Report:

Ethereum, Consensus Mechanisms, DApps, Web3, Metamask, Smart Contract

Prepared By: Abhilash Rajan
Date: 26 September 2025

Ethereum

The blockchain ledger is **equipped to store smart contracts**. Whenever a smart contract is deployed, each node in the network also gets a copy of the smart contract. Being self-executing programs, there is no way to intervene and terminate a smart contract from execution. There should be checks to ensure terminability of operations.

To ensure the above properties, it is **critical for a contract to be kept isolated in a sandbox** to save the entire ecosystem from any negative effects. This sandbox environment is the **runtime environment for smart contracts** in Ethereum and is **called Ethereum Virtual Machine (EVM)**. The components within an EVM may belong to a volatile machine state, a persistent world state or a virtual ROM. Ethereum is a transaction-based state machine.

The pre-paid fee is represented in **Ethereum's native currency- Ether**. Just like how fuel consumption is measured in litres, the **computational effort in Ethereum is measured in units of Gas**. Once the effort is measured in gas, the fee for those specific units of gas is calculated in Ether. **Gas fees are usually represented in wei**, the smallest denomination of Ether.

1 Ether = 1e18 wei (1,000,000,000,000,000,000)

EVM Components

1. **Machine State -** This is a volatile area and it changes with each instruction executed by the EVM. The machine state is cleared after processing each transaction.
    - **Program Counter -** stores address of the next instruction to be executed.
    - **Stack** - Stores intermediate values in computations & can hold 256-bit words, max size of 1024 elements.
    - **Gas available** - Gas Amount paid by the originator of a transaction.
    - **The memory area** - stores temporary variables of a smart contract. Only accessible during contract execution and the contents get removed between calls.
2. **Virtual ROM -** An immutable area where the EVM bytecode of the smart contract being executed is loaded, to use whenever required for execution.
3. **World State -** Defined by the account balances, contract storage and more. It is persistent, the data will be retained as long as the blockchain is live.

Transaction Steps

1. Before executing a transaction, the **EVM is initialized with the required data**.
2. **The ROM will be loaded** with the code of that specific smart contract **in bytecode format**.
3. **The program counter is set to zero**.
4. The smart contract's **state will be loaded to the storage** and the **memory is initialized** to all zeros.
5. **The gas available is set to the amount of gas paid** by the transaction sender. As code execution progresses, the gas is reduced according to the cost of the required operations.
6. If the gas hits 0, the execution will be halted. If the execution completes successfully, the remaining gas (if any) will be returned to the sender.
7. Finally, the **world state will be updated** to reflect the state changes imposed by the transaction.

Transaction Fee (paid in Ether) = Gas Limit *(Base fee + Tip)

A **Gas Limit** is the total amount of gas that can be spent by all transactions in the block.

Base fee is the minimum price per unit of gas the user has to pay. It will be burnt (destroyed by removing from circulation to make ETH deflationary), so users are expected to include a **Tip (priority fee)** in their transactions that incentivizes validators to process transactions and is usually set automatically by wallets.

Ethereum Account

One requires an Ethereum account to interact with the network. It can hold or send Ether to another account on the Ethereum blockchain network. Ethereum has 2 main types of accounts:
Externally Owned Accounts (EOAs) - Individual Users
Contract accounts - A smart contract code.

The public key resembles a bank account number. Similarly, the private key corresponds to the secret PIN. A private key is just a randomly picked 256-bit number. The public key is calculated from the private key using an irreversible function.

Ethereum accounts have four fields:

- **nonce** - Ensures that transactions are only processed once.
- **balance** - the amount of Ether owned by the address.
- **codeHash** - Refers to the hash of the code associated with an account in the EVM. Contract accounts have value in the codeHash field. Once assigned, this value cannot be changed.
- **storageRoot** - The 256-bit hash of the root node of a Merkle Patricia Trie, a structure used to represent the account storage.

Block Size

A block may contain an **ordered list of transactions along with some metadata** created by validators selected by the proof of stake consensus. Every new block will have a **reference to the parent block (previous block)**. These references are **hashes** that are cryptographically derived from the data of the block. The blocks are bound by a size limit. Each block has a gas limit. The size of blocks may vary according to network demands. It can increase until the maximum block limit of 30 million gas.

Ethereum protocol will work properly only if all the participants in the network have the exact same copy of the blockchain ledger. **The consensus mechanism defines the rules by which the participants agree on the same copy of the ledger**. Initially, Ethereum was based on PoW consensus, where a miner was responsible for block creation. Later, a multi-stage upgrade plan for changing the consensus to PoS consensus was devised. Under the proof of stake consensus, a group of validators is responsible for creating blocks.

One can **become a validator by depositing 32 ETH** using a specific transaction in the deposit contract in the Ethereum mainnet. Once detected by the nodes, the validator is activated. The **validators are responsible for creating new blocks (block proposer) and voting on a proposed block (committee).** When the block gets approved, the validators get a specific amount of newly created Ether as a reward and also penalized for dishonest behaviour. .

🔹 Blockchain Consensus: The Digital Agreement Engine

If thousands of strangers had to keep a single book of records without ever meeting — how would they agree on what's written inside?

Consensus is like the referee of blockchain — ensuring every transaction is valid, fair, and agreed upon by all nodes. Without it, blockchains would fall into chaos.

👉 Takeaway:
Consensus is the heartbeat of blockchain. From mining 🔨 to staking 🌱, burning 🔥 to voting 🗳️ — all methods serve one purpose:
✔️ To make strangers agree on a single, tamper-proof truth.

Major Consensus Mechanisms:

⚙️ 01. **Proof of Work (PoW)** – Computers compete by solving puzzles; secure but energy-heavy.
🌱 02. **Proof of Stake (PoS)** – Validators lock coins as collateral; efficient and scalable.
🗳️ 03. **Delegated Proof of Stake (DPoS)** – Community votes for representatives to validate on their behalf.
🔑 04. **Proof of Authority (PoA)** – Only trusted, pre-approved entities confirm transactions.
⏱️ 05. **Proof of Elapsed Time (PoET)** – Validators wait random times to fairly decide who adds blocks.
📦 06. **Practical Byzantine Fault Tolerance (PBFT)** – Nodes reach agreement through multiple message rounds.
💾 07. **Proof of Capacity (PoC)** – Disk storage space is used instead of computing power.
📡 08. **Proof of Activity (PoA)** – A hybrid of mining and staking combined for validation.
📊 09. **Proof of Importance (PoI)** – Reputation + activity + coins decide validation rights.
🔥 10. **Proof of Burn (PoB)** – Coins are destroyed to prove commitment and earn validation power.
🌍 11. **Hybrid Models** – Mixes different methods for balance between speed, scalability, and security.

dApps

The applications built on top of Ethereum maintain a decentralized nature and are known as Decentralized Applications (DApps). The code is kept open and transparent so anyone can audit and suggest changes to the instructions. They are the core of Web3, the decentralized web.. DApps on Ethereum has a flexible application stack.

Technical Stack:

- **User Interface**: Designed and developed using frameworks like React, Vue, Angular, Svelte, or using simple HTML-CSS-JS combination.
- **Libraries and Frameworks**: Web3JS, Ethers.js, Hardhat, Foundry help the communication with the blockchain.
- **Wallets**: Used to manage accounts in blockchain network. Metamask is a main example.
- **Smart Contracts**: The business logic embedded Self-Executing Programs written in Solidity, Vyper, LLL which are then compiled and deployed as bytecode to the blockchain network.
- **Ethereum Blockchain**: Simulations like Remix VM, Hardhat node are used during development for connecting Blockchain clients such as Besu, Teku, Lighthouse, and Prysm to the Ethereum network
- **OS**: The operating systems like Windows, Linux and Mac.

Web3 and Steps in Solidity for Smart Contract.sol

The Web3 creates decentralized internet to interact freely, share & collaborate on a global scale.

1. Specify the SPDX-License-Identifier tag for the source file. The tag should be specified at the beginning of each program. The syntax is:
   // SPDX-License-Identifier: name of the license
2. Specify the version of the Solidity language as it is constantly getting updated. The syntax is:
   pragma solidity 0.8.20; ← (version)
3. Create variables for storing & retrieving data from the blockchain and code functions for handling these variables inside a contract. General syntax for the contract is:
   contract [*name of contract***] {** *body* **}**
4. Declare a variable (number), of the type (uint256), to store numerical value. Syntax is:
   uint256 number;
5. Define a function {store()) to store data into the variable (number). The function is tagged with the keyword public, which means it's accessible to anyone who can interact with the contract.
   function store (uint256 num) public {
   number = num; }
6. Define another function (retrieve()) to read the data from the variable (number). The view keyword specifies how the function interacts with the state of the blockchain (either reading or writing on it).
   function retrieve() public view returns (uint256){
   return number; }

Execution of Smart Contract

Gavin James Wood invented solidity in **August 2014**. Later a team led by **Christian Reitwiessner**, developed it as a programming language targeting the EVM by the Ethereum Solidity. **To execute smart contract use a simple tool called Remix IDE**, an **open-source Ethereum Integrated Development Environment** which is available as a web and desktop application. It **has options to write, compile, debug and deploy Solidity code**. It is the preferred tool for learning smart contract development and familiarizing the Ethereum network. **Extension of Solidity file is '.sol'**.

https://remix.ethereum.org/

Write Solidity…Compile…Deploy…Interact with smart contract...

'Create New File'
'Storage.sol' as the file name. Paste our code there.
Click the tab switch icon to go to the 'SOLIDITY COMPILER' tab
Click 'Compile Storage.sol'.
If no errors, a green tick appears next to the 'SOLIDITY COMPILER' tab.
Go to the 'DEPLOY & RUN TRANSACTIONS' tab. Click on the 'Deploy' button.
If no errors, results appear under the 'Deployed/Unpinned Contracts' section.
Expand the 'STORAGE' contract entry.
Access functions written in smart contract. Write a number onto the blockchain using the text box next to the 'store' button and click the button. Click on the 'retrieve' button to read the value from the blockchain.

MetaMask Wallet

To interact with the Ethereum network, you need **a unique identifier, which is called an address**. A **wallet application is a handy tool for this** purpose. **The most widely trusted web3 wallet** is MetaMask. MetaMask is a **non custodial wallet**, **data relating with wallet and address is stored only in user PC**. MetaMask is not responsible to remember User credentials. MetaMask wallet is **connected to Ethereum Mainnet**, where ETH is obtained by buying from centralized and decentralized exchanges, earn from participants or known peers in ether, or stake rewards. Both networks require ETH to cover transaction costs.Ethereum operates on **two primary networks**:

- **Mainnet**: Where Ether (ETH) holds real-world value & production-ready dApps are deployed.
- **Testnet (Test Ether)**: A **replica of the Mainnet** with lower transaction costs and faster confirmation times. Used for **testing new protocols and dApps before deploying them** on the Mainnet.

metamask.io

Add Extension to Chrome
Create New MetaMask Wallet and Create password (save it in User PC)
Secure wallet using 12 word Secret Recovery Phrase, SRP, (save it in User PC)
On successful completion, pin the MetaMask in browser.