Ethereum

Zichao Yang

Zhongnan University of Economics & Law

Date: December 10, 2025

Part IV. Ethereum Accounts

Ethereum Accounts & Addresses

In previous lecture, we have discussed that Ethereum accounts can be categorized into:

- **Externally Owned Accounts (EOAs)**: Controlled by private keys. These accounts do not contain code but can send transactions.
- **Contract Accounts**: Controlled by smart contracts. These accounts contain code (i.e., the smart contract) that is executed when interacted with.

An **address** in Ethereum is a **unique identifier** that represents the location of an account on the blockchain.

Ethereum Account Structure

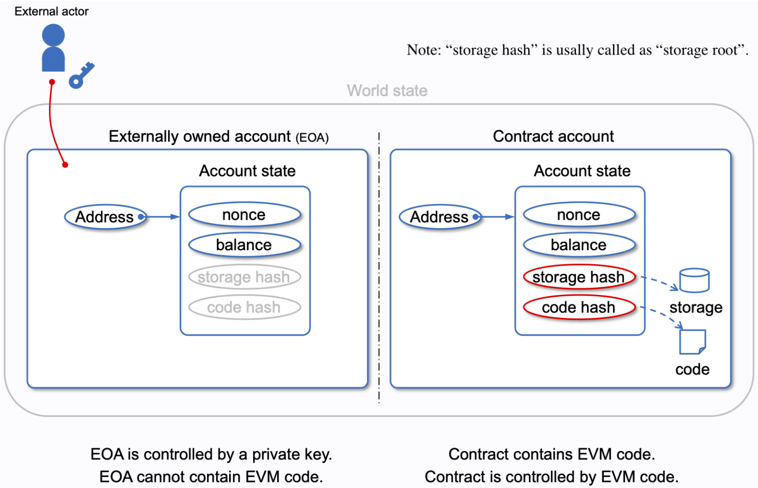

External actor
Note: “storage hash” is usally called as “storage root”.
World state
Externally owned account (EOA)
Account state
Address
nonce
balance
storage hash
code hash
Contract account
Account state
Address
nonce
balance
storage hash
code hash
storage
code
EOA is controlled by a private key.
EOA cannot contain EVM code.
Contract contains EVM code.
Contract is controlled by EVM code.

Figure 1: Ethereum Account Structure

Source: Ethereum EVM illustrated

Account Components

Both EOAs and contract accounts share the following components:

1. **Nonce**: a counter that tracks the number of transactions sent from the account.
2. **Balance**: the amount of Ether held by the account.
3. **Address**: a unique 160-bit identifier for the account, derived differently for EOAs and contract accounts.
    - EOA address is the last 20 bytes of the Keccak-256 hash of the public key.
    - Contract accounts address is derived from the sender EOA's address and its nonce at the time of deployment.

Account Components

Both EOAs and contract accounts share the following components:

4. **Storage Root**: a hash of the root node of a Merkle Patricia trie that stores the account's data.
   - For EOAs, this is typically empty since they don't have associated storage.
   - For contract accounts, this is used to store the state variables of the smart contract.

5. **Code Hash**: a hash of the contract's bytecode.
   - For EOAs, this is the hash of the empty string because they do not have associated code.
   - For contract accounts, this hash points to the bytecode stored on the blockchain.

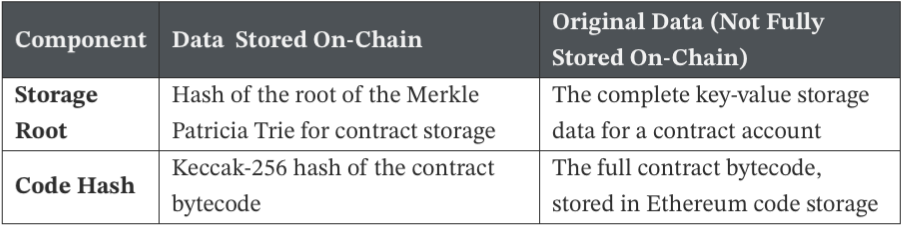
Account as A Conceptual Framework

We can also think the concept of “account” as a conceptual framework and it comprises components that are distributed between the user’s **device** and the Ethereum **blockchain**.

1. **On-Device Components**:
   - **Private Key**: Enables signing transactions and proving the ownership of EOAs.
   - **Public Key**: Derived from the private key, used to generate the EOA’s address.
2. **On-Chain Components**:
   - Address
   - Nonce
   - Balance
   - Code Hash
   - Storage Root

Account State

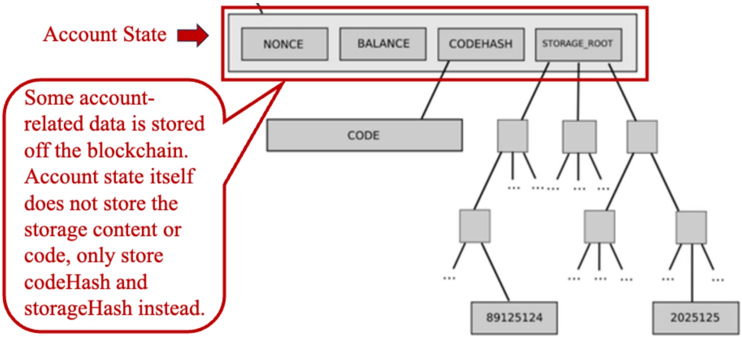
The **account state** contains all the dynamic and mutable information related to an account, such as nonce, balance, storage root, and code hash.



Component	Data Stored On-Chain	Original Data (Not Fully Stored On-Chain)
Storage Root	Hash of the root of the Merkle Patricia Trie for contract storage	The complete key-value storage data for a contract account
Code Hash	Keccak-256 hash of the contract bytecode	The full contract bytecode, stored in Ethereum code storage

Figure 2: Comparison of Data Stored On-Chain vs. Original Data

Ethereum Account Structure



Some account-related data is stored off the blockchain. Account state itself does not store the storage content or code, only store codeHash and storageHash instead.

Figure 3: Ethereum Account Structure

Part V. Ethereum Blocks

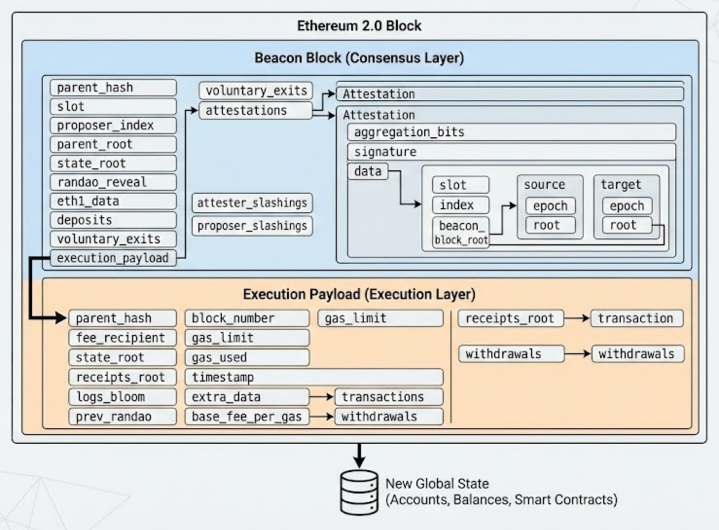
Ethereum Block

Ethereum transitioned from Proof-of-Work (PoW) to Proof-of-Stake (PoS) on September 15, 2022, resulting in a significant change to the block structure.

Most of the available information online about Ethereum's block structure is outdated.
Indicator: if you see terms like *nounce* and *difficulty* in the *block header*, the information refers to the old PoW model and is no longer accurate.

For the latest details on Ethereum's block structure, visit the official website.

A Sketch of the Block Structure



Source: Ethereum website, Credit: Google Nano Banana Pro

Proof-of-Stake (PoS) Consensus

PoS algorithms use several methods to select who will validate the next block:

- **Size of staked tokens**: the more tokens staked, the higher the chance of being chosen to validate.
- **Age of staked tokens**: the longer the tokens have been unspent, the higher the chance of being chosen to validate. Once that stake is used to verify a block, its age is reset to zero.
- **Random Selection**

Proof-of-Stake in Ethereum 2.0

Shard chain: mainly addresses the scalability concern. There are 64 shard chains in the Ethereum network. (**Note**: The idea of Shard Chain is abandoned. If you see this, skip, don't wast your time...)

Beacon chain: coordinates the information and manages validators.

Validator: the nodes in the Ethereum network who want to confirm transactions. Users need to stake 32 ETH to become validators.

Proof-of-Stake in Ethereum 2.0

The Beacon Chain (Consensus Layer):

- The “Heartbeat” of Ethereum. It coordinates validators and manages the Proof-of-Stake protocol.
- Does **not** process smart contracts directly; it ensures consensus on the data.

The Scalability Evolution (Sharding):

- **Old Plan (Abandoned)**: “Execution Sharding” (Splitting the network into 64 separate blockchains).
- **Current Reality**: “Danksharding” (Data Availability).
- Instead of splitting the chain, Ethereum acts as a data storage layer for **Layer 2 Rollups** (e.g., Arbitrum, Base, Polygon), which handle the execution. (Important EIP: EIP-4844)

Proof-of-Stake in Ethereum 2.0

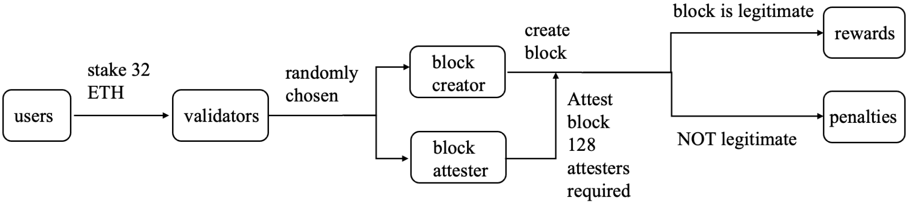


Figure 4: Proof-of-Stake in Ethereum

Source: Ethereum Foundation: Proof-of-Stake (PoS)

Proof-of-Stake in Ethereum 2.0

- **Pros**:
  - Better energy efficiency
  - Lower barriers to entry (hardware requirement is low)
  - Stronger immunity to centralization
- **Cons**:
  Compared with Proof-of-Work, Proof-of-Stake is more complex and less battle-tested in the real world.

Q: Does 51% attack still exist in proof-of-stake?

Extra: Delegated Proof of Stake

In DPoS, users do not directly become validators, they vote for delegates to become validators.

Users vote on delegates by pooling their tokens into a staking pool and linking those tokens to a particular delegate.

These delegates are usually called as witnesses or block producers.

Extra: Delegated Proof of Stake

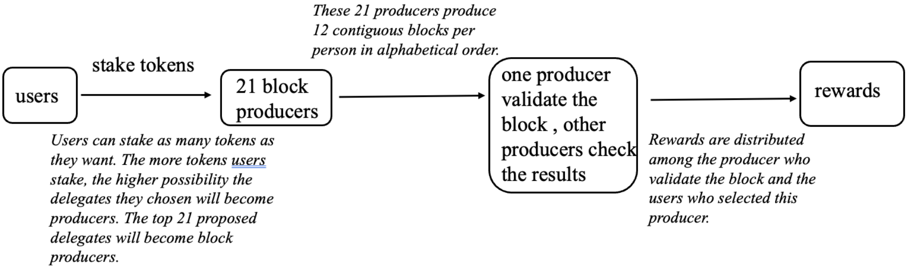

users
stake tokens
21 block producers
These 21 producers produce 12 contiguous blocks per person in alphabetical order.
one producer validate the block , other producers check the results
rewards
Users can stake as many tokens as they want. The more tokens users stake, the higher possibility the delegates they chosen will become producers. The top 21 proposed delegates will become block producers.
Rewards are distributed among the producer who validate the block and the users who selected this producer.

Figure 5: EOS: Delegated Proof of Stake

Source: EOS: Delegated Proof-of-Stake (DPoS)