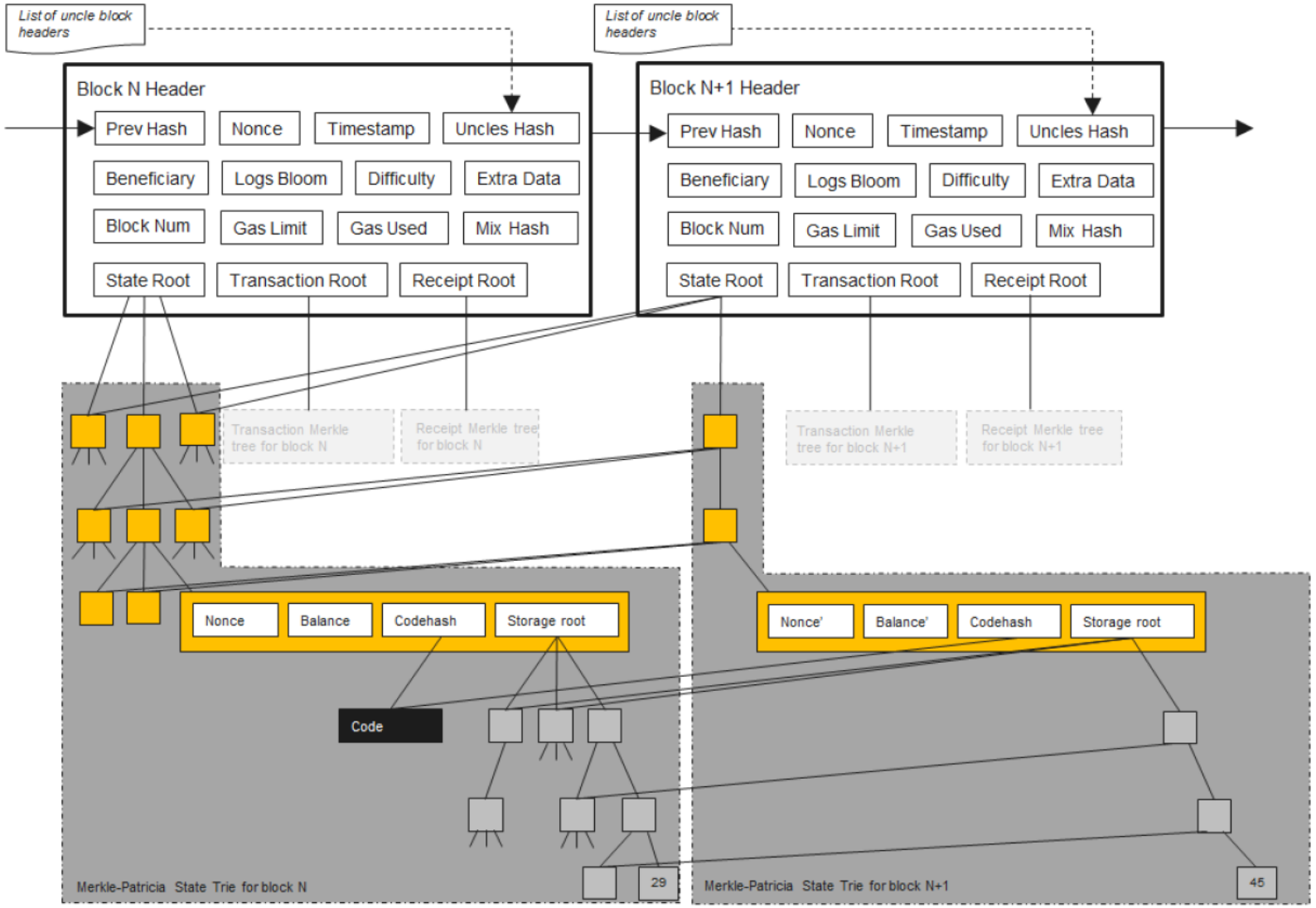
What is the purpose of the tries above?
What are the benefits of using these tries?

Information Saved in State Tries



The state root is the hash of the MPT of all accounts.
A new block only contains modifications caused by transactions.
Why such a structure? -- to support rollback

A More Comprehensive View



Block Header Information

Block Information

```
144  // Block represents an entire block in the Ethereum blockchain.
145  type Block struct {
146          header       *Header
147          uncles       []*Header
148          transactions Transactions
149
150          // caches
151          hash atomic.Value
152          size atomic.Value
153
154          // Td is used by package core to store the total difficulty
155          // of the chain up to and including the block.
156          td *big.Int
157
158          // These fields are used by package eth to track
159          // inter-peer block relay.
160          ReceivedAt   time.Time
161          ReceivedFrom interface{}
162  }
```

指向block header 的指针

指向叔父区块的指针

交易列表

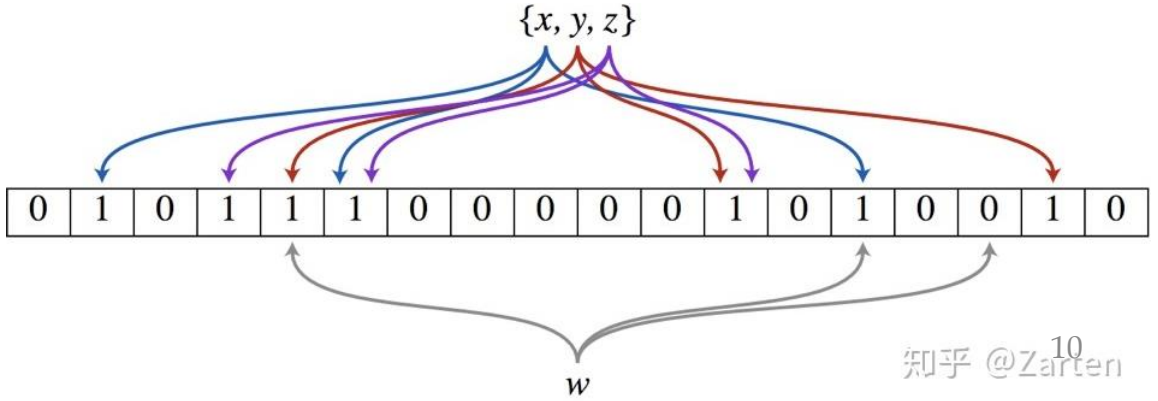
Complete block struct

Published block struct

```
177  // "external" block encoding. used for eth protocol, etc.
178  type extblock struct {
179          Header *Header
180          Txs    []*Transaction
181          Uncles []*Header
182  }
```

Bloom Filter

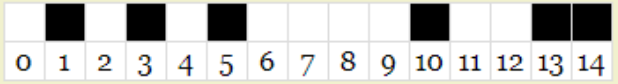
- A Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set.
  - Easy to check the **non-existence** of elements
  - False positive matches are possible
  - Very hard to delete elements



Bloom Filter

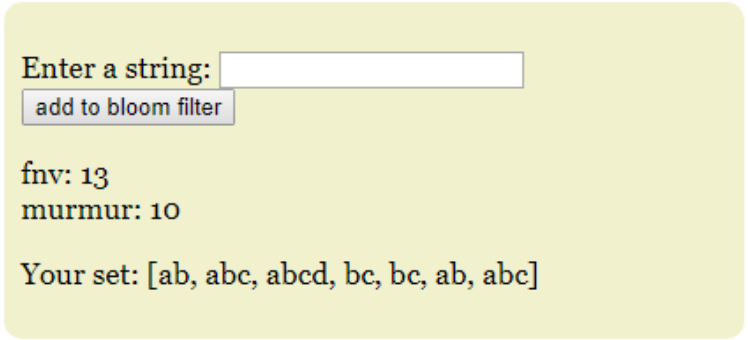
- Demonstration from https://llimllib.github.io/bloomfilter-tutorial/
- Easy to check the existence of elements but very hard to delete

The base data structure of a Bloom filter is a **Bit Vector**. Here's a small one we'll use to demonstrate:



Each empty cell in that table represents a bit, and the number below it its index. To add an element to the Bloom filter, we simply hash it a few times and set the bits in the bit vector at the index of those hashes to 1.

It's easier to see what that means than explain it, so enter some strings and see how the bit vector changes. Fnv and Murmur are two simple hash functions:

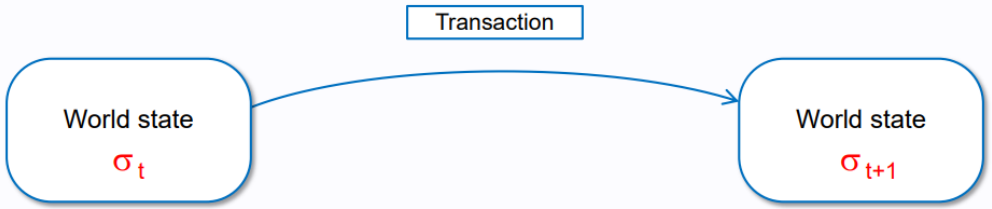


What might be an application of this data structure?

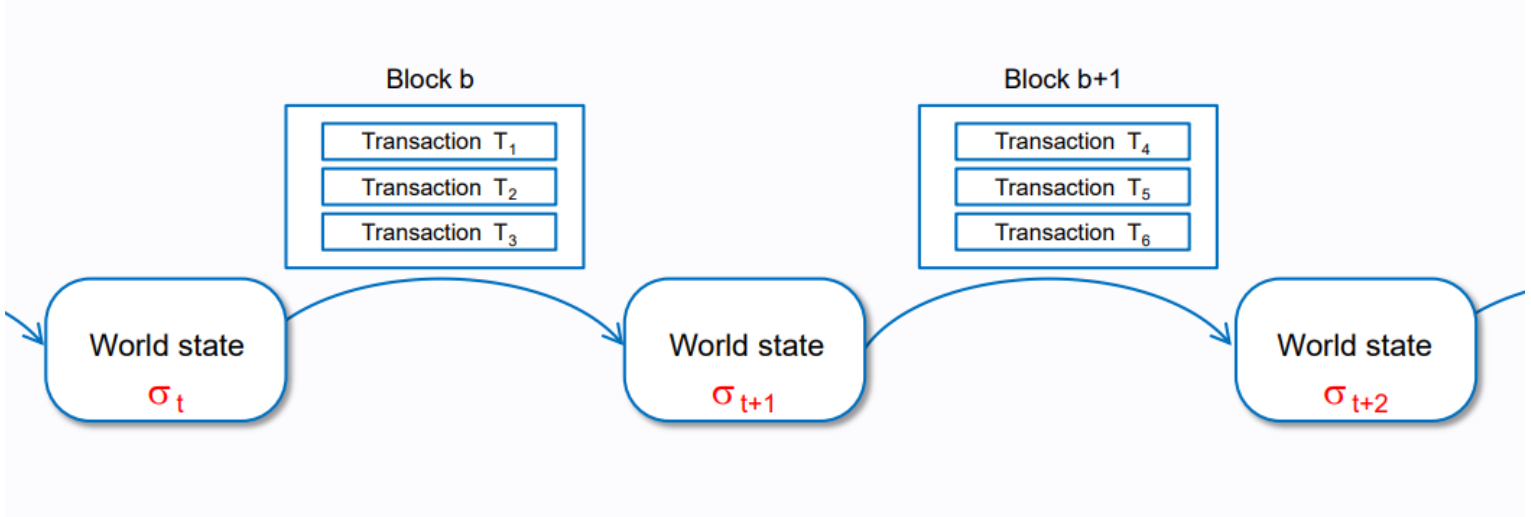
World State Revisited

- The following slides are from a very good reference:
- https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

World State Revisited



Ethereum can be viewed as a transaction-based state machine.

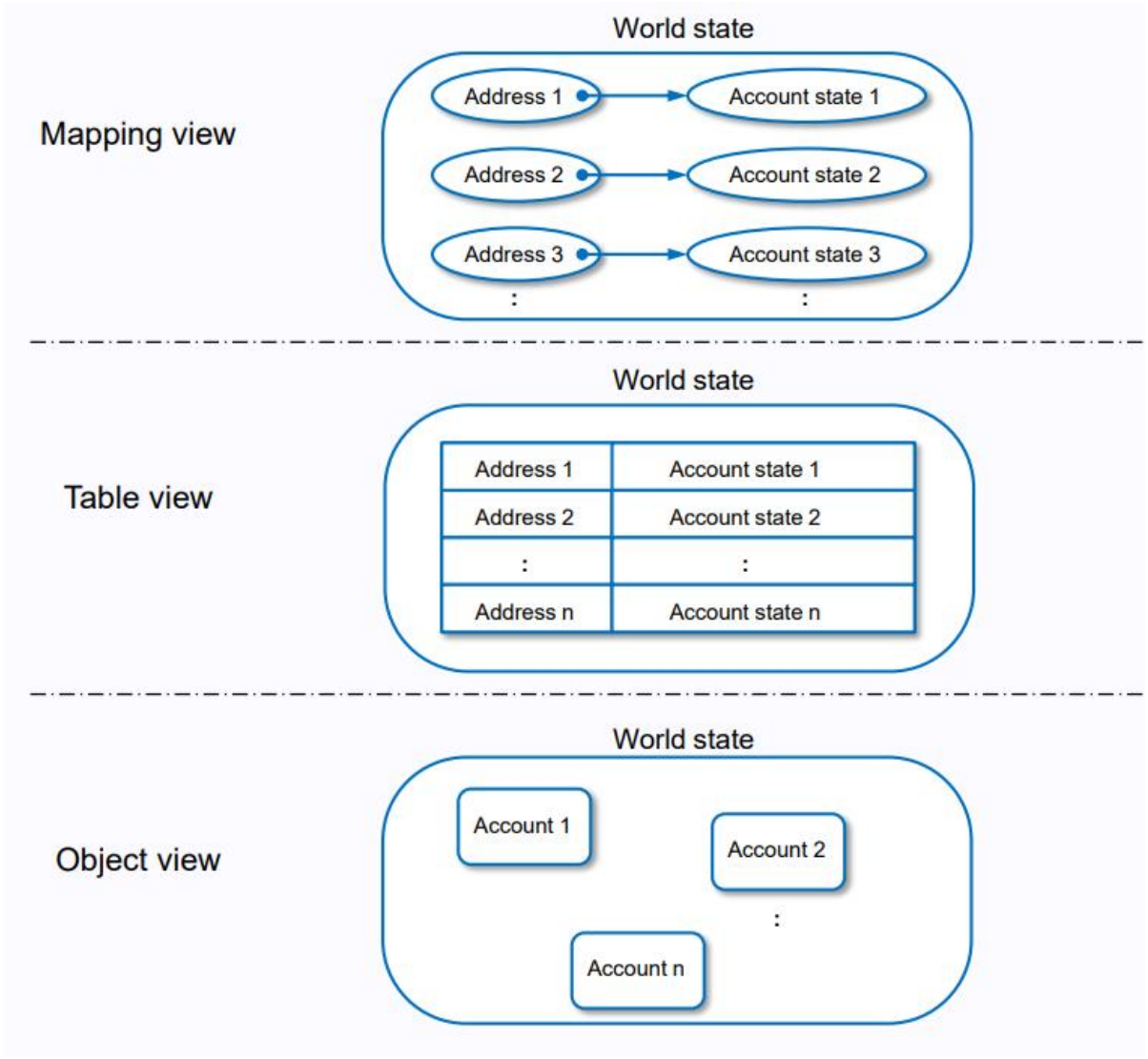


Different Views of World State

Conceptual view

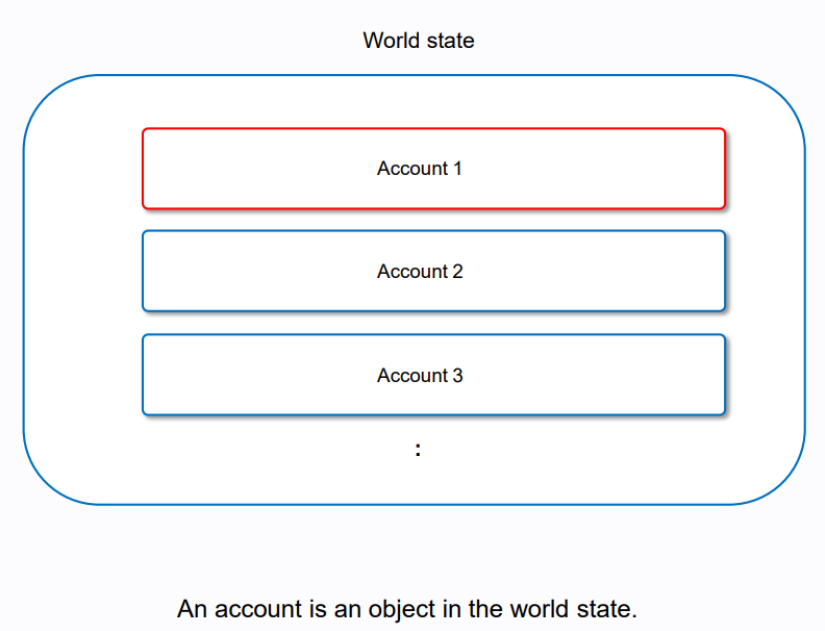
Database view

Coding view



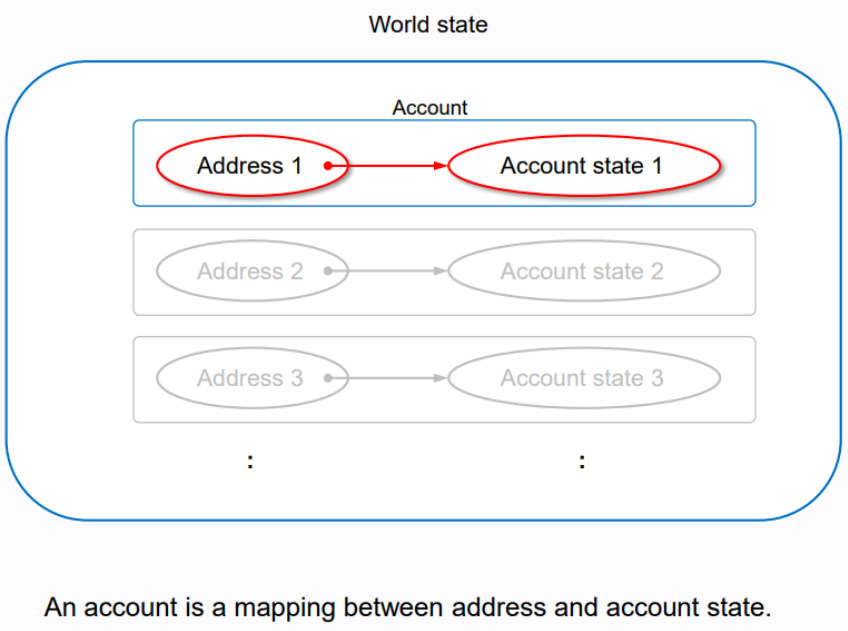
Accounts

Coding view



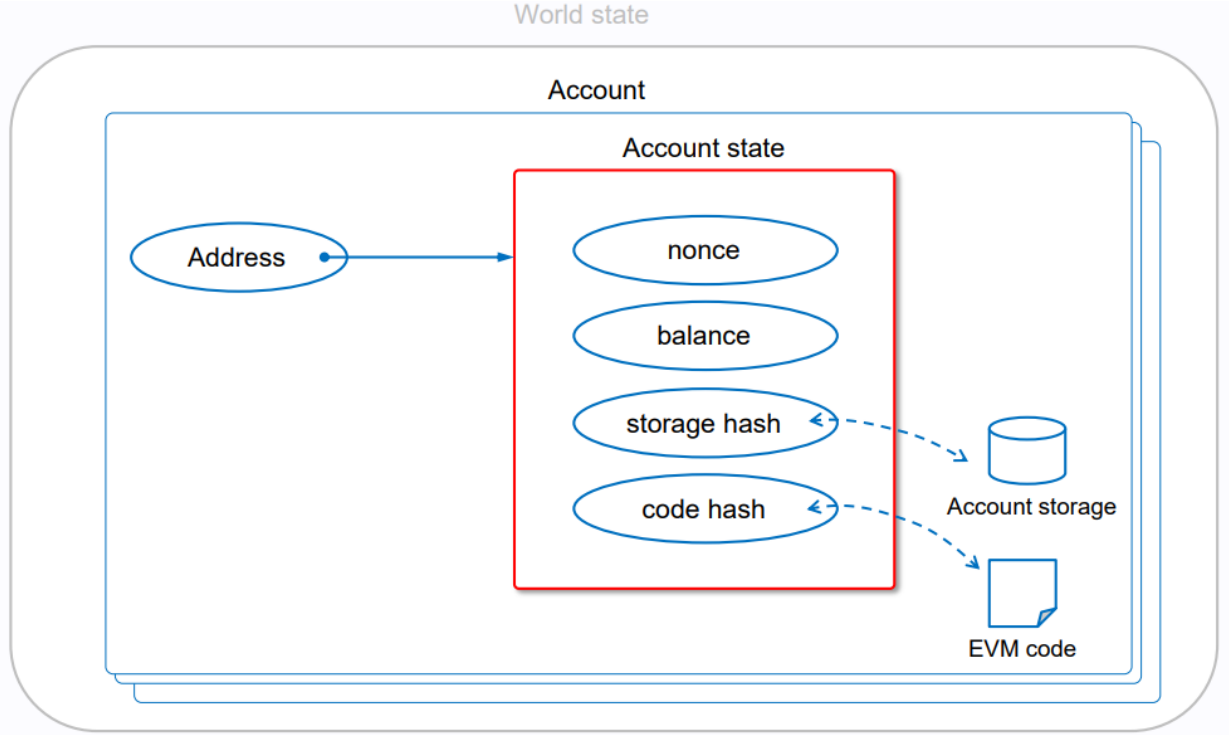
An account is an object in the world state.

Conceptual view



An account is a mapping between address and account state.

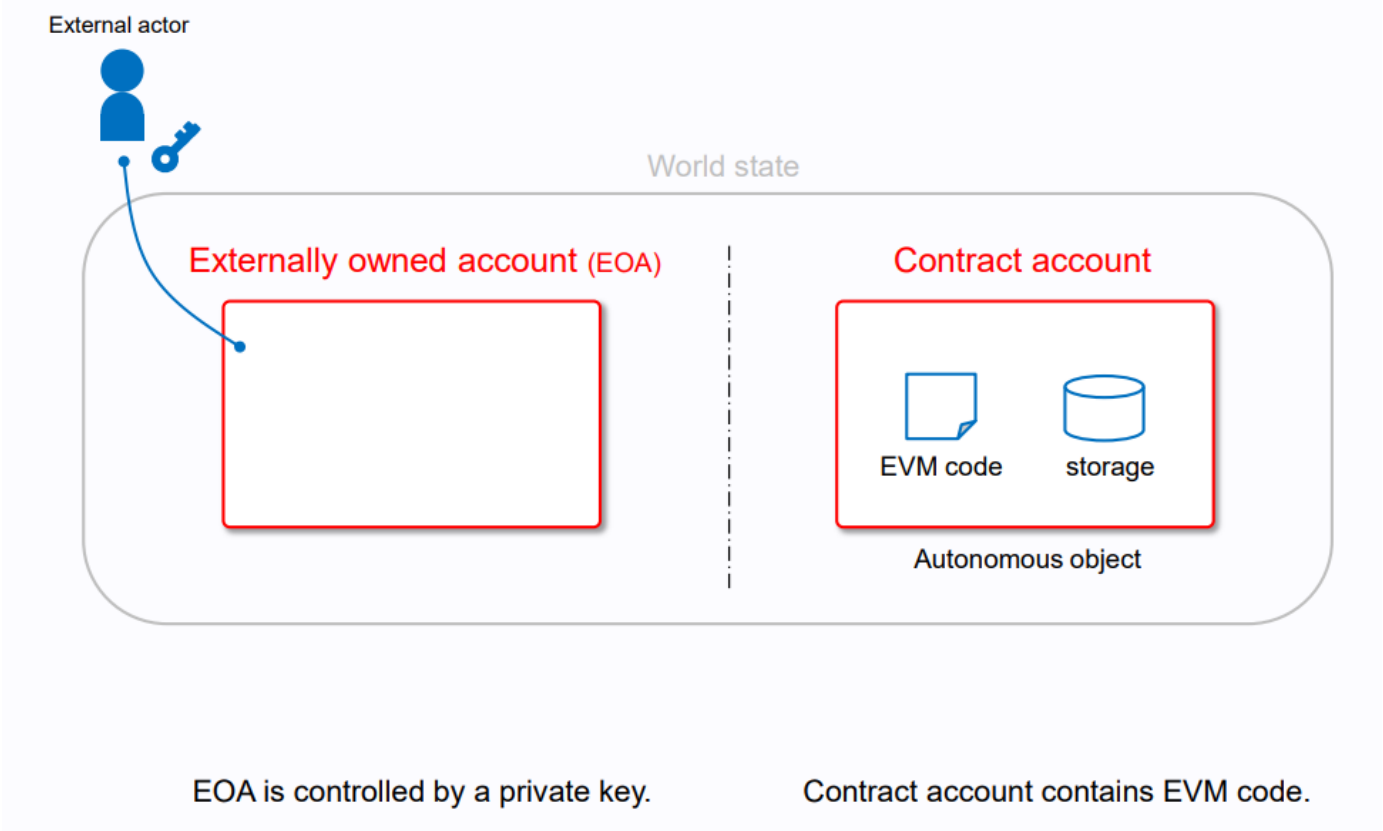
Account State



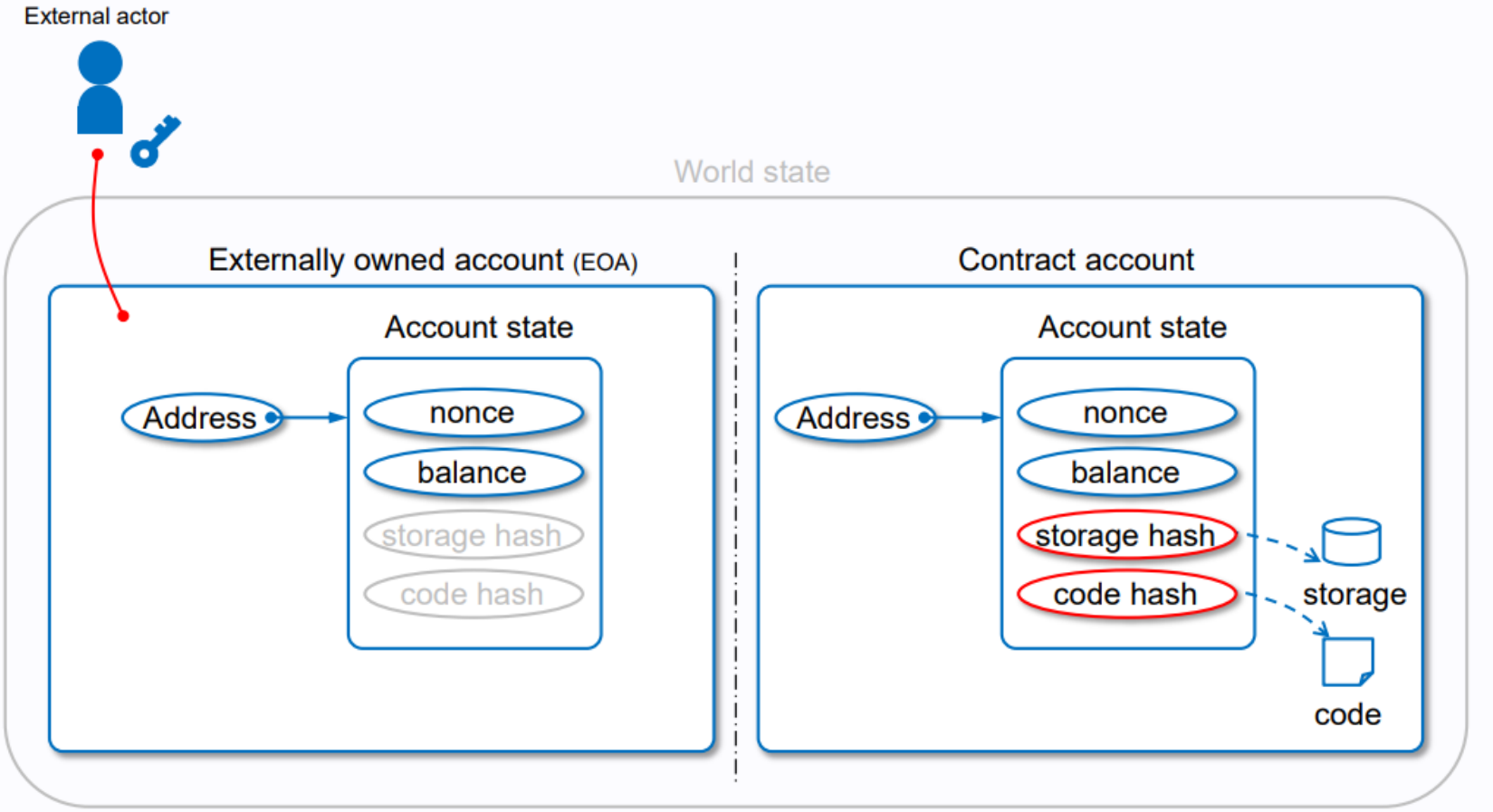
An account state could contain EVM code and storage.

Only applicable for contract accounts!

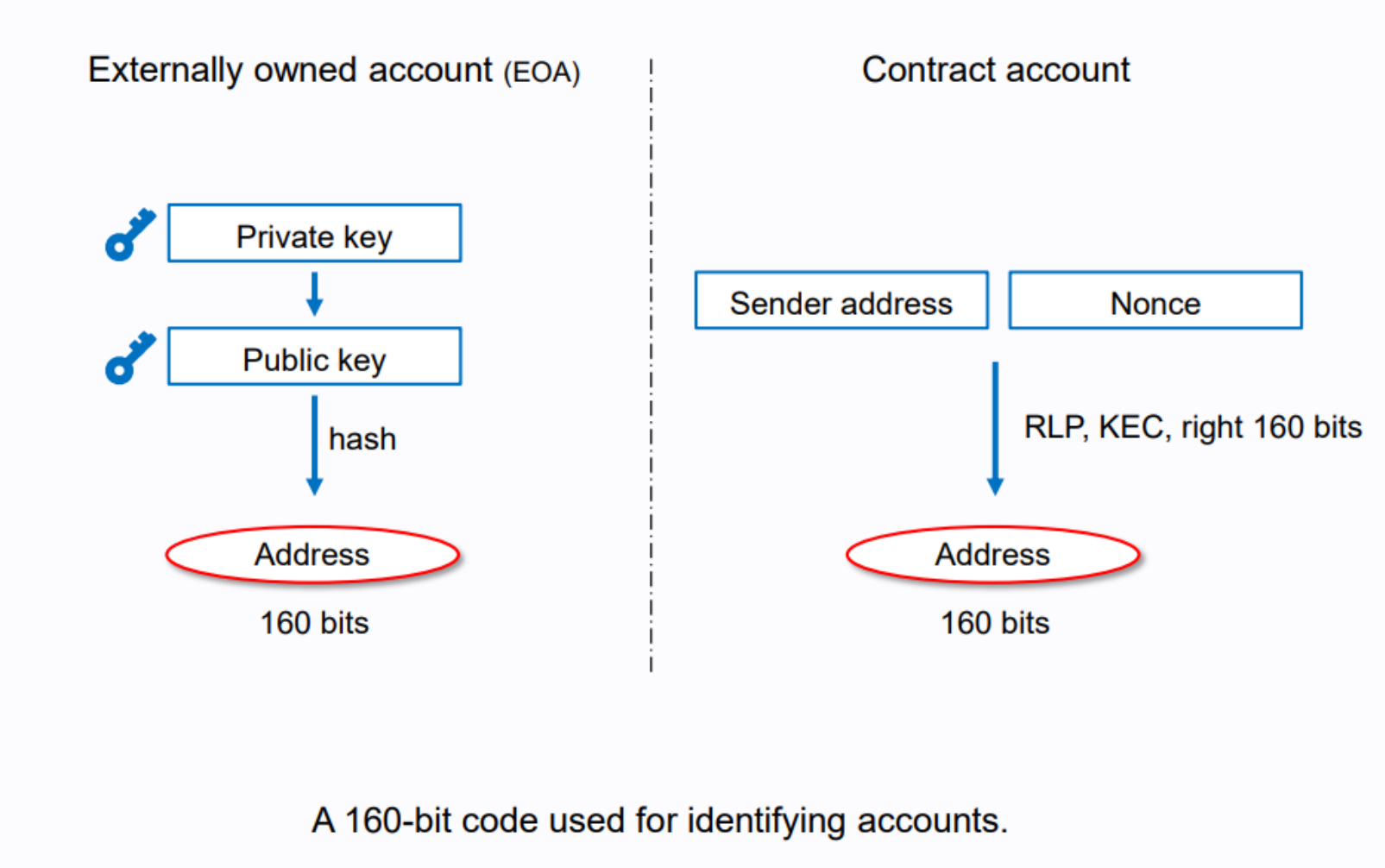
Accounts Revisited



Accounts: EOA vs. Contract

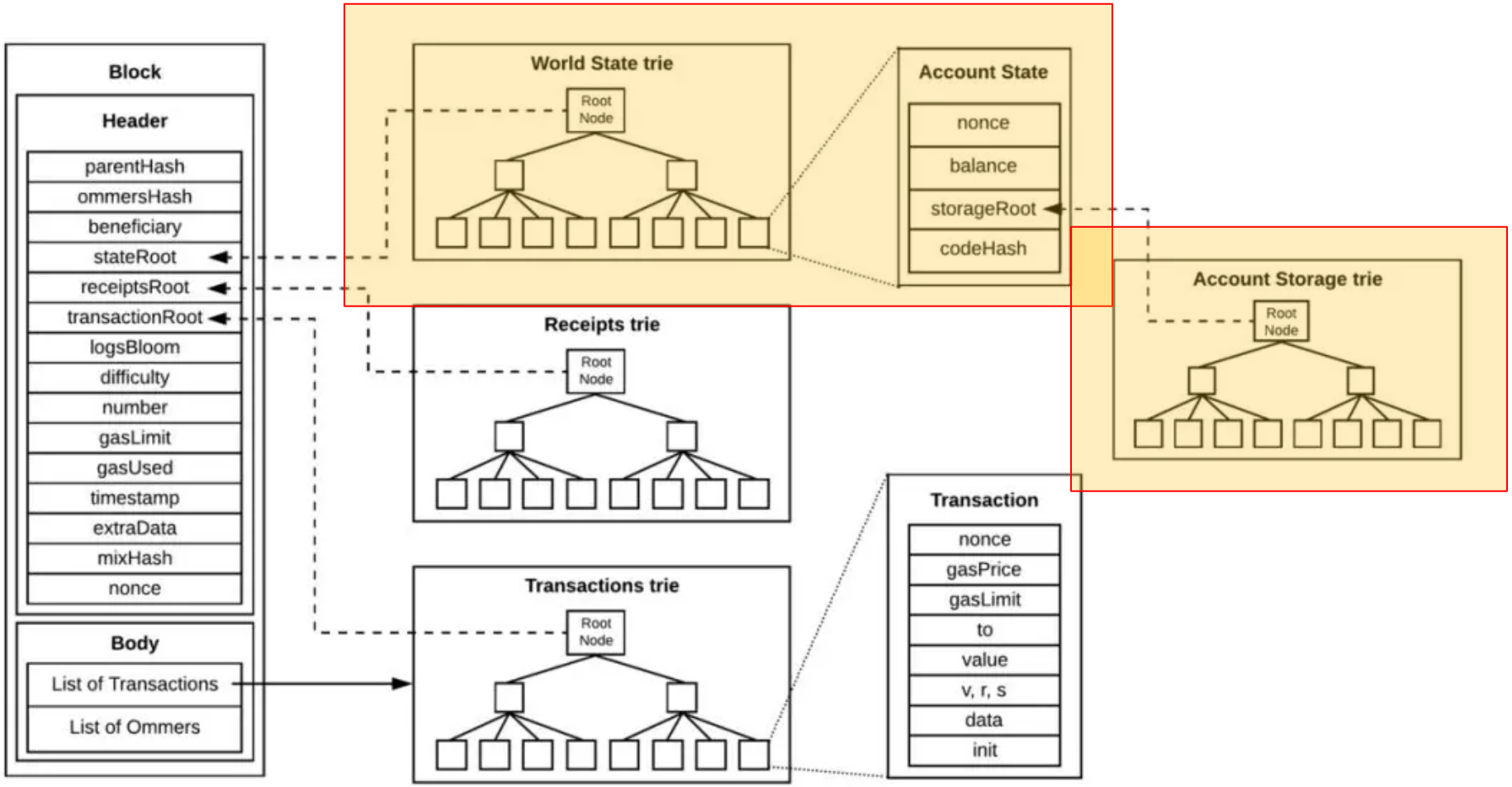


Address of Accounts



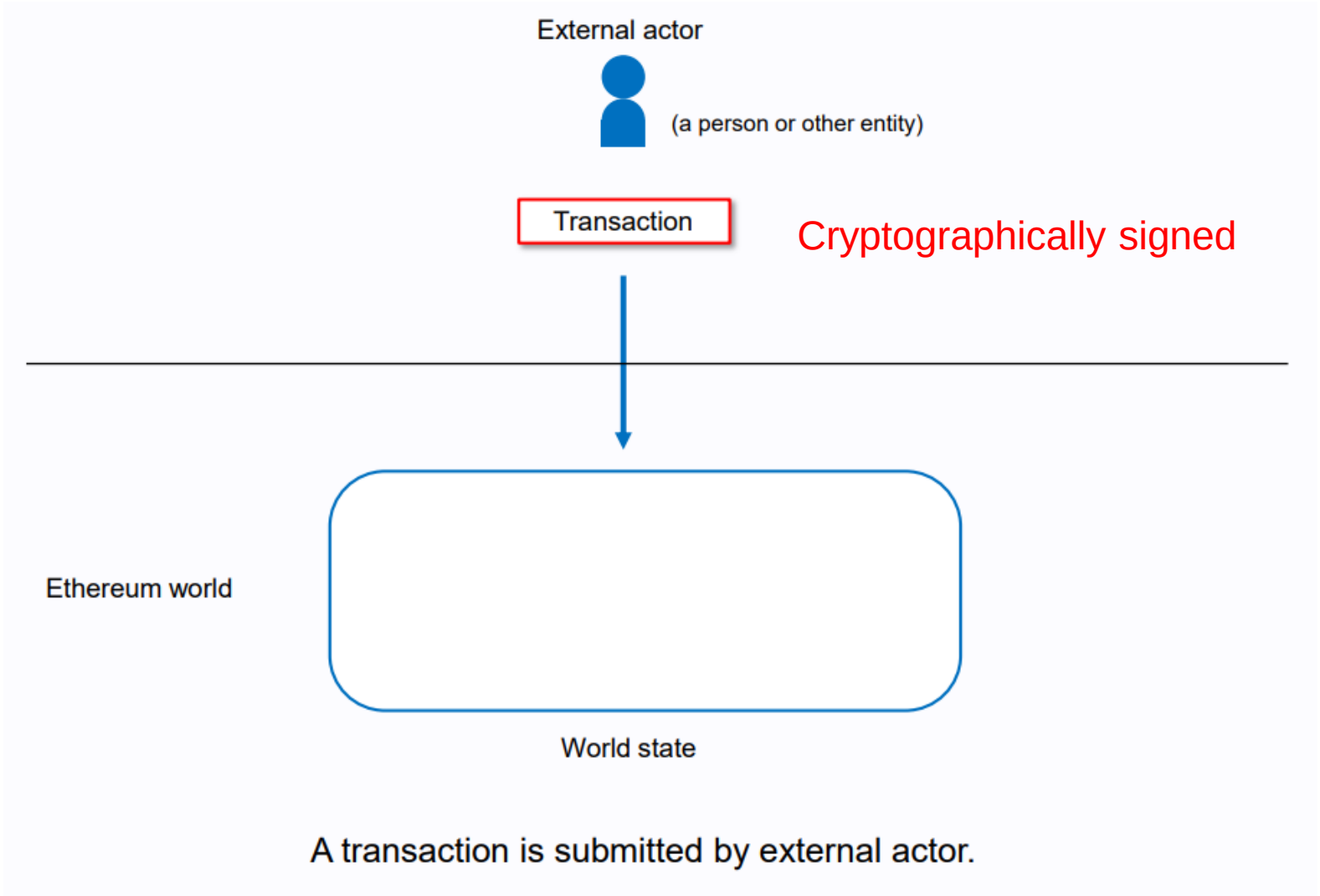
A 160-bit code used for identifying accounts.

Summary of Tries Revisited

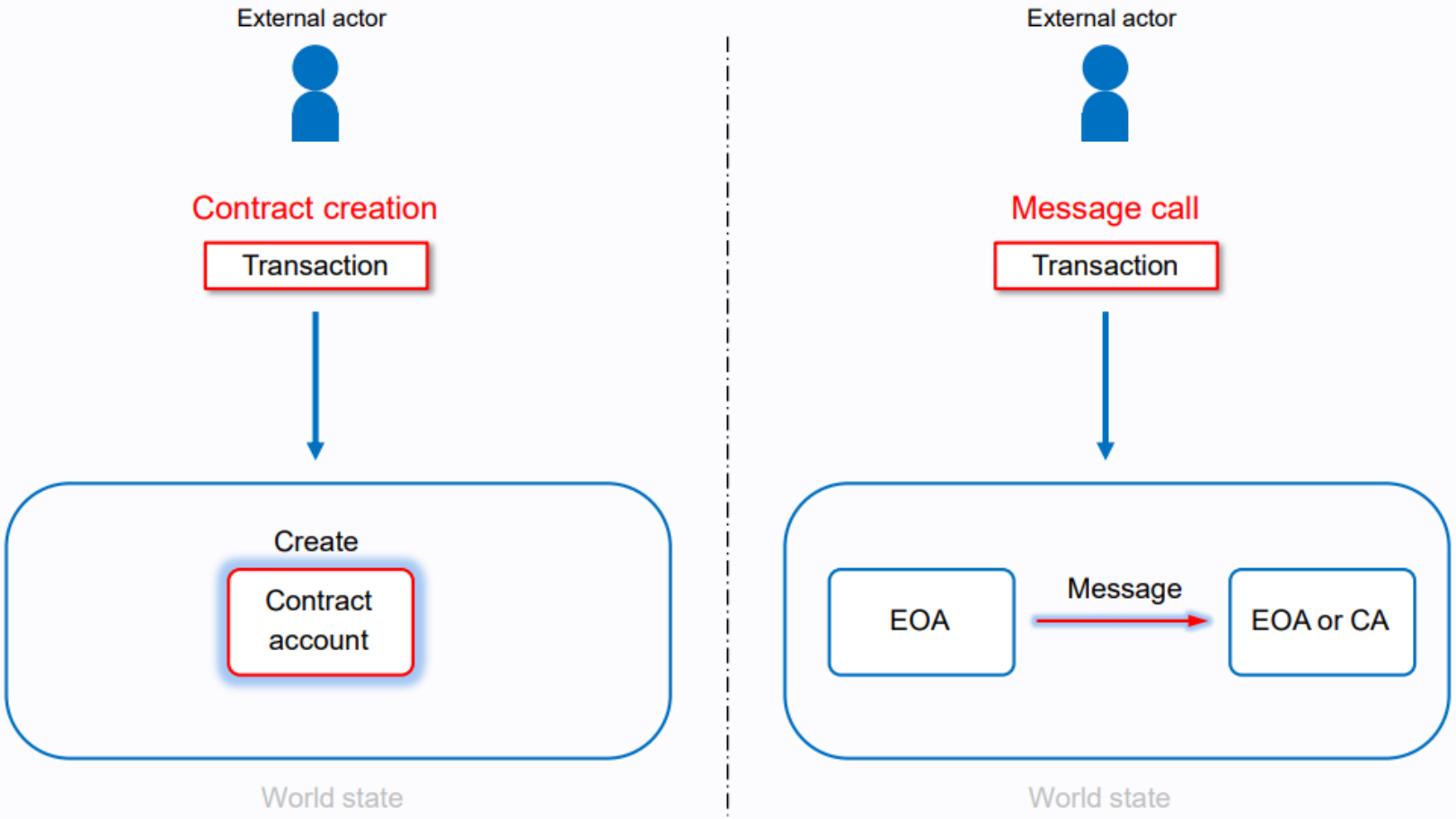


What might be a good data structure for Account Storage trie?

Transactions

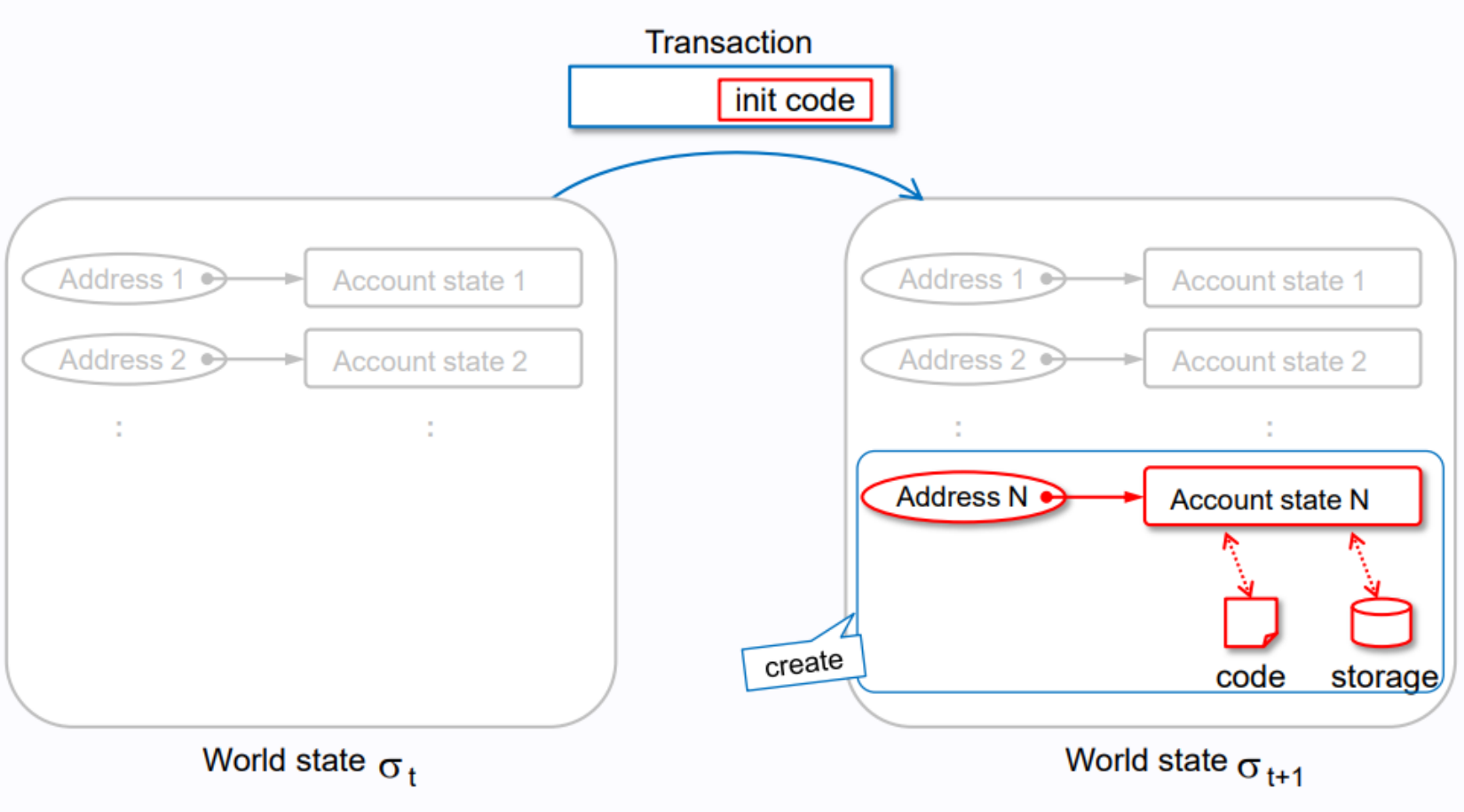


Special Transactions for Smart Contracts



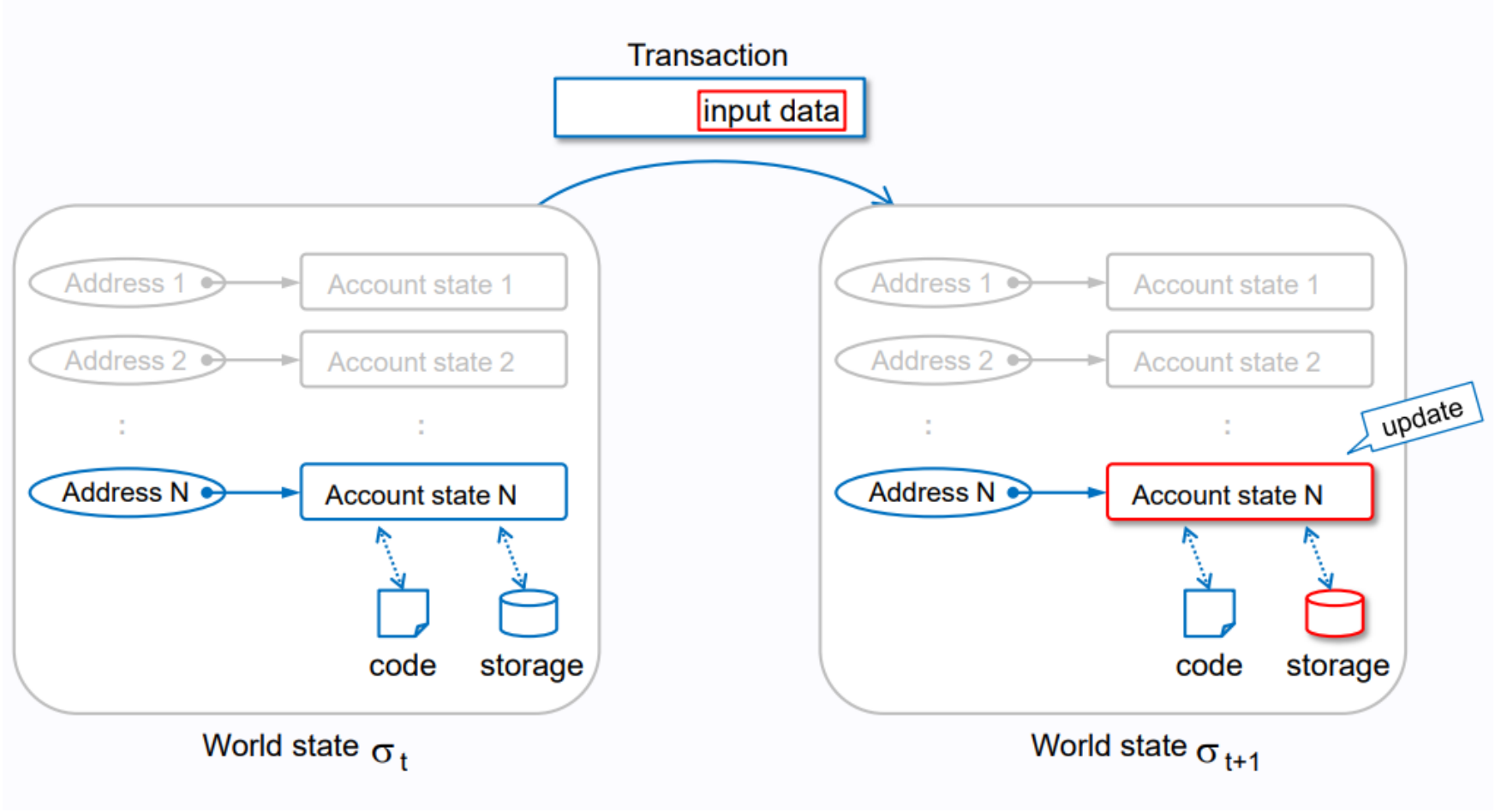
There are two practical types of transaction, contract creation and message call.

Contract Creation



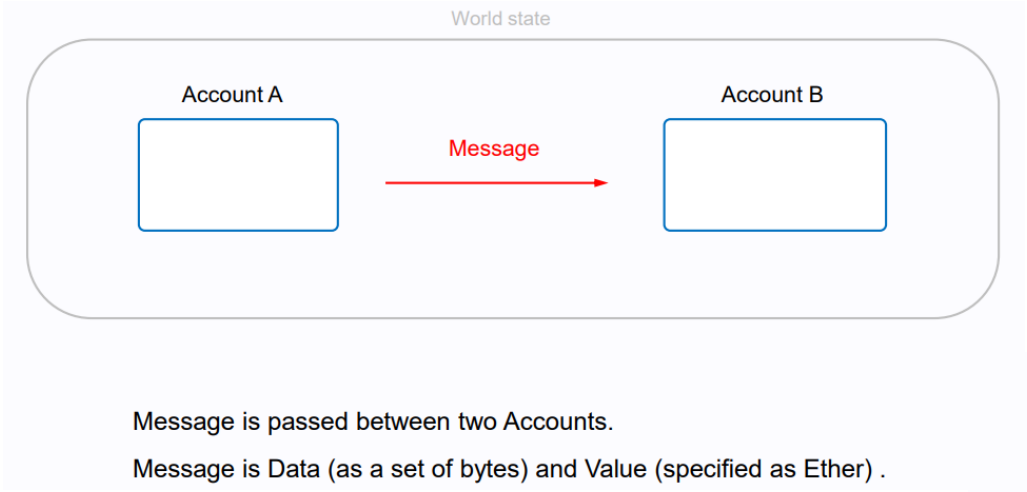
Creating a computer/server on the internet

Message Call to Invoke Contract Code



Running a computer on the internet to update the world state

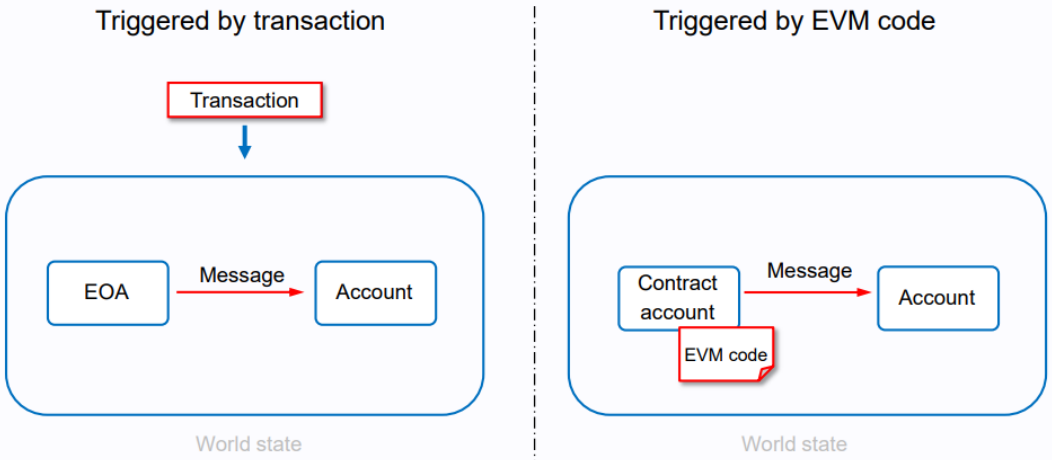
Messages



Message is passed between two Accounts.

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

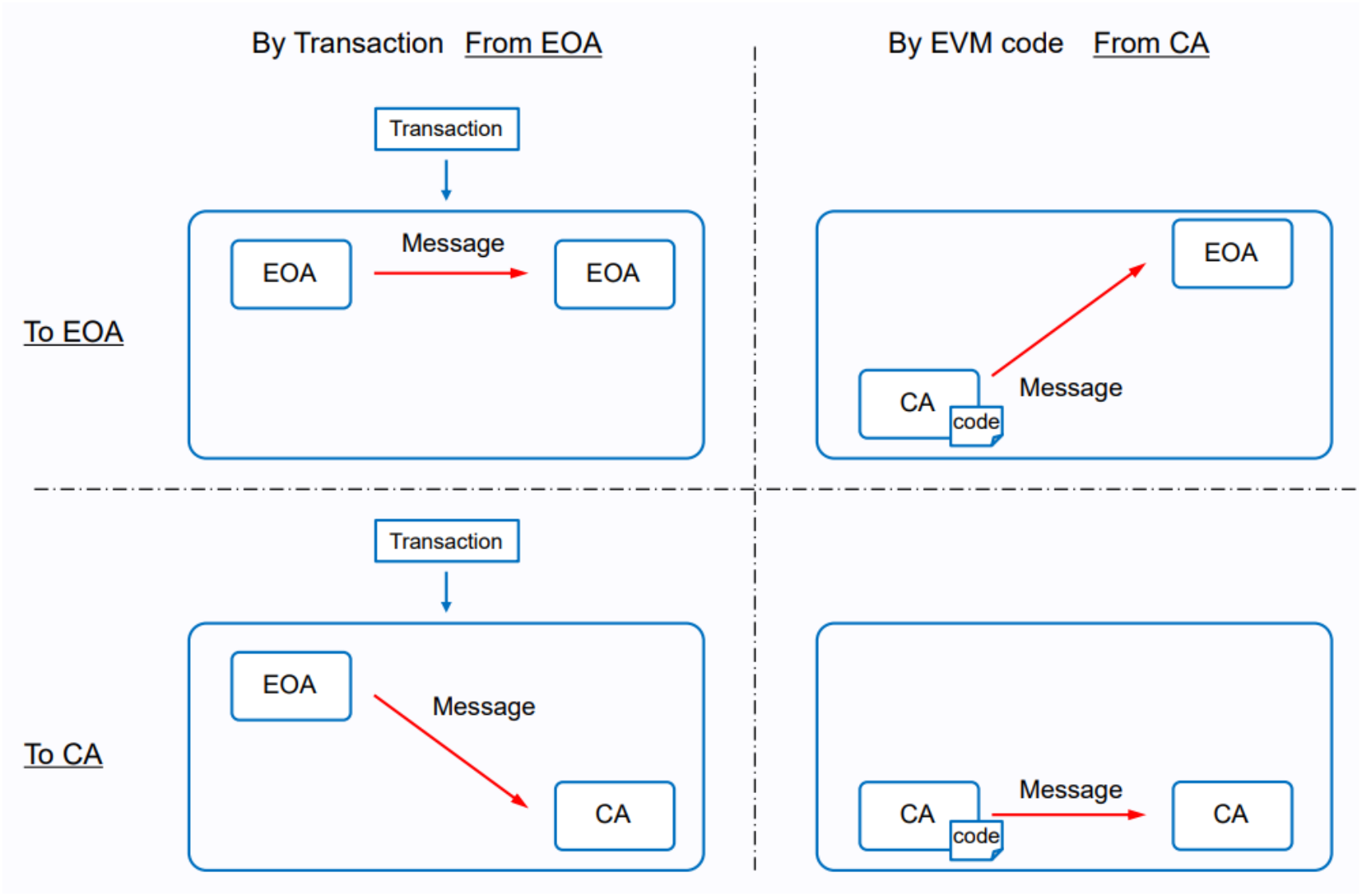
The simplest one is money transfer transaction



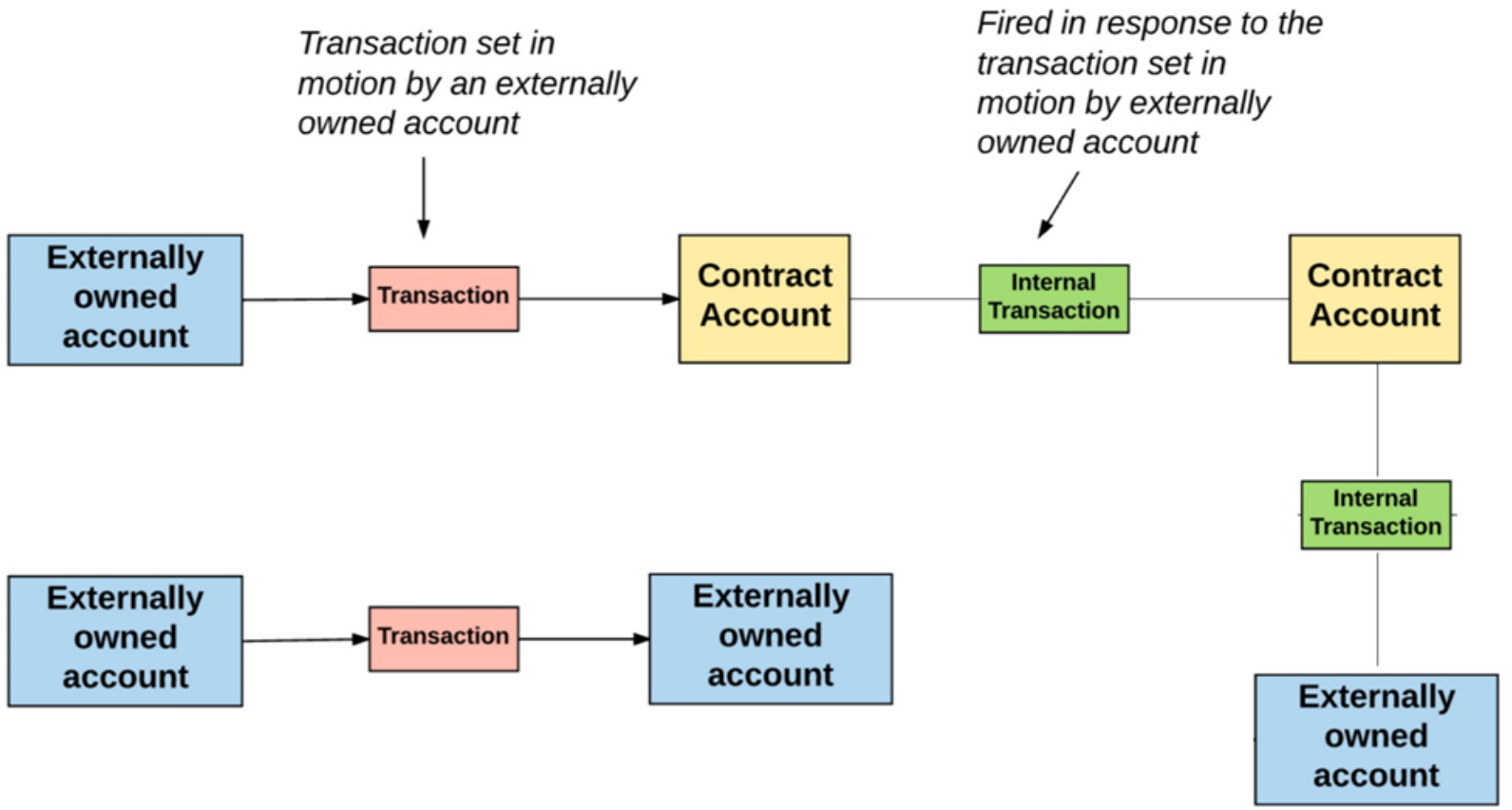
Transaction triggers an associated message.

EVM can also send a message.

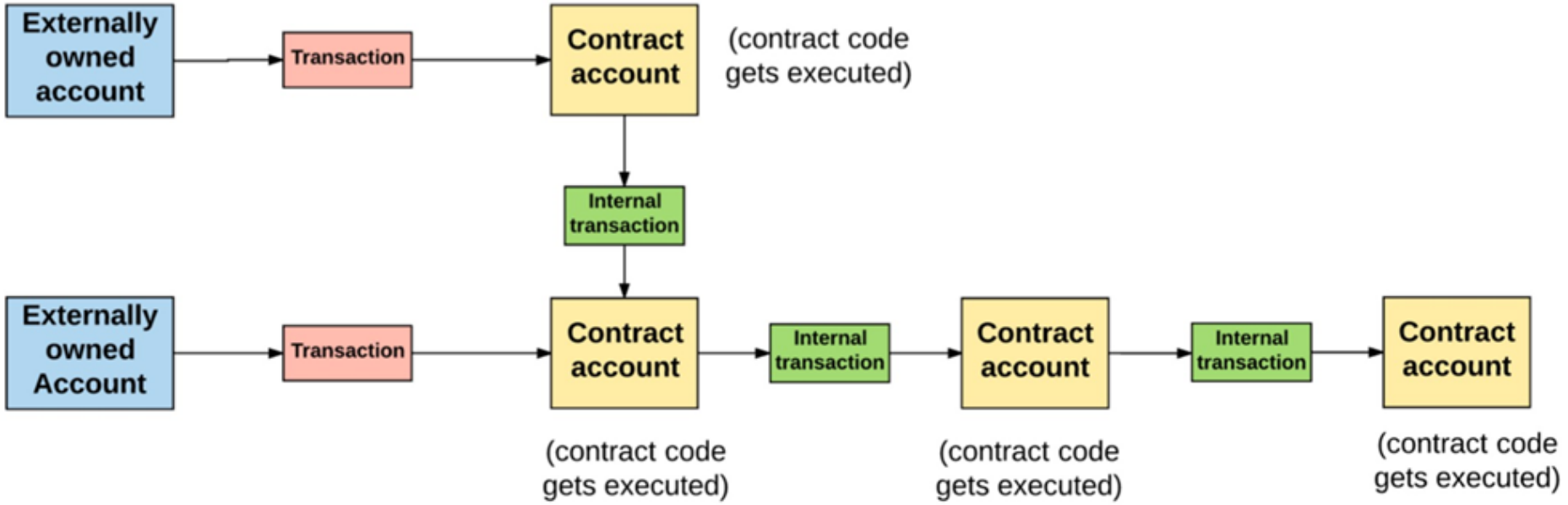
Message Calling



Message Calling Chain

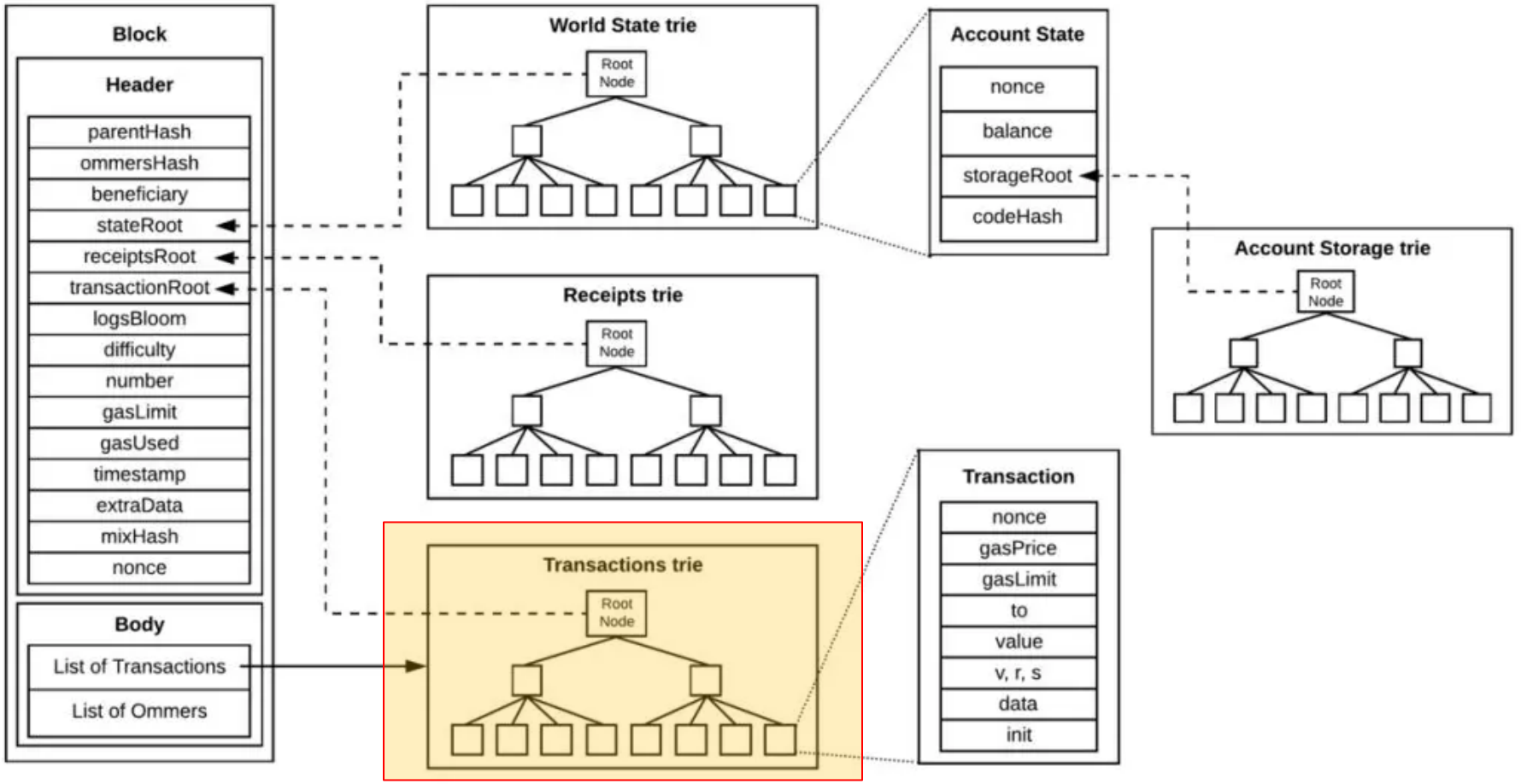


Another Calling Chain Example



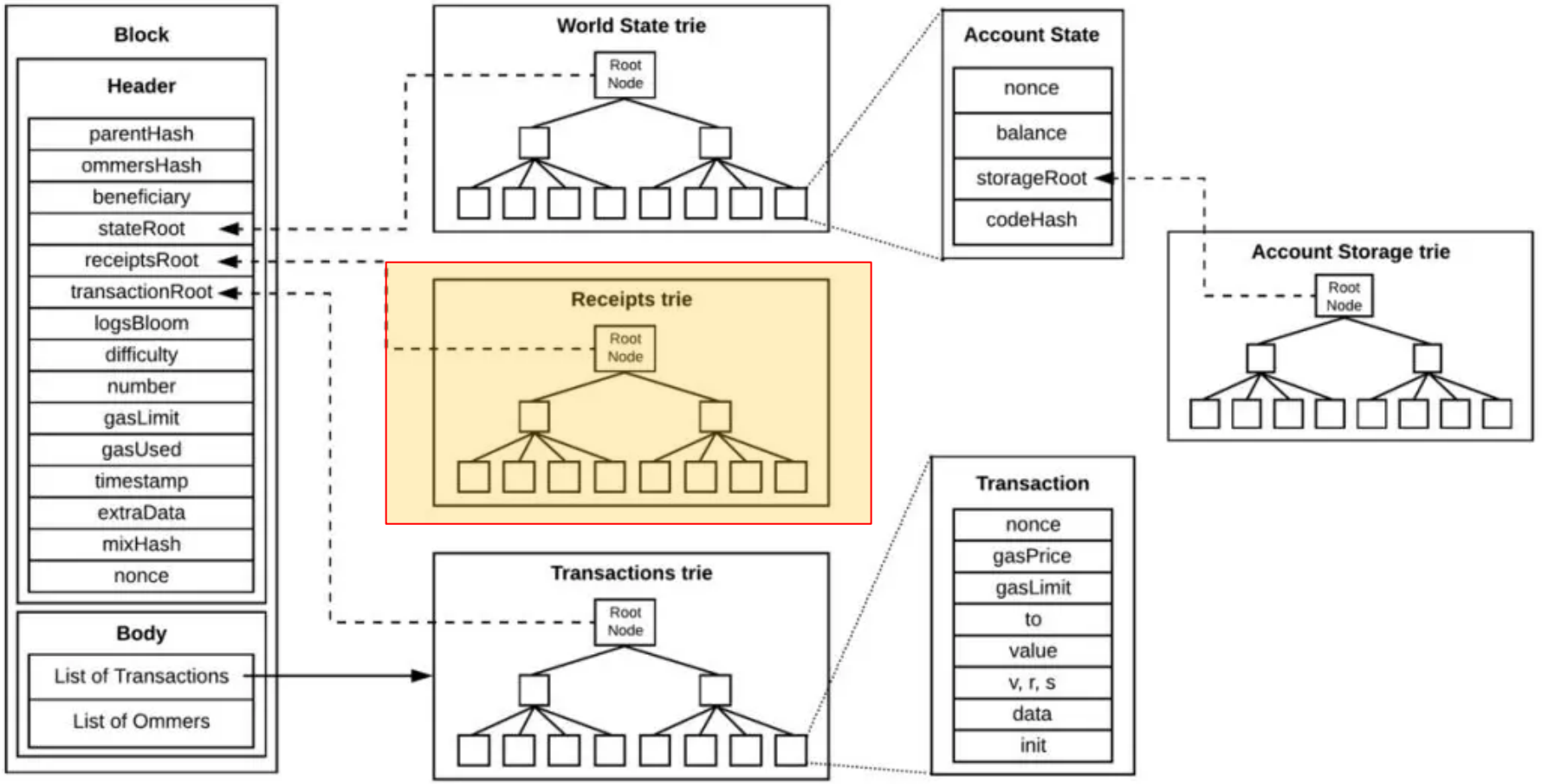
Can you think an application of this design? – Multisig?

Summary of Tries Revisited



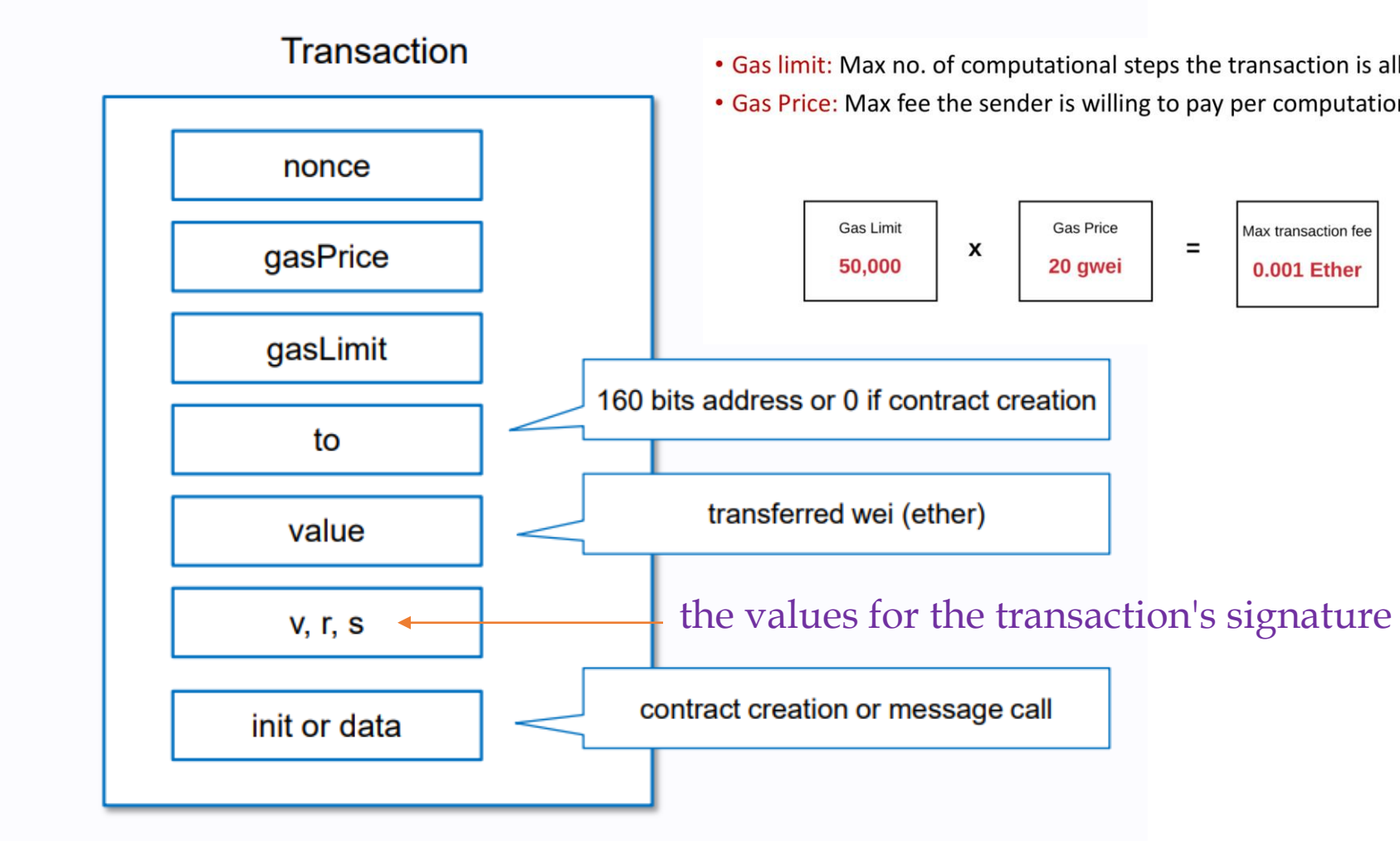
Note that transaction tries independent across different blocks.

Summary of Tries Revisited



Note that transaction tries independent across different blocks.

Fields of Transactions



Receipt Trie

- Why do we need receipt trie?
  - Will state trie be sufficient?
  - Will state trie + transaction trie be sufficient?
- Transaction receipt tries record transaction outcome
  - [details in section 4.4.1 of the yellow paper]: http://gavwood.com/paper.pdf
  - ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER (EIP-150 REVISION)

Receipt Trie Parameters

- Transaction receipt tries record includes:
  - post-transaction state,
  - the cumulative gas used,
  - the set of logs created through execution of the transaction, and
  - the Bloom filter composed from information in those logs
- Transaction receipt tries always accompany transaction tries
  - Therefore independent across different blocks
  - Useful to collectively handle frequent query cases conveniently

Quiz: Which Trie to Use?

- Different tries are used under different scenarios:
  - What is the current balance of a given account? (S)
  - Does a given account exist? (S)
  - Given a transaction, figure out whether it has been included in a particular block? (T)
  - Tell me all instances of an event of type X (eg. a crowdfunding contract reaching its goal) emitted by a particular address in the past 30 days (R: through the set of logs stored in receipt tries)