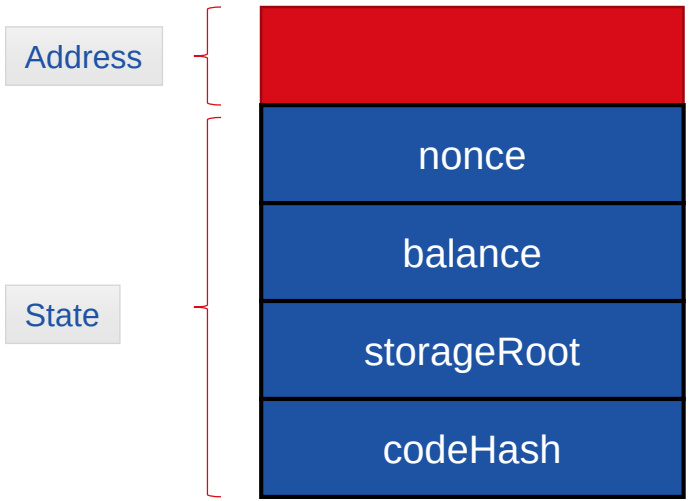
WeBank

以太坊基础知识

2019年7月

FISCO金链盟

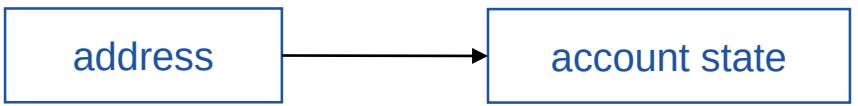
账户数据结构



- Address: 20-byte（160bit），从公钥计算得来；
- nonce：如果是外部账户，表示从这个账户发出的交易个数；如果是合约账户，表示这个账户创建的合约个数；
- balance：账户余额，以“卫”为单位，每个以太是10的18次方个“卫”；
- StorageRoot：该账户相关的存储的Merkle Patricia Tree的根hash；
- codeHash：如果是外部账户，它是空字符串的hash，如果是合约账户表示EVM代码的hash

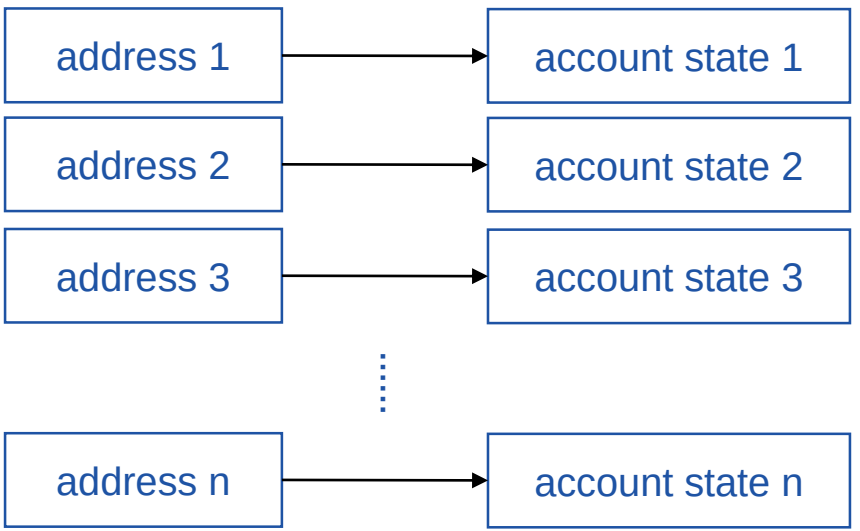
账户 (Account)

账户=地址+状态



World State=所有账户状态

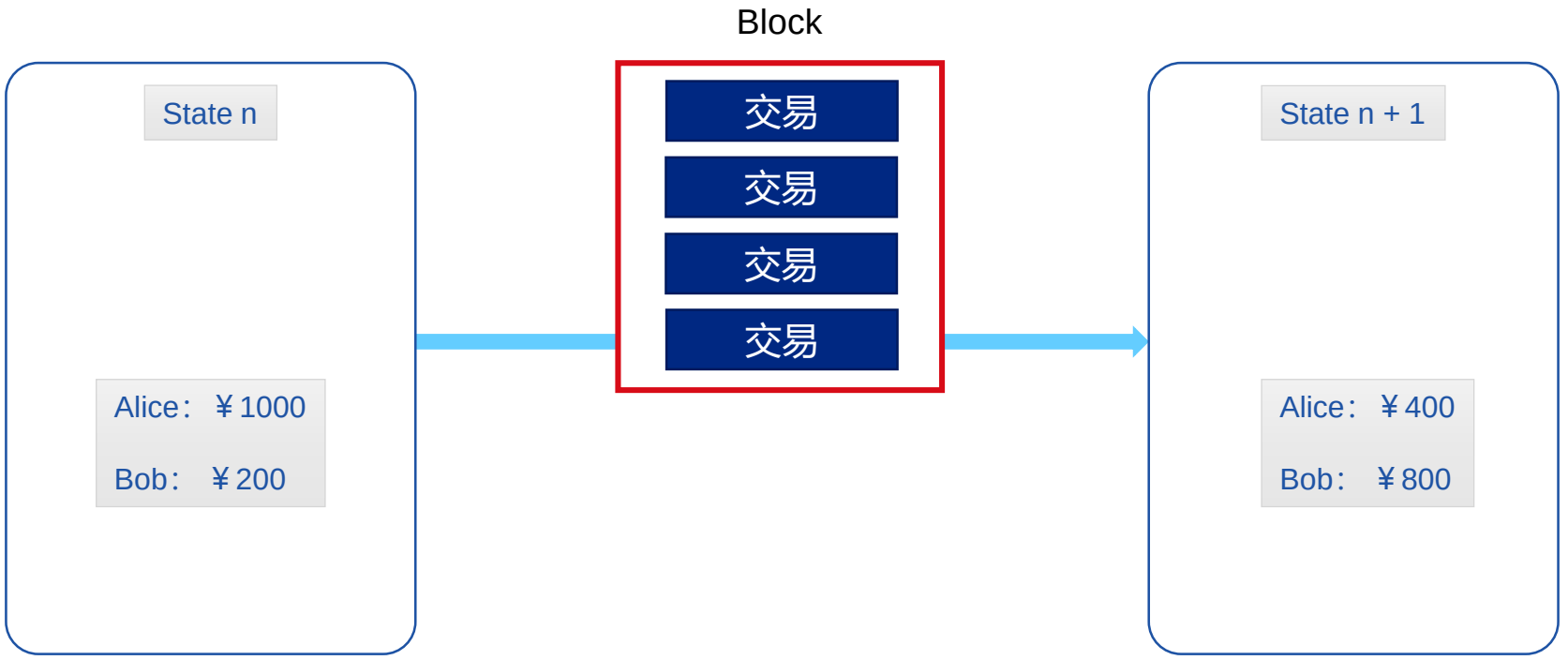
- 账户地址到账户状态的映射表，以Merkel Patricia Tree的结构存储。



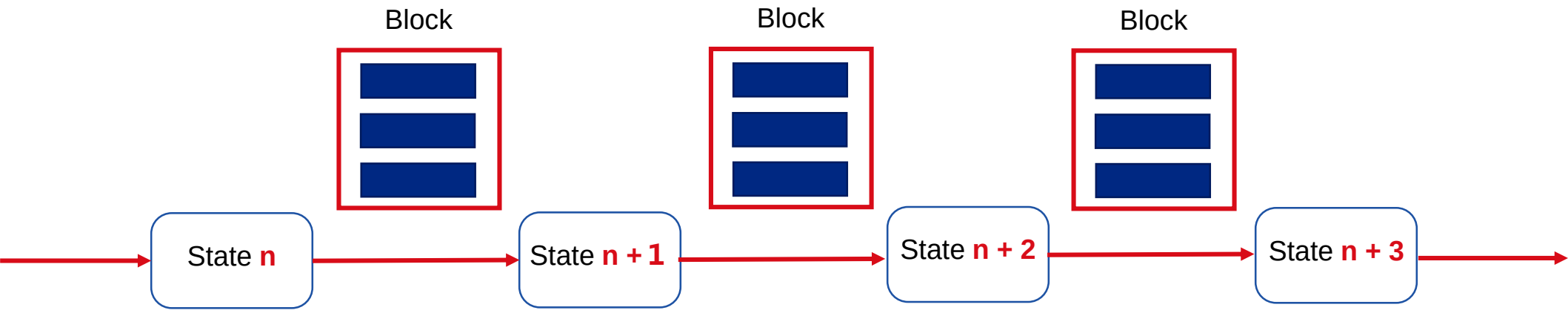
交易改变账户状态


State n
Alice：￥1000
Bob：￥200
Alice转账给
Bob ￥600
State n + 1
Alice：￥400
Bob：￥800

一个块一次共识


Block
State n
交易
交易
交易
交易
State n + 1
Alice：￥1000
Bob：￥200
Alice：￥400
Bob：￥800

区块链=状态链



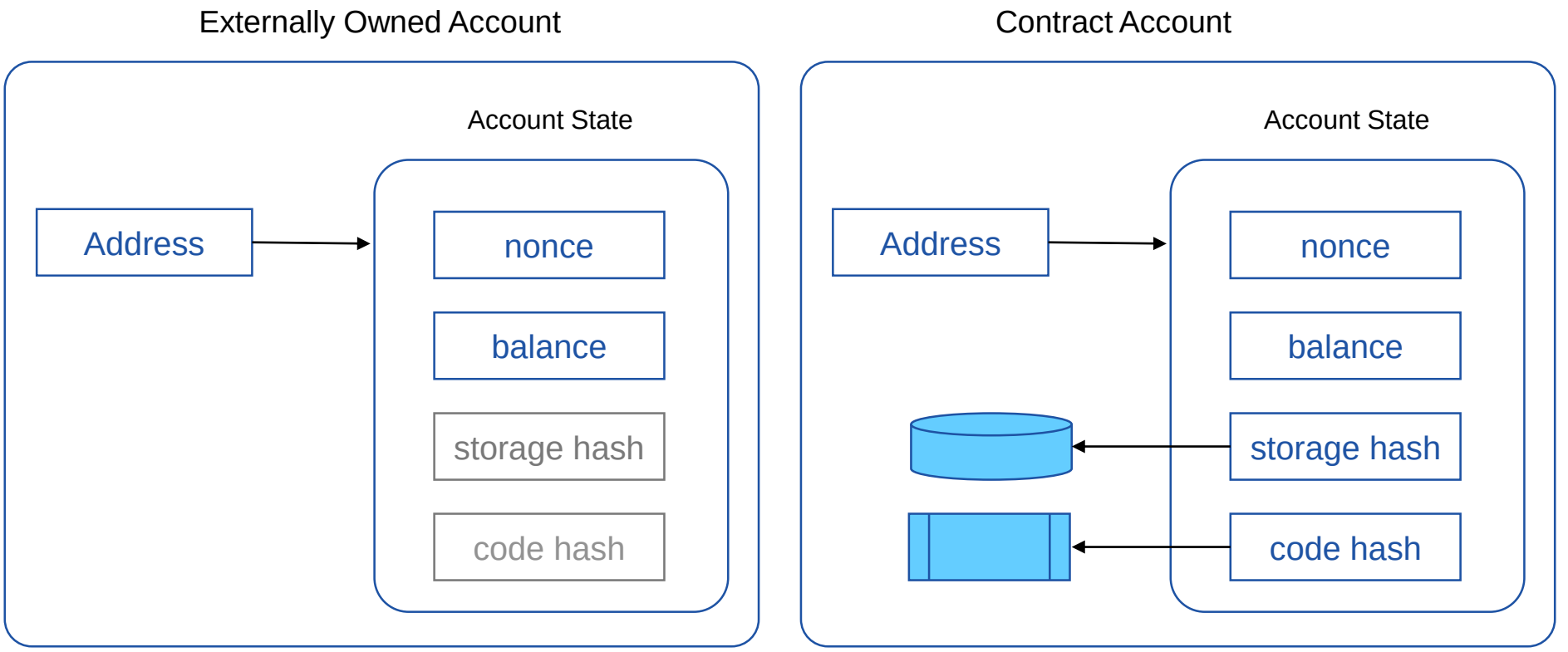
外部账户和合约账户

账户是以太坊的基础，以太坊账户分类：

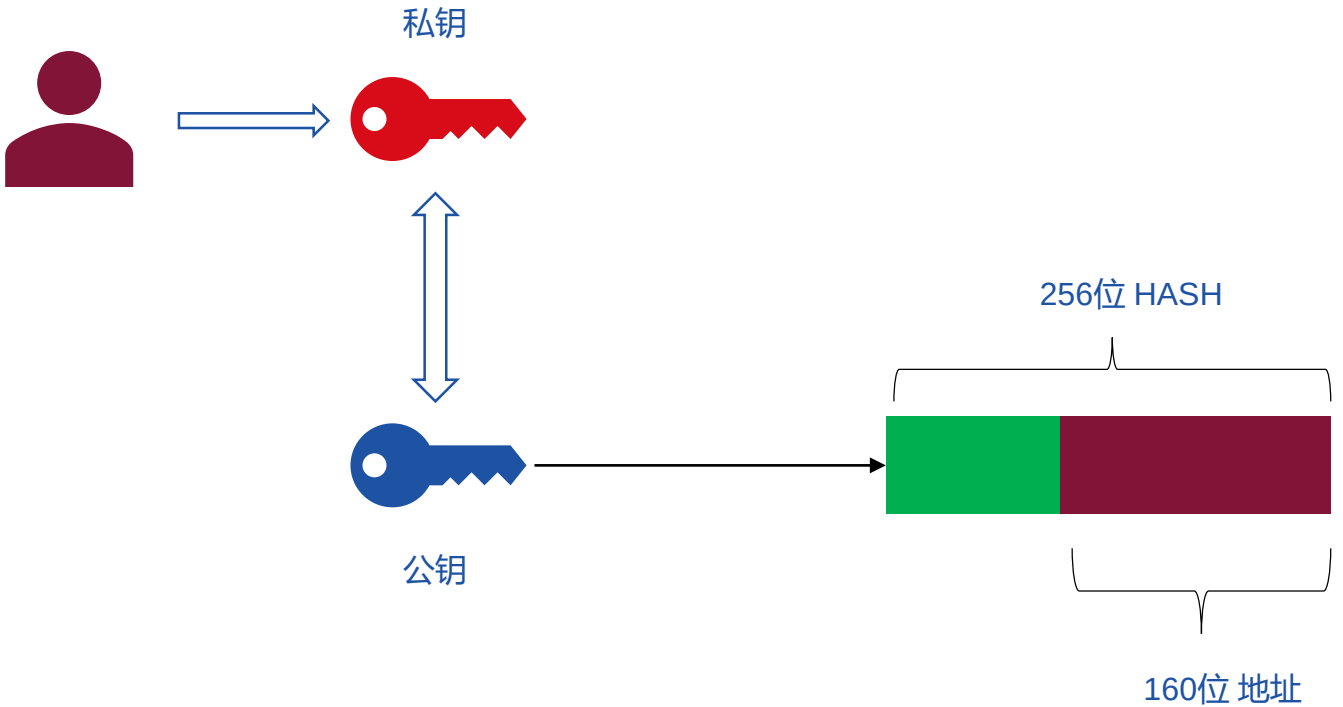
- **Externally Owned Accounts (EOAs)，外部账户，这类账户由私钥控制；**
- **Contract Accounts，合约账户，由合约代码控制，只能通过EOA执行。**

用户拥有私钥，私钥控制外部账户，用户间接控制外部账户；合约账户人们常说的“智能合约”指的是合约账户。

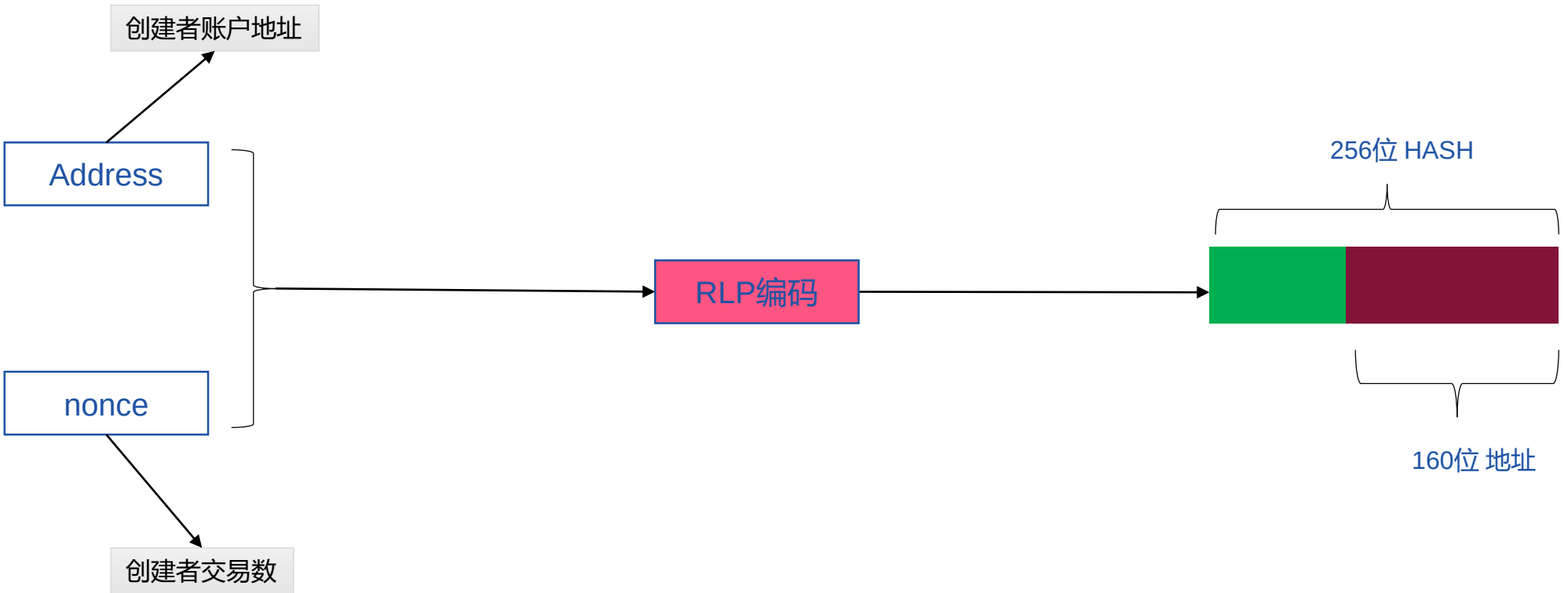
账户与合约的链上结构


Externally Owned Account
Account State
Address
nonce
balance
storage hash
code hash
Contract Account
Account State
Address
nonce
balance
storage hash
code hash

外部账户地址

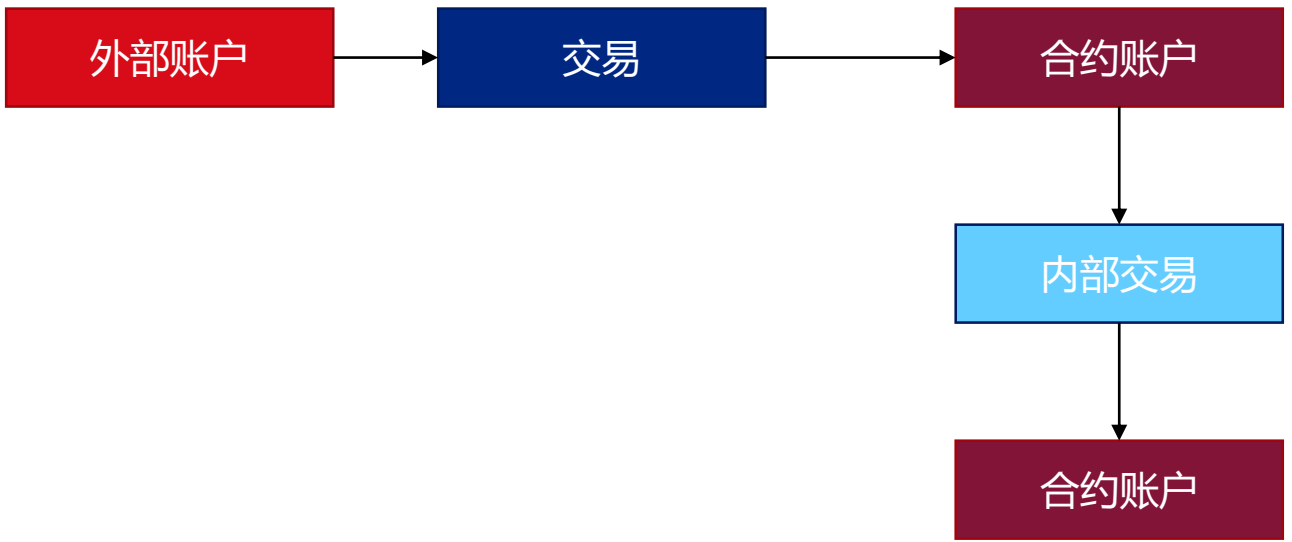


合约账户地址



合约账户的执行

- 合约账户只有接受到外部账户的执行指令时才会执行；
- 执行者需要支付一定的费用——称为gas——合约账户才会执行；
- 费用由节点收集

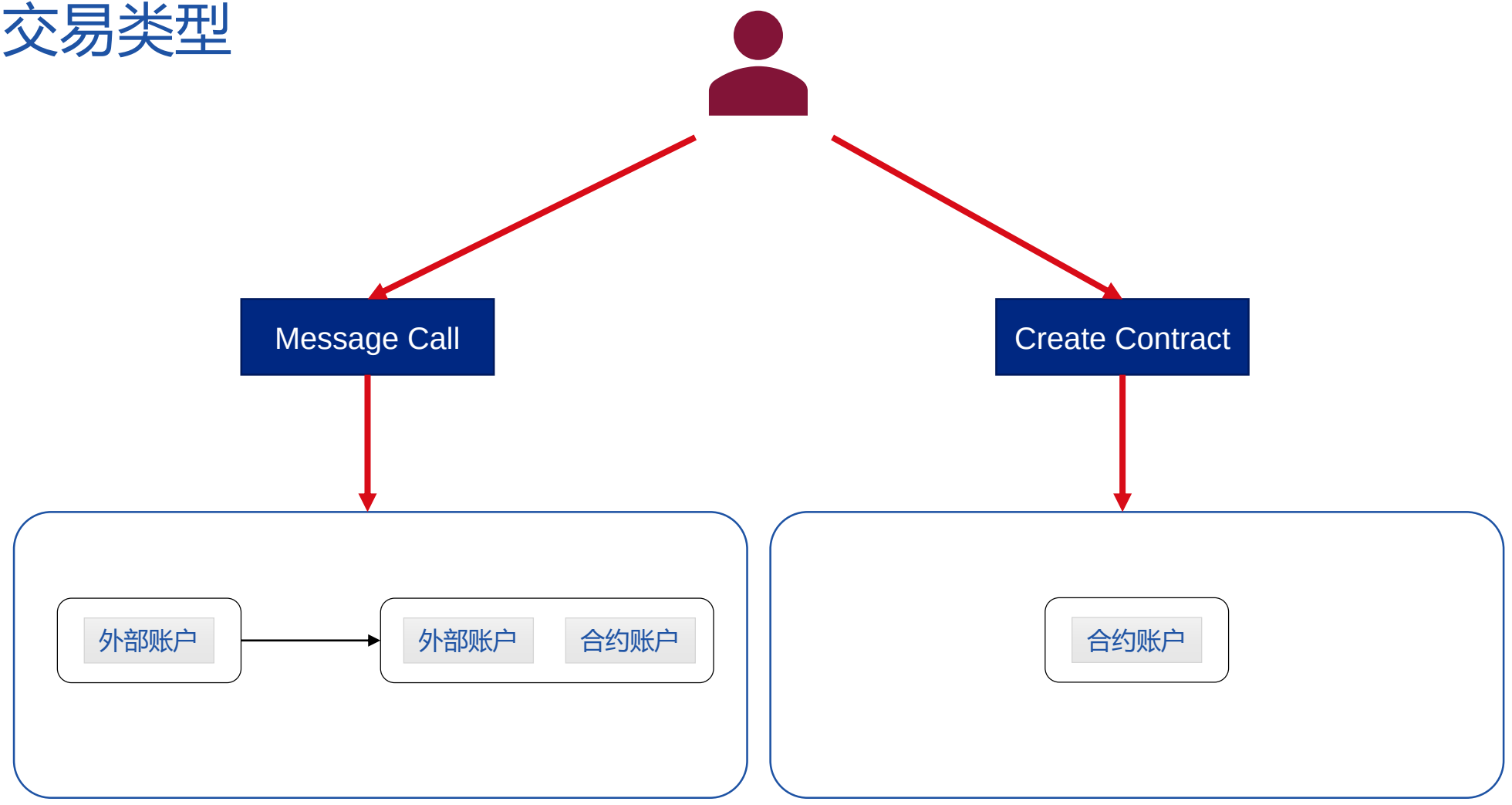


交易 (Transaction & Message Call)

交易

- 由外部账户发起；
- 节点收到后验证有效性；
- （共识）
- 在EVM中执行合约；
- 每执行一条指令都要消耗一定的gas；
- 改变账户状态；
- 产生回执（收据）；
- 日志

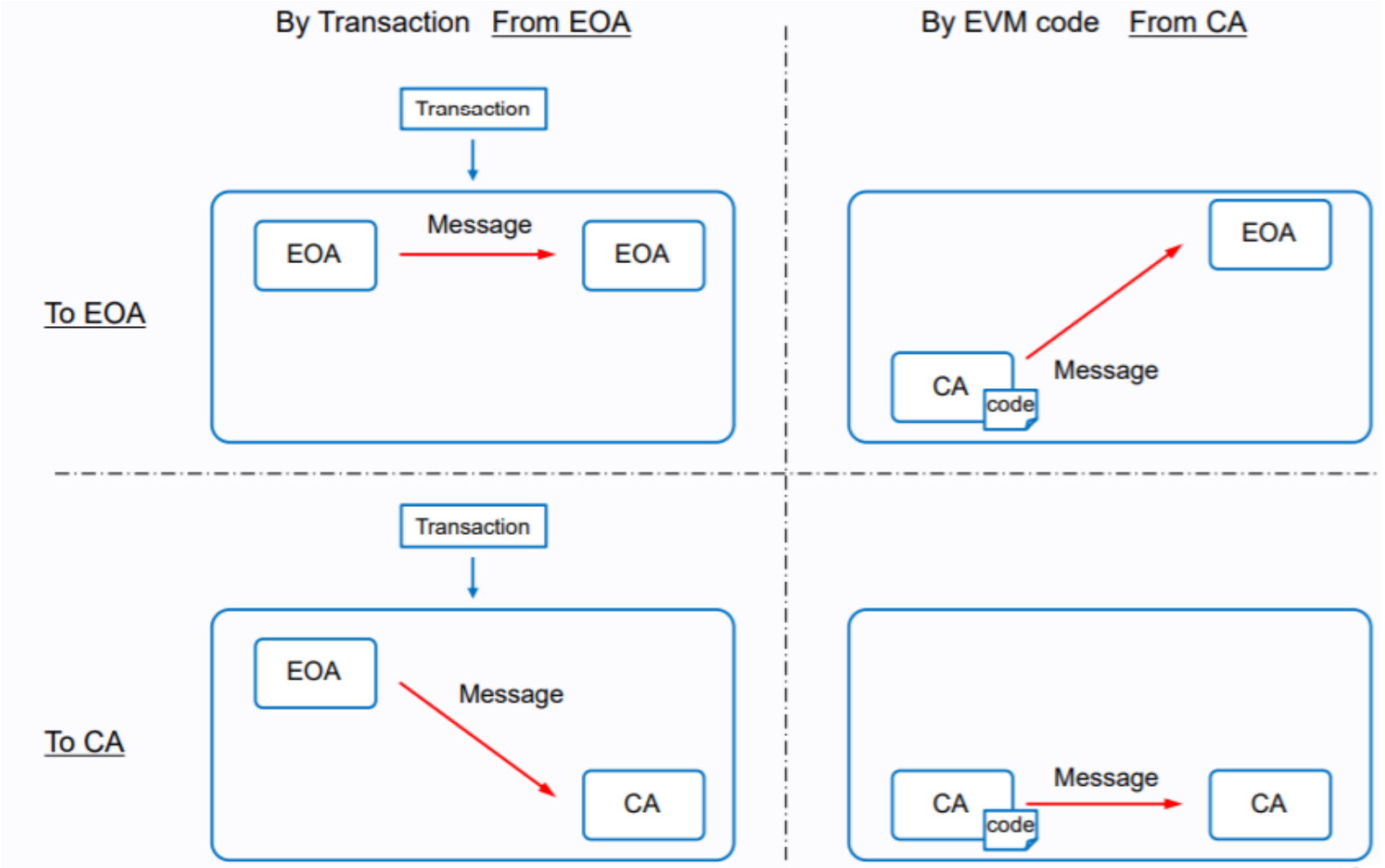
交易类型



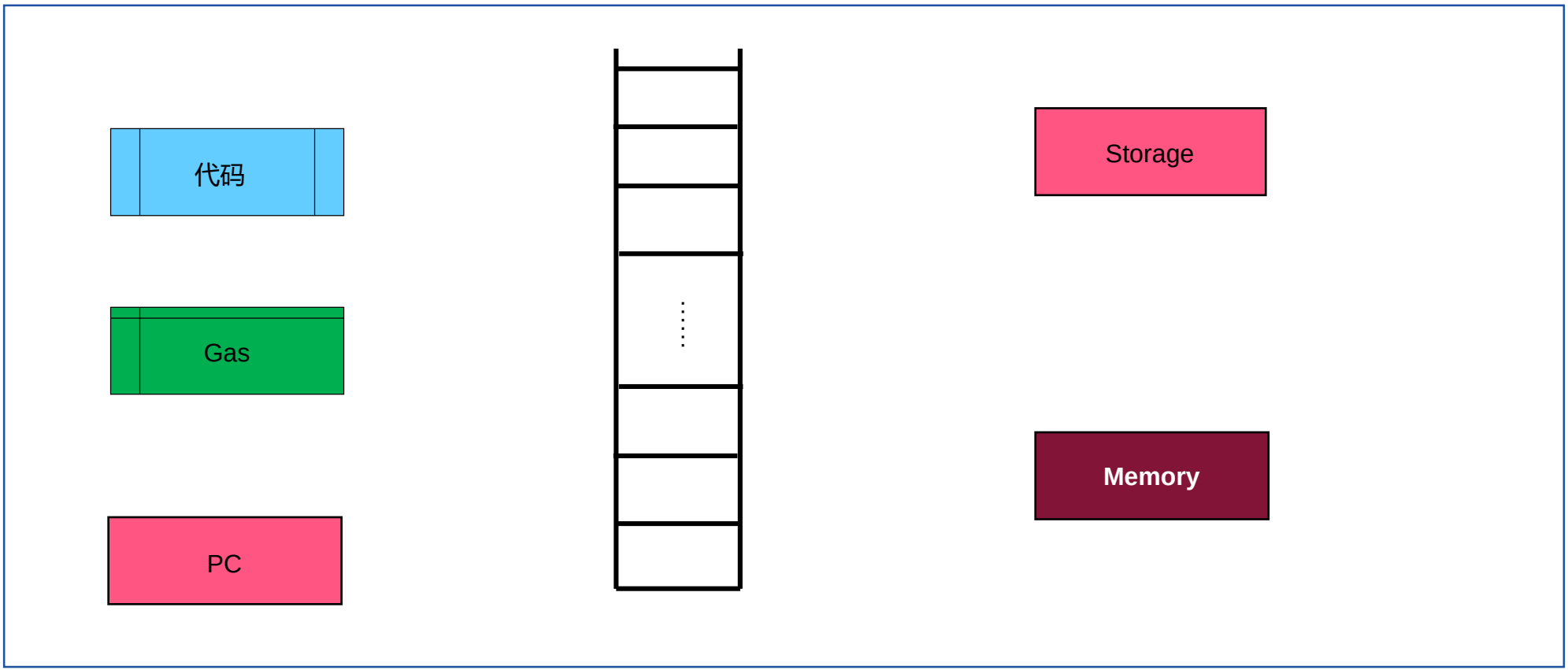
交易数据

- from：交易发送者；
- to：交易接收者，为空表示创建智能合约；
- value：转账数量；
- Gas Limit：允许消耗最大gas；
- GasPrice：gas价格；
- nonce：统一用户的最大交易标记；
- hash：以上内容生成的hash；
- r, s, v：交易签名的三部分；
- init or Data：合约代码或者Message Call Input data

4 types of message

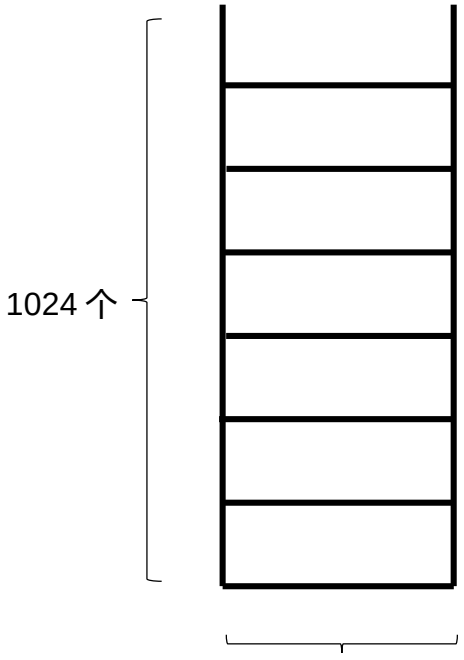


Ethereum Virtual Machine



Ethereum Virtual Machine

栈



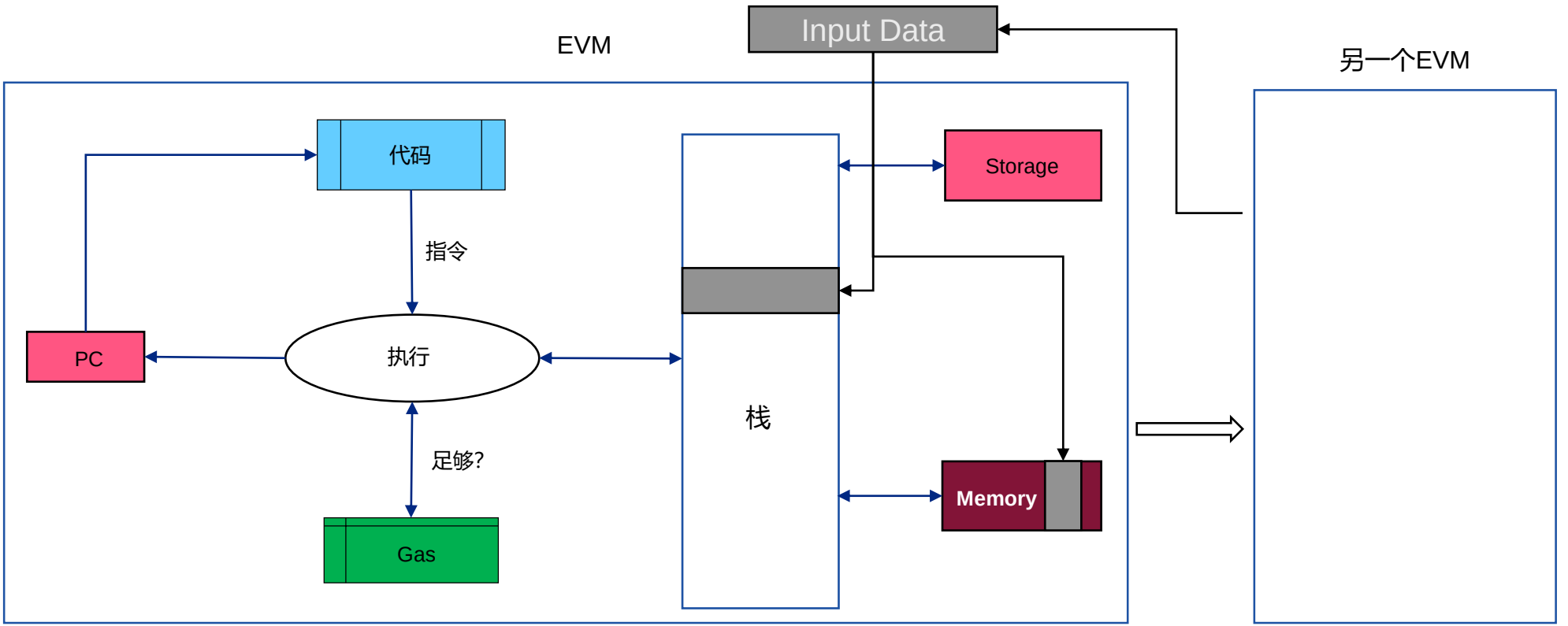
存储

Key	Value
0	0
1	18
2^{256}-1	0

内存

Value
4
95
……
……
……

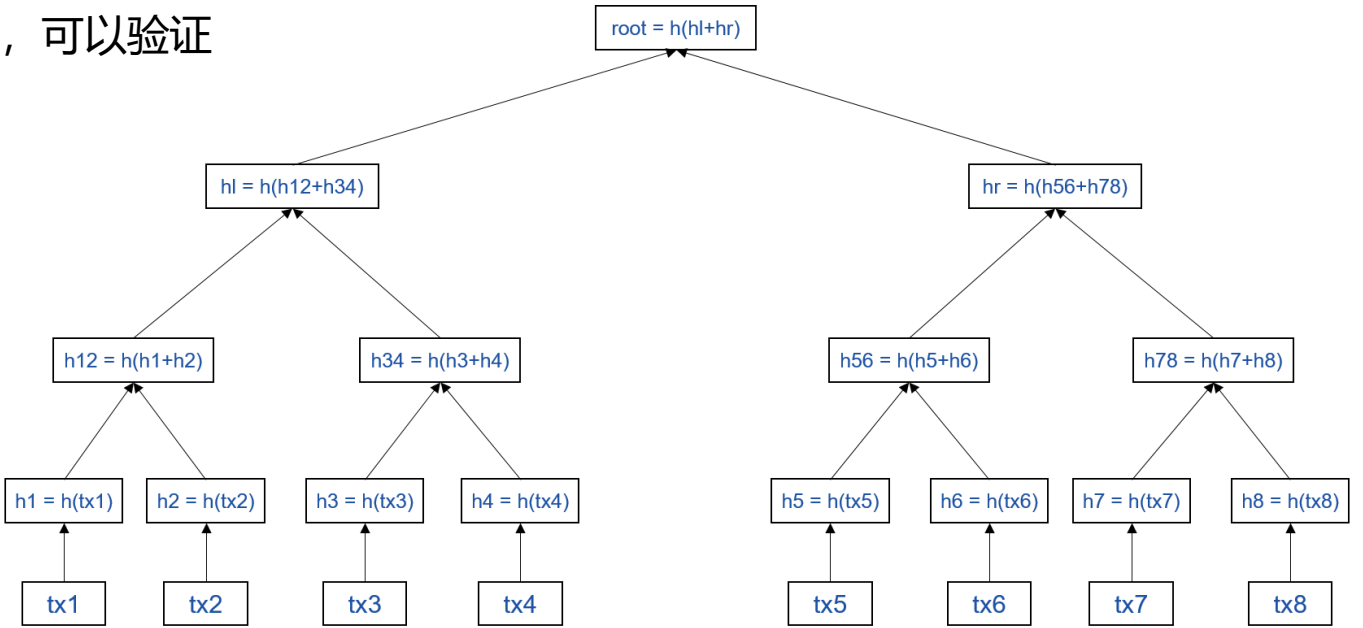
运行中的EVM



存储 (Storage)

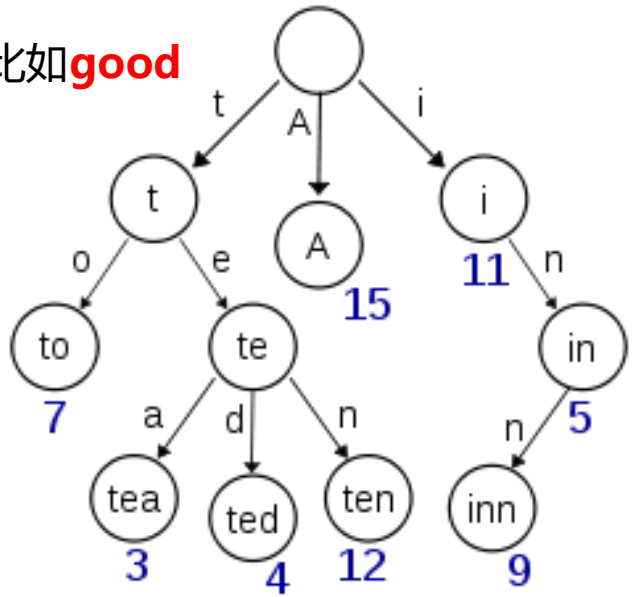
Merkle 树

- 很容易更新、添加后者删除树节点
- 不改变HASH就没办法改变树的任何部分
- 容易验证
- 树和数据可以从不同的节点同步，可以验证



Trie 树

- 树的最大深度就是key的最大长度
- key离得越近，value离得也越近
- 不是平衡树，如果没有相同的前缀需要存储更多的节点，比如**good**

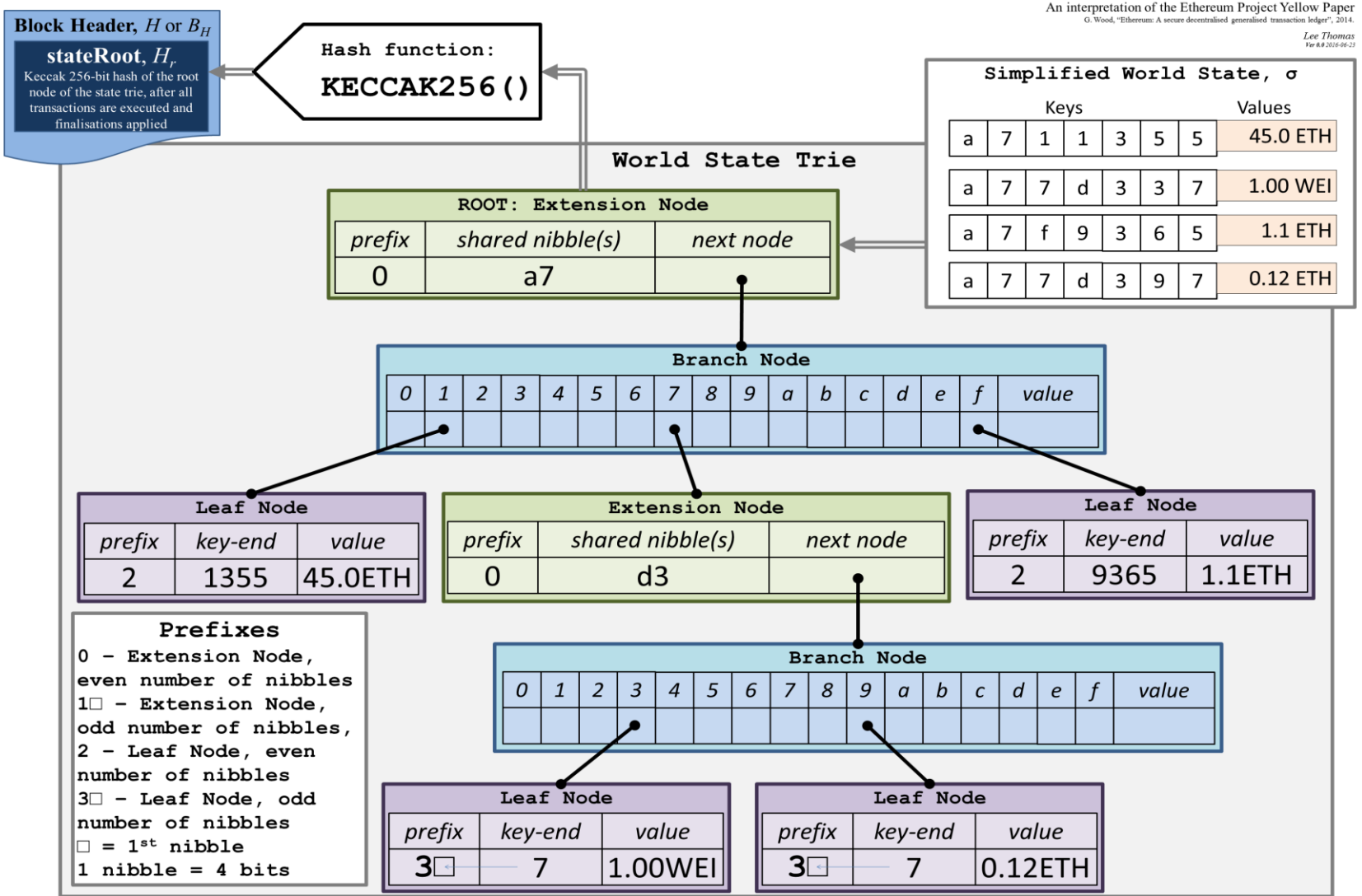


https://en.wikipedia.org/wiki/Trie

Merkle Patricia Trie

- 数据存储在LevelDB数据库中，其中Key：节点RLP编码的SHA3 hash值，Value：节点的RLP编码；
- 节点类型：
  - 空节点：空，没有值；
  - 叶子节点：Key-Value列表，key是十六进制编码，Value是RLP；
  - 扩展节点：Key-Value列表，Value是其他节点的哈希值；
  - 分支节点：一个长度17的列表，前16个表示对应key中的16个可能的十六进制字符，最后一个是value

MPT示例


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 0.9 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

RLP（Recursive Length Prefix）

- 如果是单字节，且值范围为[0x00, 0x7f]，RLP就是其本身；
- 如果是长度0-55的字符串，RLP编码是0x80-0xb7（0x80+55）的单字节前缀，后跟字符；
- 如果长度大于55，RLP编码格式是[prefix,len,字符..]，prefix=0xb7+字符串二进制长度的字节长度，len=字符串长度；
- 如果是长度0-55的列表： RLP编码格式是[prefix,len,RLP_item]， prefix =0xc0+列表长度，len=字符串长度，后面是各个元素的RLP编码；
- 如果是长度大于55的列表： RLP编码格式是[prefix,len,RLP_item]， prefix =0xf7+字符串二进制长度的字节长度，len=列表长度，后面是各个元素的RLP编码；

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

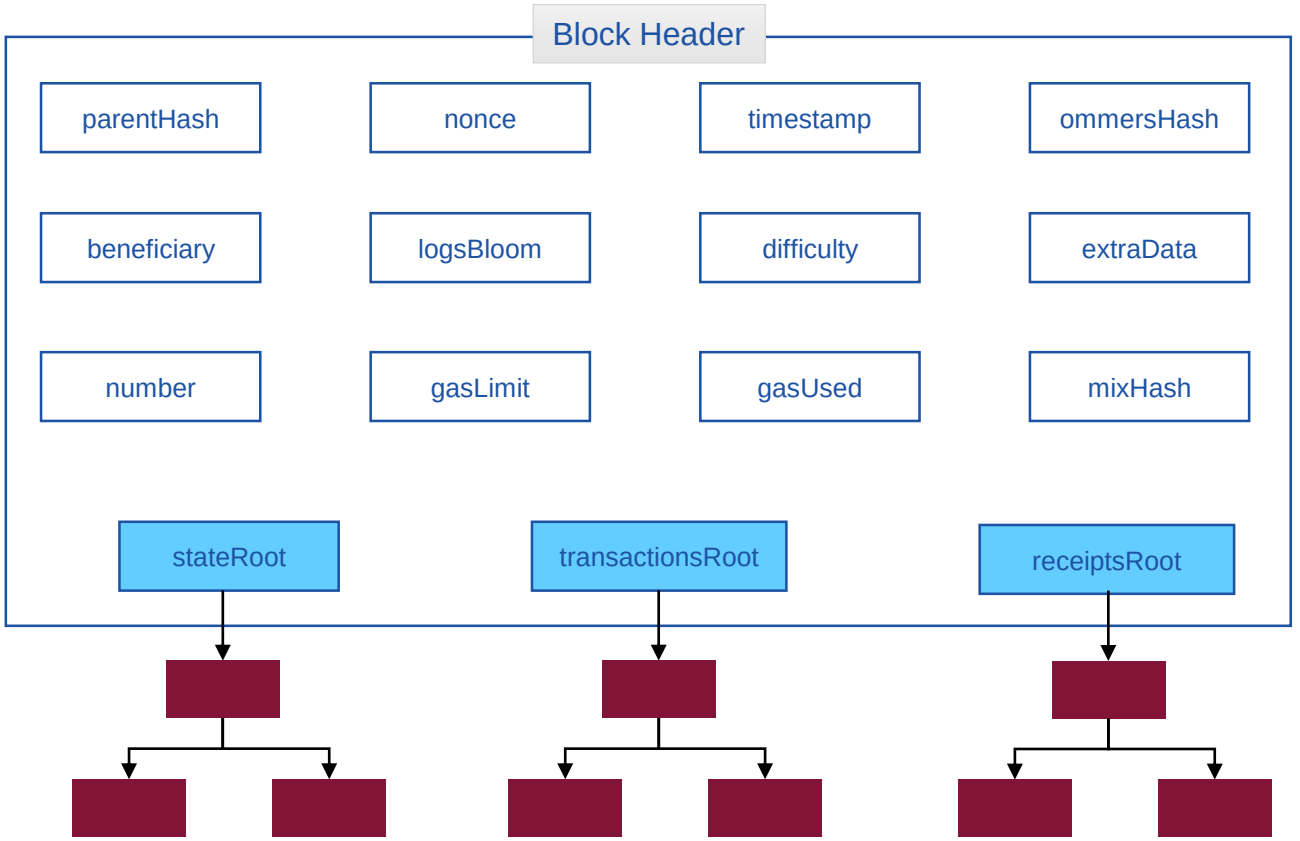
RLP

- [0x00 .. 0x7f] : 单字节字符;
- [0x80 .. 0xb7] : 长度小于等于55的字符串;
- [0xb8 .. 0xbf] :长度大于55的字符串;
- [0xc0 .. 0xf7] : 最多55个元素的列表;
- [0xf8 .. 0xff] : 55个以上的列表

RLP举例

- The string "dog" = [0x83, 'd', 'o', 'g']
- The list ["cat", "dog"] = [0xc8, 0x83, 'c', 'a', 't', 0x83, 'd', 'o', 'g']
- The empty string ('null') = [0x80]
- The empty list = [0xc0]
- The integer 0 = [0x80]
- The encoded integer 0 ('\x00') = [0x00]
- The encoded integer 15 ('\x0f') = [0x0f]
- The encoded integer 1024 ('\x04\x00') = [0x82, 0x04, 0x00]
- The set theoretical representation of three, [[], [[]], [[], [[]]]] = [0xc7, 0xc0, 0xc1, 0xc0, 0xc3, 0xc0, 0xc1, 0xc0]
- The string "Lorem ipsum dolor sit amet, consectetur adipisicing elit" = [0xb8, 0x38, 'L', 'o', 'r', 'e', 'm', ' ', ... , 'e', 'l', 'i', 't']

区块头


Block Header
parentHash
nonce
timestamp
ommersHash
beneficiary
logsBloom
difficulty
extraData
number
gasLimit
gasUsed
mixHash
stateRoot
transactionsRoot
receiptsRoot

WeBank

谢谢！