

RareSkills - Advanced Solidity

Meek Msaki (Meek#6464)

May 2023

Advanced solidity bootcamp with Dominik Teiml. Summary of [the ethereum yellow paper](#).

1 Chapter 1

2 Chapter 2

3 Chapter 3

- σ - (sigma) World state
- μ - (mu) Machine state
- Υ - (upsilon) Ethereum state transition
- C - General cost function
- KEC - Keccak-256 hash function
- $KEC512$ - Keccak512 hash function
- T - Ethereum transaction
- δ - (delta) the number of items required on a stack for a given operation
- \mathbf{o} - the output data of a message call
- \mathbb{N} - a set of scalar non-negative numbers e.g. set of integers smaller than 2^{256} denoted by \mathbb{N}_{256}
- \mathbb{B} - a set of byte sequences, e.g. bytes 32 is denoted by \mathbb{B}_{32}
- f - a function
- ℓ - a function which evaluates the last item in the given sequence

4 Chapter 4

4.1 The World State

- σ - represents world state or state. It is a mapping between addresses (160-bit) and account states, a data structure serialised as Recursive Length Prefix (RLP)
 - $\sigma[a] = \mathbf{Account\ state}$ - The state of ethereum accounts has four fields
 - * $\sigma[a]_n = \mathbf{account's\ nonce\ state}$
 - * $\sigma[a]_b = \mathbf{account's\ balance\ state}$
 - * $\sigma[a]_s = 256\text{-bit\ hash\ storageRoot\ of\ merkle\ patricia\ trie\ root\ node\ that\ encodes\ storage\ contents\ (a\ mapping\ between\ 256\text{-bit\ integer\ values)}$

- For Eternally Owned Accounts, $\sigma[a]_s = \emptyset$
- For Contract accounts $\sigma[a]_s \neq \emptyset$
- It's not hash of the merkle trie root but for the key/value pairs stored within $\sigma[a]_s \equiv \text{TRIE}(L_I^*((\sigma[a]_s)))$
- * $\sigma[a]_c = \text{codeHash}$ - Keccak 256-bit hash of contract bytecode that gets executed when the account receives a message call
 - For Eternally Owned Accounts, $\sigma[a]_c = \emptyset$
 - For contract accounts $\sigma[a]_c \neq \emptyset$
 - $\text{KEC}(\mathbf{b}) = \sigma[a]_c$, \mathbf{b} to denote the contract's EVM bytecode
- An account is *empty* if it has no code $\sigma[a]_c = \emptyset$, zero nonce $\sigma[a]_n = 0$ and zero balance $\sigma[a]_b = 0$
 - * $\text{EMPTY}(\sigma, a) \equiv \sigma[a]_c = \text{KEC}(\emptyset) \wedge \sigma[a]_n = 0 \wedge \sigma[a]_b = 0$
- An account is *dead* if its account state is non-existent or empty
 - * $\text{DEAD}(\sigma, a) \equiv \sigma[a] = \emptyset \vee \text{EMPTY}(\sigma, a)$

4.2 The Transaction

- T - a single cryptographically signed instruction by an EOA.
- S - maps transaction to the sender with ECDSA SECP-256k1 curve (hash of the transaction excepting three signature fields).
- Assert the sender of a transaction T represents $S(T)$
- $L_T(T) \equiv \begin{cases} (T_n, T_p, T_g, T_t, T_v, T_i, T_w, T_r, T_s) & \text{if } T_t = \emptyset \\ (T_n, T_p, T_g, T_t, T_v, T_d, T_w, T_r, T_s) & \text{otherwise} \end{cases}$ - Specifies how to serialize a transaction

The sender of a transaction cannot be a contract

There are two subtype of transactions

- Those that result in message calls
- Those that result in creation of new accounts with associated code ('contract creation')

EIP-1559: Fee market change (type 2) transactions have:

- T_n - Transaction nonce for the sender, $T_n \in \mathbb{N}_{256}$
- T_g - Gas Price, $T_p \in \mathbb{N}_{256}$
- T_g - Gas Limit, $T_g \in \mathbb{N}_{256}$
- T_t - To (160-bit address), $T_t \in \begin{cases} \mathbb{B}_{20} & \text{if } T_t \neq \emptyset \\ \mathbb{B}_0 & \text{contract creation} \end{cases}$
- T_i - EVM-code for account initialization, $T_i \in \mathbb{B}$
- T_r, T_s - Signature of the transaction used to determine the sender of the transaction
- T_w - **ChainId** and **yParity** are combined to form single value $T_w = 2\beta + 35 + T_y$ (see EIP-155 by Buterin [2016b]), $T_w \in \mathbb{N}_{256}$
- T_x - EIP-2718 transaction type, $T_x \in \{1, 2\}$
- T_p - Gas Price, $T_p \in \mathbb{N}_{256}$
- T_g - Gas Limit, $T_g \in \mathbb{N}_{256}$
- T_v - Value, $T_v \in \mathbb{N}_{256}$
- T_r, T_s - Signature of the transaction used to determine the sender of the transaction, $T_r \in \mathbb{N}_{256}$, $T_s \in \mathbb{N}_{256}$
- T_A - Access list (Legacy transactions don't have access list)
- T_c - Chain ID, $T_c = \beta$

- T_y - yParity signature, $T_y \in \mathbb{N}_1$ (bool)
- T_i - init is an unlimited size byte array specifying the EVM-code for contract initialization procedure, it is fragment that returns a **body**, $T_i \in \mathbb{B}$
- T_d - data byte array specifying input data of a message call, $T_d \in \mathbb{B}$

```
@dataclass
class Transaction1559Payload:
    chain_id: int = 0
    signer_nonce: int = 0
    max_priority_fee_per_gas: int = 0
    max_fee_per_gas: int = 0
    gas_limit: int = 0
    destination: int = 0
    amount: int = 0
    payload: bytes = bytes()
    access_list: List[Tuple[int, List[int]]] = field(default_factory=list)
    signature_y_parity: bool = False
    signature_r: int = 0
    signature_s: int = 0
```

EIP-2930 (type 1) transactions also have:

- T_A - Access list (Legacy transactions don't have access list)
- T_c - Chain ID must equal chain ID β , $T_c = \beta$
- T_y - yParity signature

```
@dataclass
class Transaction2930Payload:
    chain_id: int = 0
    signer_nonce: int = 0
    gas_price: int = 0
    gas_limit: int = 0
    destination: int = 0
    amount: int = 0
    payload: bytes = bytes()
    access_list: List[Tuple[int, List[int]]] = field(default_factory=list)
    signature_y_parity: bool = False
    signature_r: int = 0
    signature_s: int = 0
```

4.3 The Block

The Block, $B \equiv (B_H, B_T, B_U)$, is a collection of the following relevant pieces of information:

1. **T** - information corresponding to transactions
 - B_T - A series of transactions from this block
2. **U** - A set of block headers, (*ommers*²) or uncles
 - B_U - A list of ommer block headers
3. H - represents block header, which contains: $\mathbb{B} = \{B : B \in \mathbb{B} \vee \|B\| = n\}$
 - (a) H_p - **parentHash** - keccak256-bit hash of the parent's block header
 - $H_p \in \mathbb{B}_{32}$
 - $H_p \equiv KEC(P(B_H))$ where $P(B_H)$ is the parent block header of B
 - $TRIE(L_s(\sigma)) = P(B_H)_{H_r}$

- (b) H_o - **ommersHash** - keccak256-bit hash of the ommers list of this block
 - $H_o \in \mathbb{B}_{32}$
 - $H_o \equiv KEC(RLP(L_H^*(B_U)))$
- (c) H_c - **beneficiary** - 160-bit address that receives of all fees collected from successful mining this block
 - $H_c \in \mathbb{B}_{20}$
- (d) H_r - **stateRoot** - keccak256 hash of the root node of the state trie after all transaction are executed on this block, $H_r \equiv KEC(TRIE(L_s(\Pi(\sigma, B))))$
 - $H_r \in \mathbb{B}_{32}$
 - Π - transaction state's accumulation function
- (e) H_t - **transactionRoot** - keccak26-Bit hash of trie's root node with this block's transactions
 - $H_t \in \mathbb{B}_{32}$
 - $H_t \equiv KEC(TRIE(\{\forall i < \|B_T\|, i \in \mathbb{N} : p_T(i, B_T[i])\}))$
- (f) H_e - **receiptsRoot** - keccak256-bit hash of the trie's root node containing receipt's of each transaction from this block, $H_e \equiv KEC(TRIE(\{\forall i < \|B_R\|, i \in \mathbb{N} : p_R(i, B_R[i])\}))$
 - $H_e \in \mathbb{B}_{32}$
 - B_R - values stemming from the computation of transactions, specifically transaction receipts
- (g) H_b - **logsBloom** - Bloom filter from indexable info (logger address and log topics) contained in each log entry from the receipt of each transaction in this block
 - $H_b \in \mathbb{B}_{256}$
- (h) H_d - **difficulty** - a scalar value of the difficulty level of this block
 - $H_d \in \mathbb{N}$
- (i) H_i - **number** - a scalar value equal to the number of ancestor blocks
 - $H_i \in \mathbb{N}$
- (j) H_l - **gasLimit** - a scalar value equal to the current limit of gas expenditure per block
 - $H_l \in \mathbb{N}$
- (k) H_g - **gasUsed** - a scalar value equal to the total gas used in transactions in this block
 - $H_g \in \mathbb{N}$
- (l) H_s - **timestamp** - a scalar value equal to the output of Unix's time() at the block's inception
 - $H_s \in \mathbb{N}_{256}$
- (m) H_x - **extraData** - arbitrary byte array data relevant to this block, must be 32 bytes or less
 - $H_x \in \mathbb{B}$
- (n) H_m - **mixHash** - 256-bit hash which combined with nonce, proves a sufficient amount of computation has been carried out in this block
 - $H_m \in \mathbb{B}_{32}$
- (o) H_n - **nonce** - 64-bit value which combined with mix-hash proves sufficient computation has been carried out
 - $H_n \in \mathbb{B}_8$

4.3.1 Transaction Receipt

- $B_R[i]$ - receipt for the i^{th} transaction in an index-keyed trie who's root is recorded in the header as H_e
- $R \equiv (R_x, R_z, R_u, R_b, R_l)$
- $L_R(R) \equiv (R_z, R_u, R_b, R_l)$ - the L_R function prepares tx receipt to be transformed into an RLP-serialized bytes array

- R - Transaction receipt (a tuple of 5 items):
 1. R_x - equal to the corresponding transaction type
 2. R_z - status code of the transaction, $R_z \in \mathbb{N}$ (non-negative integer)
 3. R_u - cumulative gas used, $R_u \in \mathbb{N}$ (non-negative integer)
 4. R_l - a series of logs entries (O_0, O_1, \dots) created through execution of the transaction:
 - (a) A log entry, O , is a tuple of: $O \equiv (O_a, (O_{t0}, O_{t1}, \dots), O_d)$
 - i. O_a - logger's address, $O_a \in \mathbb{B}_{20}$
 - ii. O_t - series of 32-byte log topics, $\forall x \in O_t : x \in \mathbb{B}_{32}$
 - iii. O_d - some number of bytes data, $O_d \in \mathbb{B}$
 5. R_b - the Bloom filter composed from the information in the logs, $R_b \in \mathbb{B}_{256}$
 - (a) M - a function (bloom filter) to reduce a log entry into a single 256-byte hash:
 - i. $M(O) \equiv \bigvee_{x \in O_a \cup O_t} (M_{3:2048}(x))$
 - A. $M_{3:2048}$ - a specialized Bloom filter that sets three bits out of 2048, given an arbitrary byte sequence
 - B. $M_{3:2048}$ - takes the low-order 11 bits of each of the first 3 pairs of bytes in a keccak-256 hash of the byte sequence
 - C. $M_{3:2048}(\mathbf{x} : \mathbf{x} \in \mathbb{B}) \equiv \mathbf{y} : \mathbf{y} \in \mathbb{B}_{256}$ where:
 - $\mathbf{y} = (0, 0, 0, \dots, 0)$ except:
 - $\forall i \in \{0, 2, 4\} : B_{2047-m(\mathbf{x}, i)}(\mathbf{y}) = 1$
 - B - a bit reference function
 - D. $m(\mathbf{x}, i) \equiv KEC(\mathbf{x})[i, i + 1] \bmod 2048$

4.3.2 Holistic Validity

- identity of state when transactions executed in order on the base state
 - $H_r \equiv TRIE(L_s(\Pi(\sigma, B))) \wedge$
 - $H_o \equiv KEC(RLP(L_H^*(B_U))) \wedge$
 - $H_t \equiv TRIE(\{\forall i < \|B_{\mathbf{T}}\|, i \in \mathbb{N} : p_T(i, B_{\mathbf{T}}[i])\}) \wedge$
 - $H_e \equiv TRIE(\{\forall i < \|B_R\|, i \in \mathbb{N} : p_R(i, B_R[i])\}) \wedge$
 - $\bigvee_{r \in B_R} (r_b)$

(TODO: to be continued... pg 6/41)

4.3.3 Serialization

- $L_H(H)$ - preparation function for block header
 - $L_H(H) \equiv (H_p, H_o, H_r, H_t, H_e, H_b, H_d, H_i, H_l, H_g, H_s, H_x, H_m, H_n)$
- $L_B(B)$ - preparation function for block
 - $L_B(B) \equiv (L_H(B_H), \tilde{L}_T^*(B_H), L_H^*(B_U))$
 - \tilde{L}_T - takes special care of EIP-2718 transactions

4.3.4 Block Header Validity

- $P(H) \equiv B' : KEC(RPL(B'_H)) = H_p$
- $H_i \equiv P(H)_{Hi} + 1$ - the block number is the parent's block number uncremented by one
- $D(H)$ - canonical difficulty fo a block header H
- ς_2 - *Homestead* diffuculty parameter
- ϵ - exponetial difficulty symbol

5 Chapter 5

6 Chapter 6 (Transaction Execution)

- Υ - state transition function
- σ' - the post-transactional state
- $\sigma' = \Upsilon(\sigma, T)$ - Υ is the function, T is the transaction and σ the state
- Υ^g - evaluates amount of gas used in the transaction
- Υ^l - evaluates transaction's accrued log items
- Υ^z - evaluates the status code resulting from the transaction

6.1 Substate

Accrued substate (A) is the information that is acted upon immediately following the transaction execution

- $A \equiv (A_s, A_l, A_t, A_r, A_a, A_K)$ - A is a tuple
- A_s - the self destruct set, a set of accounts that will be discarded following transaction completion
- A_l - log series of archived and indexable 'checkpoints' in VM code execution
- A_t - set of touched accounts, which the empty ones are deleted at the end of transaction
- A_r - the refund balance, from SSTORE instruction when contract storage is reset to zero from non-zero
- A_a - the set of accessed account addresses
- A_K - a tuple of a 20-byte account address and a 32-byte storage slot (a set of storage keys)
- A^0 - an empty accrued substate
- $A^0 \equiv (\emptyset, (), \emptyset, 0, \pi, \emptyset)$, where: π is a set of precompiled addresses
- π - a set of all precompiled addresses

6.2 Execution

- g_0 - amount of gas this transaction requires to be paid prior to execution