Blockchain Development on Metis

Fast Track Course - 2024

METIS

About Me

Some things you should know

- I want to help you learn about Web3 development.
- PhD in Physics (2023).
- Entrepreneur (Founder of Bright NFT).
- Dev Rels at Metis L2.
- Experience collaborating with dozens of projects in Web3.
- One Skill: Problem Solving.
- Practice > Theory.

Víctor Muñoz

Program Details

(Tentative)

Module 1: Introduction to Blockchain and Ethereum

Module 2: Solidity and Smart Contract Development

Module 3: Advanced Solidity Concepts

Module 4: Development Tools

Module 5: Frontend Integration

Module 6: Introduction to Metis L2 Network

Module 7: Advanced Metis L2 Topics

What is Metis?

OPTIMISTIC ROLLUPS SCALING ETHEREUM

Metis is an EVM-equivalent Optimistic Rollup built to scale Ethereum and onboard real-world use cases, besides enhancing existing ones.

Metis offers fast transactions, low gas fees, and allows developers to build on an easy-to-code blockchain.

Users, developers, and enterprises can benefit from enhanced security, enhanced capital efficiency, and progressive decentralization through our sequencer technology.

METIS

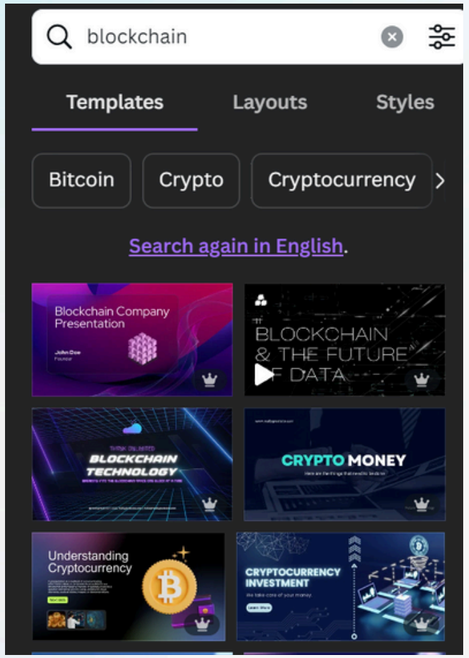
Introduction to Blockchain and Ethereum

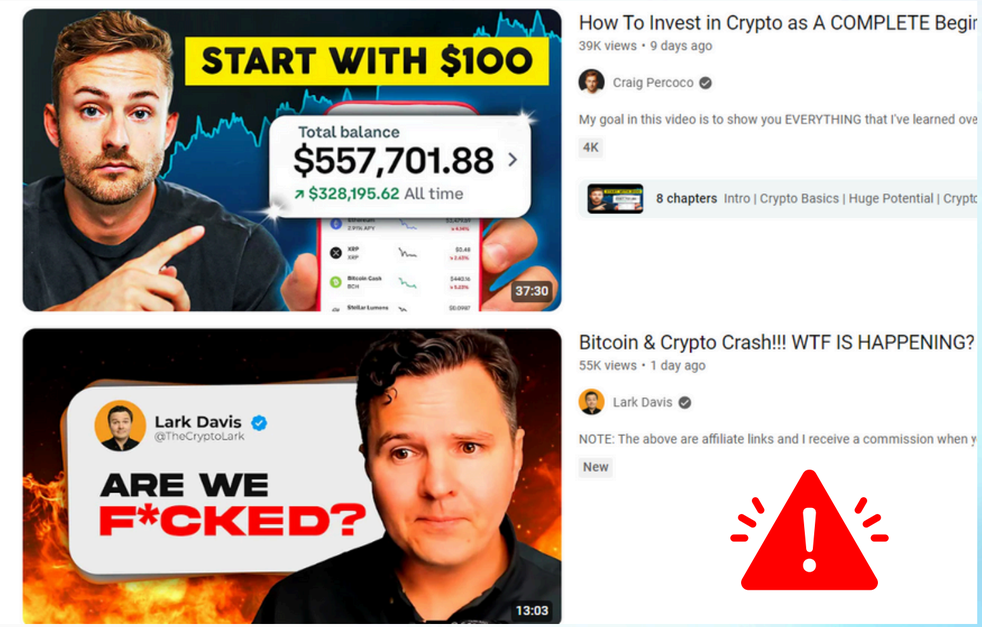
Infrastructure for a Trustless Economy

The Oxford Dictionaries : "A system in which a record of transactions made in cryptocurrency is maintained across several computers that are linked in a peer-to-peer network".

Investopedia: “Blockchains can make data immutable and reduce the need for trusted third parties.”

What to avoid:



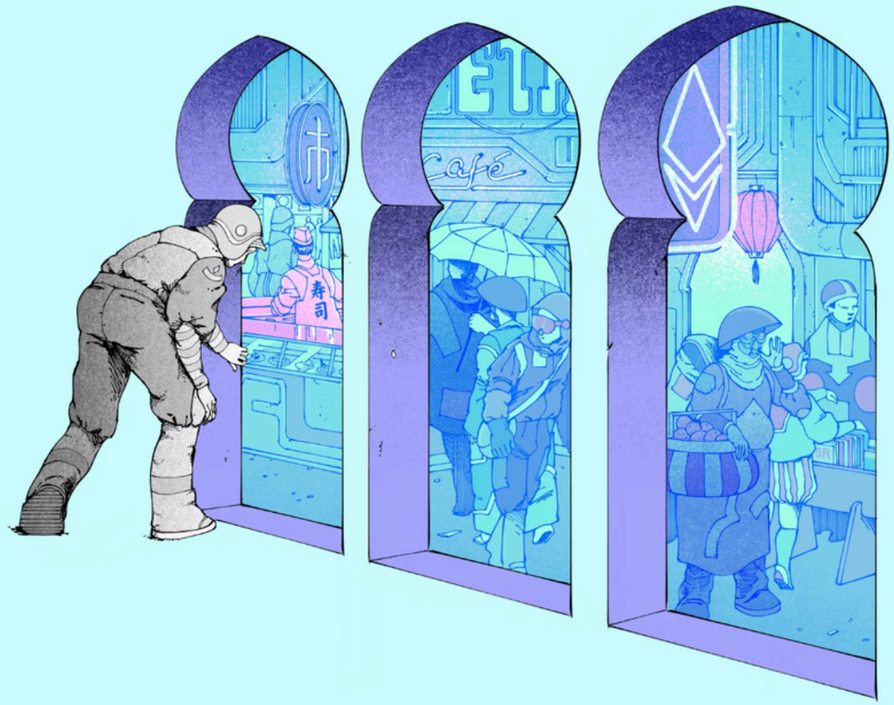


Introduction to Blockchain and Ethereum

Few Facts

- Public Blockchains: Typically require a cryptocurrency.
- Private/Consortium Blockchains: Usually do not need a cryptocurrency.
- Hybrid Blockchains: May or may not use a cryptocurrency, depending on the design.

Introduction to Blockchain and Ethereum



What is Ethereum?

Ethereum is a technology that's home to digital money, global payments, and applications. The community has built a booming digital economy, bold new ways for creators to earn online, and so much more. It's open to everyone, wherever you are in the world – all you need is the internet.

What is Ethereum? | More on digital money

Ethereum

An open source, globally decentralized computing infrastructure that executes programs called smart contracts, written in a Turing-complete programming language, translated into bytecode and run on a virtual machine.

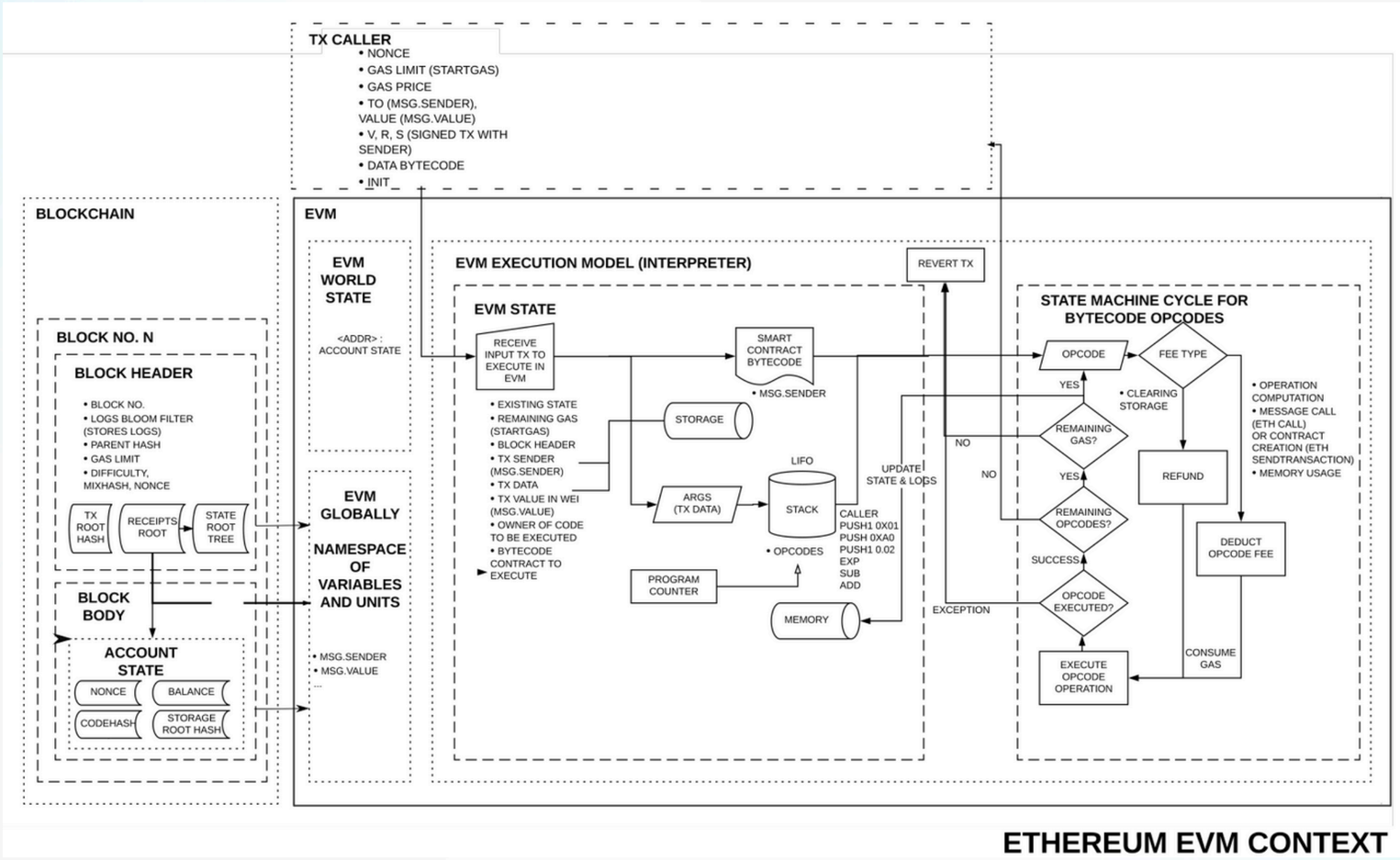
It uses a blockchain to synchronize and store the system’s state changes (key/value tuples), along with a cryptocurrency called ether to meter and constrain execution resource costs. It enables developers to build decentralized applications with built-in economic functions.

Ethereum

(PDF) Title: Research on Malicious Smart Contracts Detection in...

PDF | On Jun 14, 2019, □□□□ □□□ and others published Title: Research on Malicious Smart Contracts Detection in...

researchgate.net



What about Bitcoin?

It operates on a peer-to-peer network without a central authority, utilizing blockchain technology to record transactions on a public, immutable ledger. Bitcoin transactions are secured through cryptographic techniques, and new bitcoins are generated via mining, where miners solve complex puzzles to validate blocks and are rewarded with new coins. The total supply of Bitcoin is capped at 21 million, making it deflationary.

What problem does Bitcoin solve?

Double Spending: Bitcoin solves the double-spending problem, which is the risk that a digital currency can be spent more than once. Through its blockchain technology, each transaction is verified by a network of nodes and recorded in a public ledger, ensuring that each bitcoin is spent only once.

Traditional financial systems prevent double spending through centralized control, real-time verification, and robust clearing and settlement processes. Bitcoin, on the other hand, uses a decentralized ledger and consensus mechanisms to achieve the same goal without relying on a central authority.

What problem does Bitcoin solve?

https://bitcoin.org/bitcoin.pdf

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

Blockchain History

Early Concepts
1982: David Chaum introduces the concept of a blockchain-like protocol in his dissertation.
1991: Stuart Haber and W. Scott Stornetta propose a cryptographically secured chain of blocks.

Genesis of Bitcoin
2008: Satoshi Nakamoto publishes the Bitcoin whitepaper, outlining a peer-to-peer electronic cash system.
2009: Bitcoin network goes live with the mining of the genesis block (Block 0).

Evolution and Ethereum
2013: Vitalik Buterin proposes Ethereum, a decentralized platform for smart contracts.
2015: Ethereum network launches, enabling decentralized applications (dApps).

Growing Adoption
2017: ICO boom raises billions for blockchain projects; Bitcoin reaches near $20,000.
2018: Blockchain technology gains mainstream attention; numerous enterprise applications emerge.

Recent Developments
2020: DeFi (Decentralized Finance) experiences significant growth, leveraging smart contracts.
2021: NFTs (Non-Fungible Tokens) become a major trend, with record-breaking sales.
2022: Ethereum begins transitioning to Proof of Stake (PoS) with Ethereum 2.0 to enhance scalability and efficiency.

Crypto

Securing information by transforming it into an unreadable format, only accessible to those with a decryption key.

Cryptography in Blockchain

- **Public Key Cryptography**:
  - Digital Signatures: Transactions are signed with a private key, providing proof of ownership and authenticity.
  - Verification: The corresponding public key allows anyone to verify the signature without revealing the private key.

- **Hash Functions**:
  - Block Hashing: Each block contains a hash of the previous block, linking them together and ensuring immutability.
  - Proof of Work: Miners solve cryptographic puzzles involving hash functions to validate transactions and add new blocks.

Concrete Applications:

- Transaction Security: Ensures that only the owner of a private key can authorize transactions, preventing fraud.
- Data Integrity: Hash functions ensure that any alteration to a block's data would change its hash, making tampering detectable.
- Trustless Verification: Public key cryptography allows for secure, trustless transactions without needing a central authority.

Blockchain as a Computer Program

Software Nature of Blockchain

Definition: At its core, a blockchain is a distributed ledger technology implemented through software.

Components: It consists of a series of interlinked blocks that record transactions or data entries. These blocks are maintained and managed by a network of computers (nodes) running the blockchain software.

Blockchain as a Computer Program

Functions and Characteristics

- Distributed System: A blockchain operates on a distributed network of computers, each running the same program (the blockchain protocol). This network collectively maintains and verifies the ledger.
- Consensus Mechanisms: The software includes algorithms to achieve consensus among nodes, ensuring that all copies of the ledger are identical and up-to-date. Examples include Proof of Work (PoW) and Proof of Stake (PoS).
- Smart Contracts: Many blockchains, like Ethereum, support smart contracts—self-executing contracts with the terms directly written into code. These are also computer programs that run on the blockchain.

Blockchain as a Computer Program

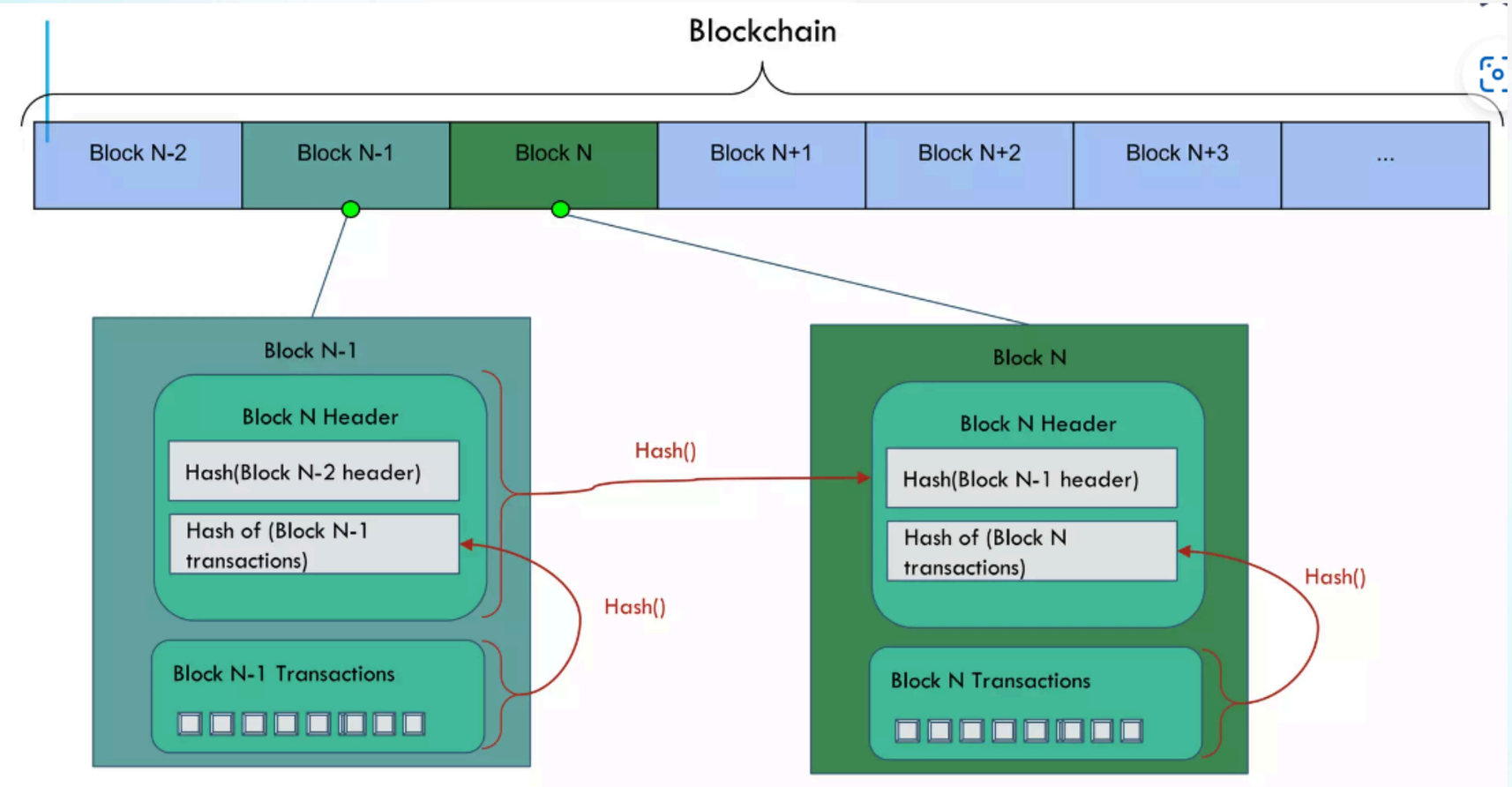
Immutability and Security

- Immutable Ledger: The blockchain program ensures that once data is written to the blockchain, it cannot be altered without altering all subsequent blocks and gaining majority consensus, which is computationally impractical.
- Cryptographic Security: The software uses cryptographic techniques to secure data, create digital signatures, and ensure the integrity and authenticity of transactions.

Lingo

vmmunoza/
Blockchain-Lingo

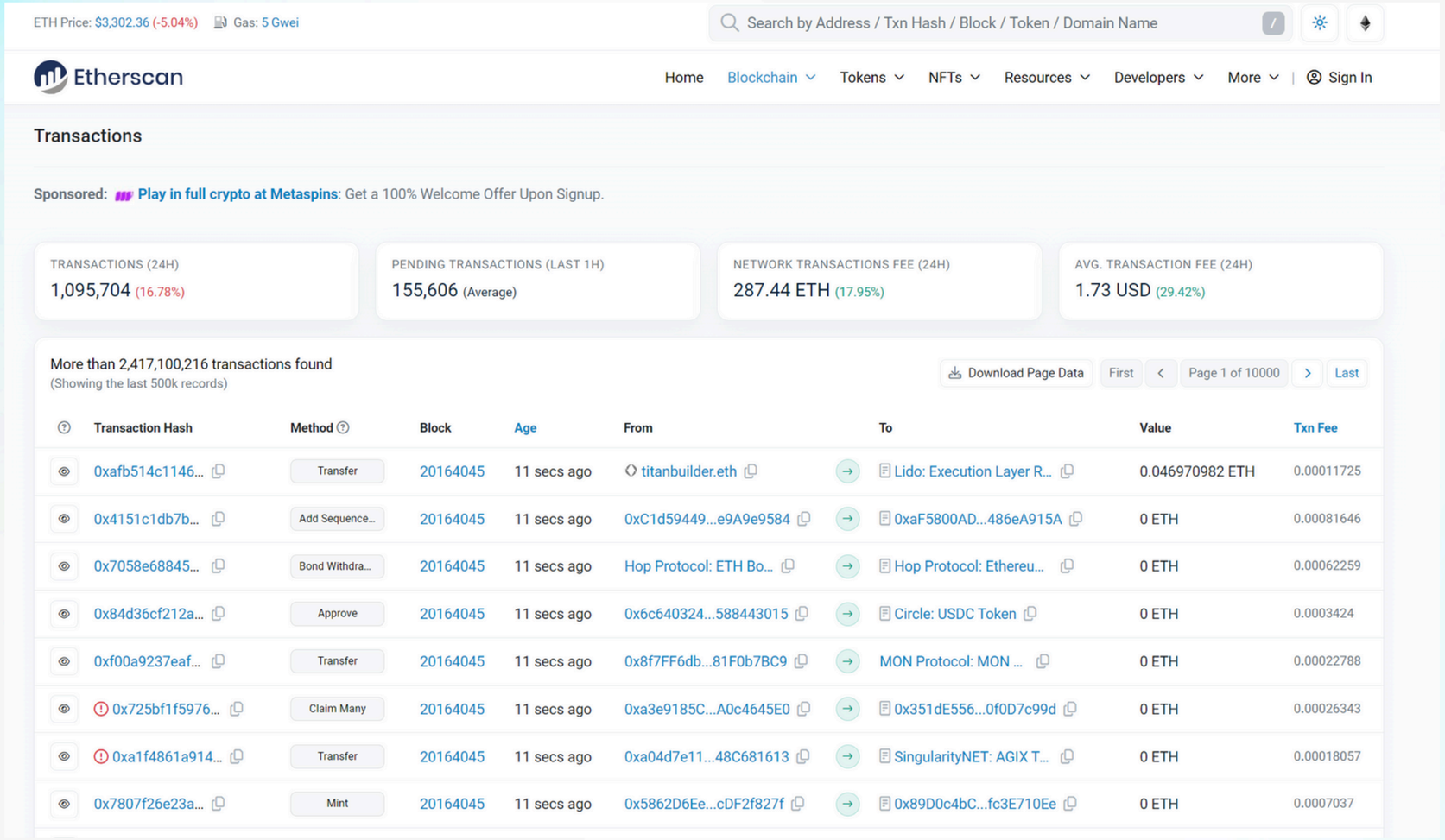
Block-chain


Blockchain
Block N-2
Block N-1
Block N
Block N+1
Block N+2
Block N+3
...
Block N-1
Block N Header
Hash(Block N-2 header)
Hash of (Block N-1 transactions)
Block N-1 Transactions
Hash()
Hash()
Block N
Block N Header
Hash(Block N-1 header)
Hash of (Block N transactions)
Block N Transactions
Hash()

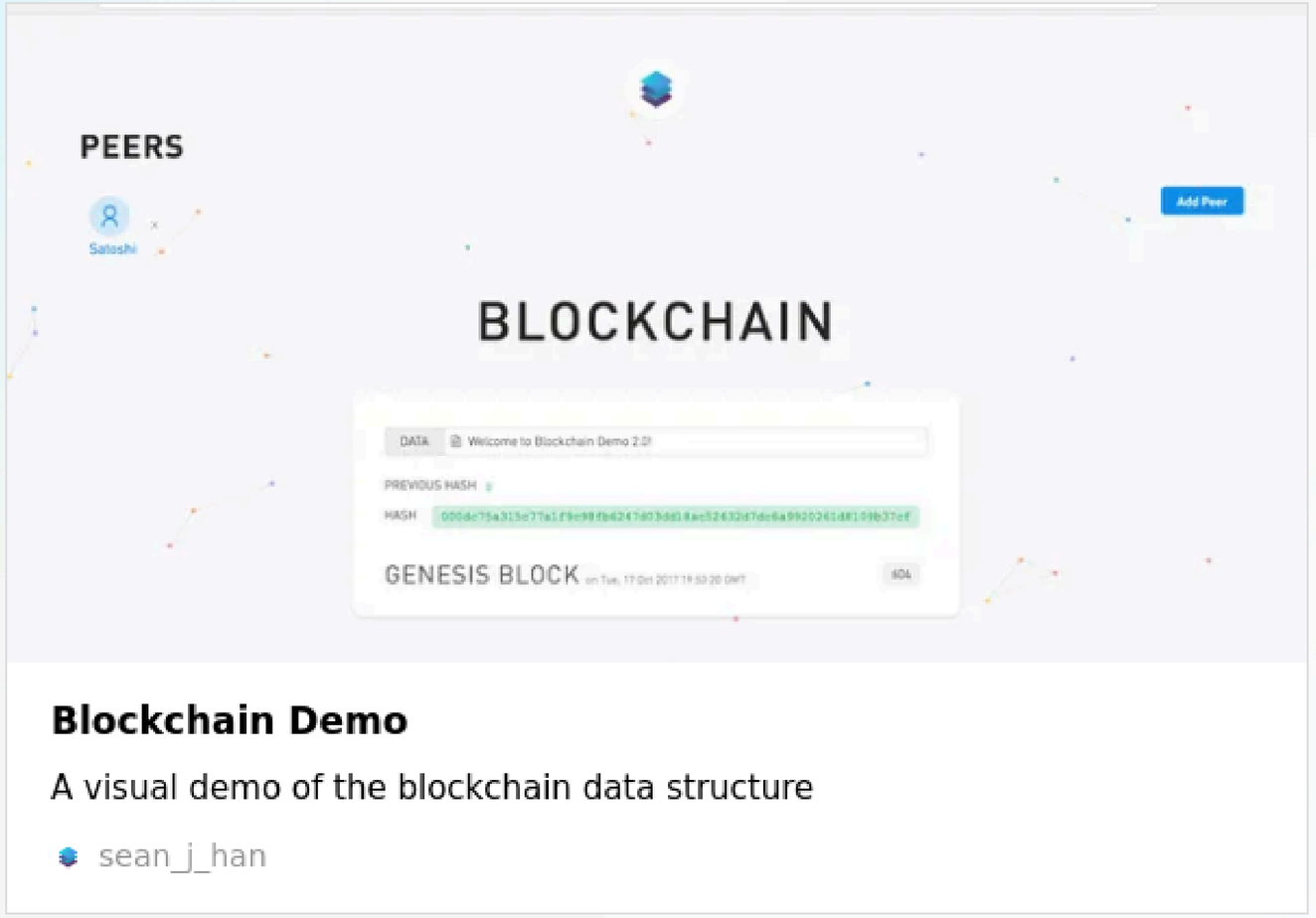
A Block


Block 525252
Block 525253
Timestamp: 27th June, 2018
18:31:00 IST
Transactions: .....
Previous Block reference: 525252
Next block reference: 525254
Validation nonce: Hash(T1T2T3T4)
Block 525254

A Block Explorer



Blockchain Demo



Blockchain Demo

A visual demo of the blockchain data structure

sean_j_han

Wallets: digital tool to store and manage cryptocurrency assets

Basic Cryptography in Wallets

- **Public Key Cryptography**: Ensures secure transactions and asset ownership.
  - **Public Key**:
    - Acts as an address to receive funds.
    - Shared publicly for others to send cryptocurrency to you.
  - **Private Key**:
    - A secret key that grants access to your funds.
    - Must be kept confidential and secure.

How Keys Work

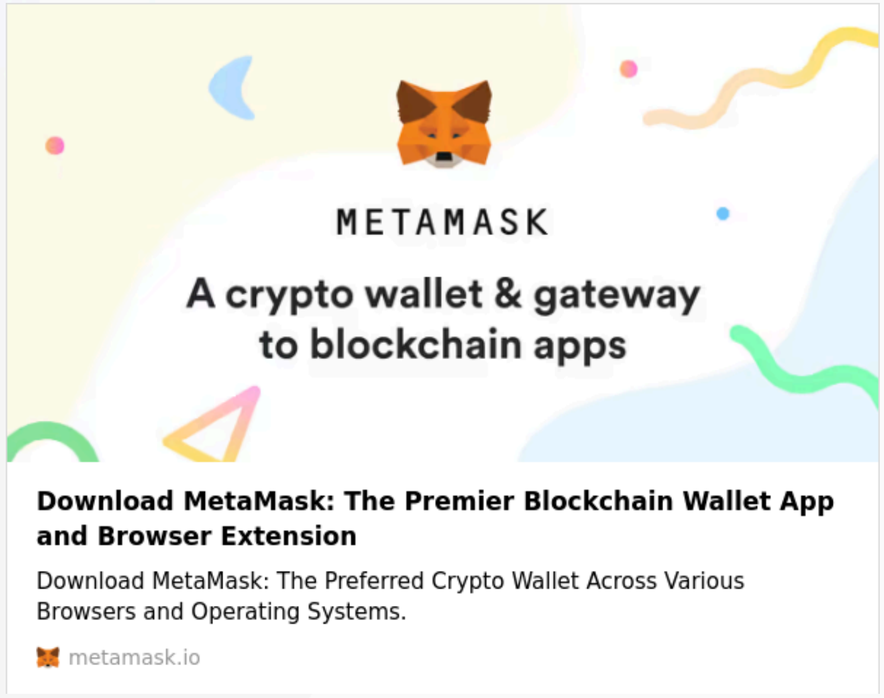
- **Key Generation**:
  - Generated as a pair (public and private).
  - Ensures that only the private key holder can access and manage the funds.
- **Transaction Process**:
  - **Sending Funds**:
    - Sign the transaction with your private key.
    - Broadcast the transaction to the network.
  - **Receiving Funds**:
    - Share your public key (address) with the sender.
    - Funds are sent to your address and recorded on the blockchain.

An externally owned account (EOA) is an account controlled by a cryptographic keypair and is most commonly referred to as a wallet. A key pair consists of a public key (a.k.a public address) and a private key

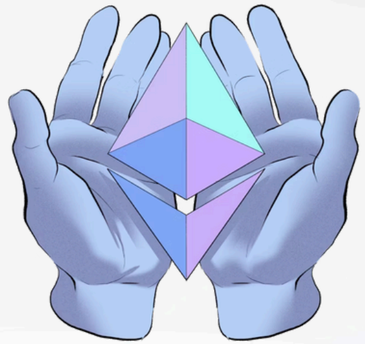
Wallets: digital tool to store and manage cryptocurrency assets

Time for playing with txns!

Hands-on Break



Ethereum and Smart Contracts



An open source, globally decentralized computing infrastructure that executes programs called smart contracts, written in a Turing-complete programming language, translated into bytecode and run on a virtual machine.

Smart contracts written in high-level languages like Solidity are compiled into Ethereum Virtual Machine (EVM) bytecode before deployment. This bytecode is what gets executed on the Ethereum network

Programming language can perform any computation, given enough time and resources

The EVM executes the bytecode in a controlled and consistent manner across all nodes in the Ethereum network

Smart Contracts

Self-executing contract with the terms of the agreement directly written into code. These contracts automatically enforce and execute the terms when predefined conditions are met, without the need for intermediaries

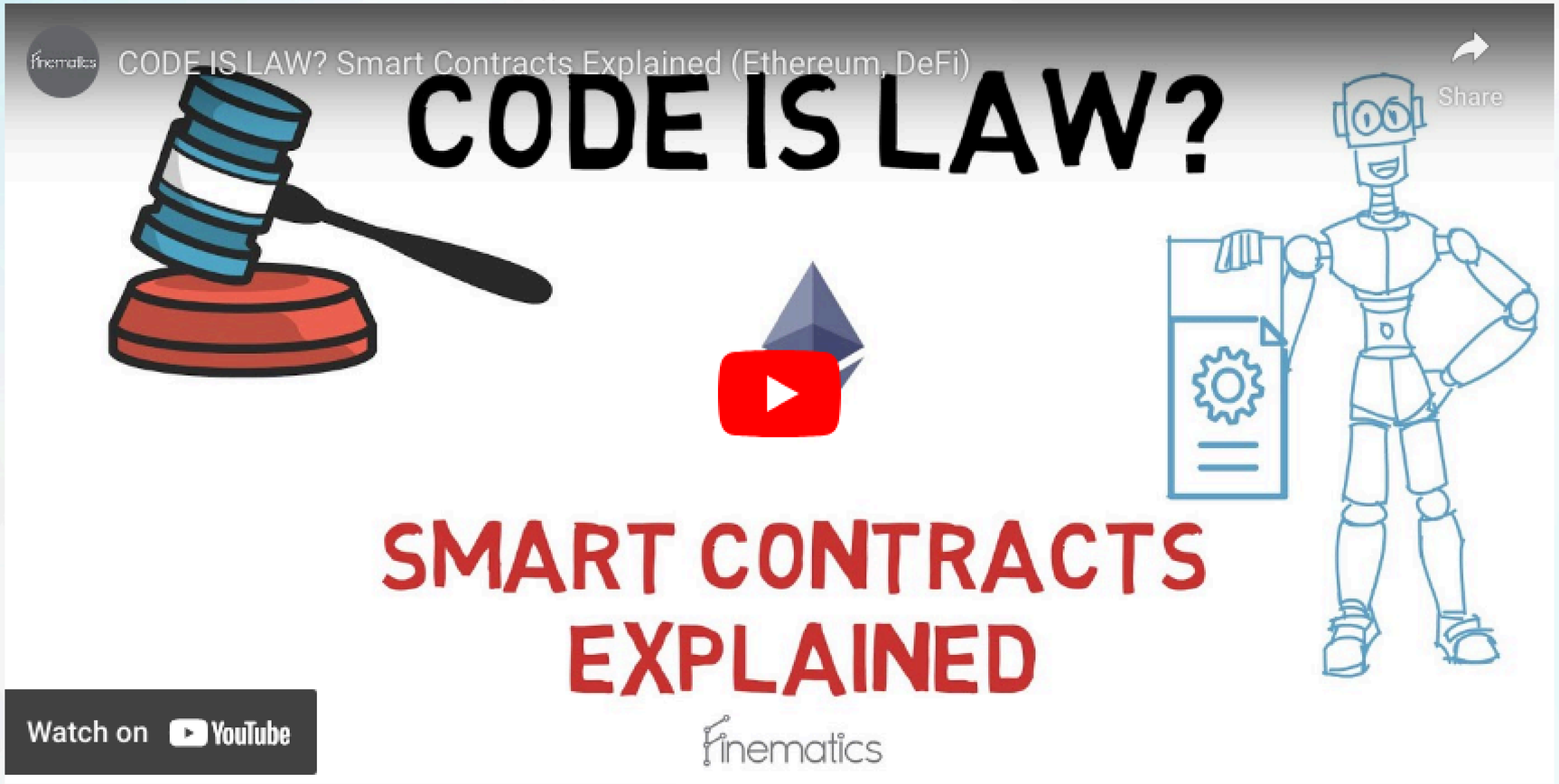
A digital vending machine

A simple metaphor for a smart contract is a vending machine, which works somewhat similarly to a smart contract - specific inputs guarantee predetermined outputs.

- You select a product
- The vending machine displays the price
- You pay the price
- The vending machine verifies that you paid the right amount
- The vending machine gives you your item

The vending machine will only dispense your desired product after all requirements are met. If you don't select a product or insert enough money, the vending machine won't give out your product.

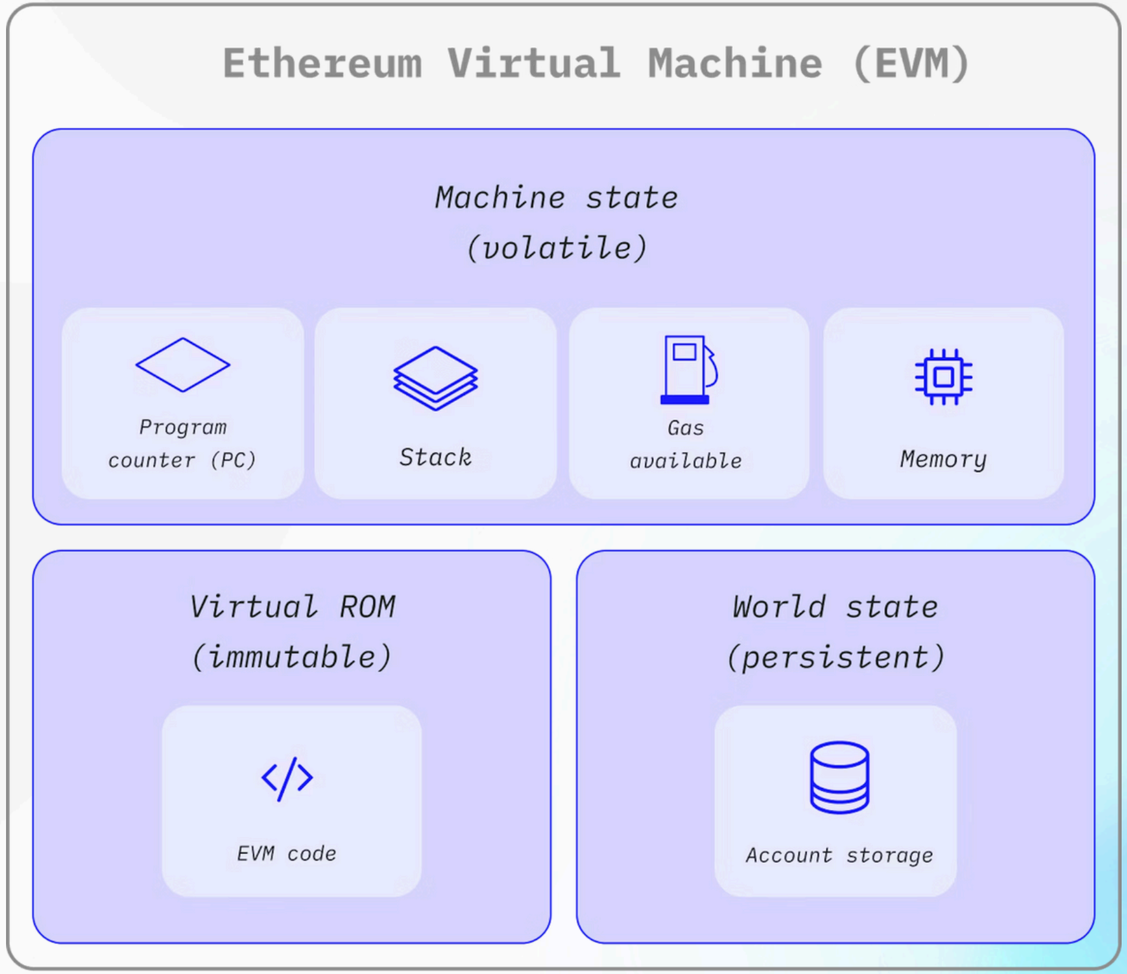
Ethereum and Smart Contracts



EVM

Ethereum is a distributed state machine. Ethereum's state is a large data structure which holds not only all accounts and balances, but a machine state, which can change from block to block according to a pre-defined set of rules, and which can execute arbitrary machine code. The specific rules of changing state from block to block are defined by the EVM.

The Ethereum Virtual Machine (EVM) is a decentralized virtual environment that executes code consistently and securely across all Ethereum nodes. Nodes run the EVM to execute smart contracts, using "gas" to measure the computational effort required for operations.



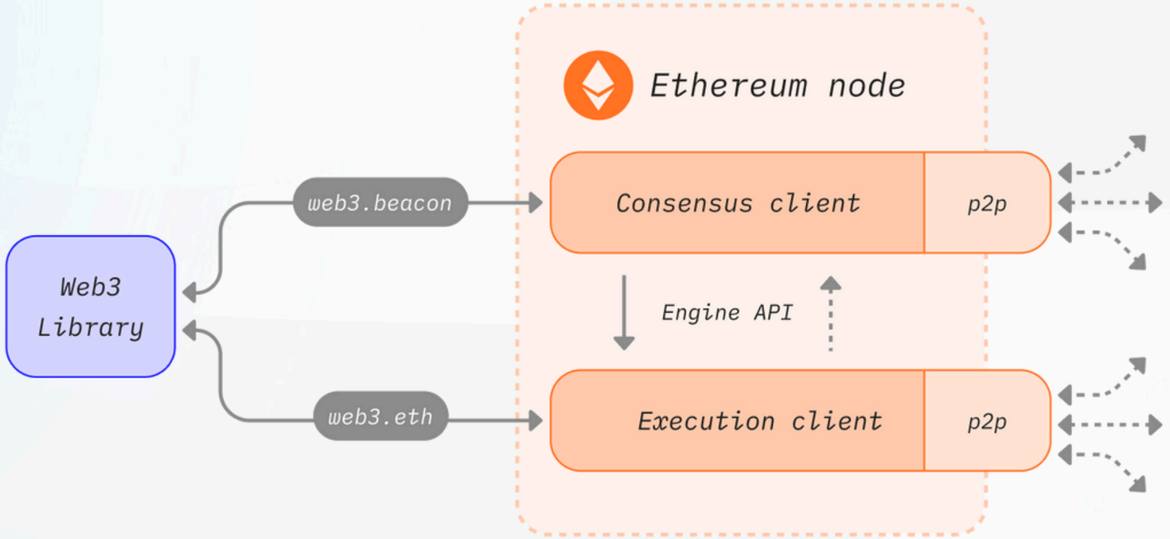
NODES AND CLIENTS

A client is an implementation of Ethereum software that verifies data against the protocol rules and keeps the network secure.

A node is any instance of Ethereum software that to run two clients: a consensus client and an execution client.

Feature	Consensus Client	Execution Client
Primary Role	Participates in consensus mechanism (Proof of Stake)	Manages transaction execution and smart contract operations
Key Functions	Validates blocks and attestations	Executes transactions
	Manages validators	Maintains blockchain state
	Maintains the beacon chain	Executes smart contracts
Examples	Lighthouse, Prysm, Teku, Nimbus	Geth (Go-Ethereum), OpenEthereum, Besu, Nethermind
Network Layer	Consensus Layer	Execution Layer
Interaction	Ensures network reaches consensus	Processes and validates transactions
	Handles validator duties	Interacts with the Ethereum Virtual Machine (EVM)
State Management	Manages the overall consensus state	Manages the blockchain's transactional state
Node Operation	Run by validators to propose and attest blocks	Run by nodes to process transactions and manage blockchain data
User Interaction	Indirect (mainly through staking and validation processes)	Direct (users interact through wallets and dApps)

NODES AND CLIENTS



Interaction with Users

- **End Users**: Typically interact with the Ethereum network through wallets (e.g., MetaMask) and dApps. These wallets and dApps connect to execution clients to send transactions and read blockchain data.
- **Node Operators**: Run both execution and consensus clients to participate fully in the network. For example, an Ethereum node might run Geth (execution client) and Lighthouse (consensus client) together to be a full participant in Ethereum 2.0.

Proof of Stake Consensus

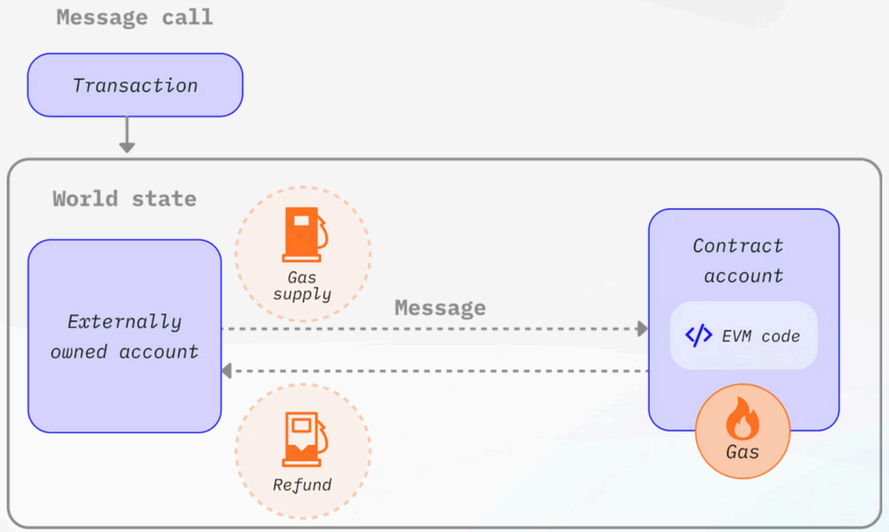
Consensus mechanism used by blockchain networks to achieve distributed consensus.

Basic Concept:

- **Stake: In PoS, validators (also known as "stakers") are chosen to propose and validate blocks based on the number of cryptocurrency tokens they hold and are willing to "stake" as collateral.**

- **Validator Selection: Validators are selected in a pseudo-random manner, weighted by the amount of cryptocurrency they have staked. The more tokens a validator stakes, the higher the chance they have of being selected to propose a new block.**

Gas



Concept	Description	Conversion	Example Use Case
Wei	The smallest unit of Ether (ETH).	1 ETH = 1,000,000,000,000,000,000 wei (10^18 wei)	Used for precise calculations and fees.
Gwei	A commonly used denomination of Ether for gas prices.	1 Gwei = 1,000,000,000 wei (10^9 wei)	Gas prices in transactions.
Gas	A unit of computational work required to execute operations on the Ethereum network.	N/A	Measures the amount of work for transactions and smart contracts.

Gas

https://etherscan.io/gastracker

Empowering Crypto Entrepreneurs

Explore tools like Gas Fees Calculator, discover Web3 Grants, and more for Crypto Entrepreneurs.

Cryptoneur.xyz

Blocknative Gas Fee Estimator for ETH & MATIC - Chrome Web Store

Get into the next block without overpaying with the most reliable and accurate Ethereum and Polygon...

google.com

Solidity and Smart Contracts

Smart Contracts: Self-executing contracts with the terms directly written into code.

Solidity: A high-level programming language for writing smart contracts on the Ethereum blockchain.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract HelloWorld {
    // Storage variable to hold the greeting message
    string public greet = "Hello World!";

    /**
     * @dev Returns the greeting message.
     */
    function sayHelloWorld() public view returns (string memory) {
        return greet;
    }
}
```

Solidity and Smart Contracts

Smart Contracts: Self-executing contracts with the terms directly written into code.

Solidity: A high-level programming language for writing smart contracts on the Ethereum blockchain.

```
// contracts/MyNFT.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import {ERC721} from "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract MyNFT is ERC721 {
    constructor() ERC721("MyNFT", "MNFT") {
    }
```

Feature	Description
Community Support	Solidity has a large, accessible community. As the first smart contract programming language developed for the Ethereum network, it enjoys wide community support, making it easier for new developers to get help.
Turing-Complete	Solidity is Turing-complete, meaning it can compute all computable functions and is not limited to running just a handful of algorithms.
Modern Programming Concepts	Solidity offers features found in most modern programming languages, including functions, string manipulation, classes, variables, arithmetic operations, and supports mapping data structures that act as hash tables with key types and key-value pairs.
Ease of Learning	Solidity doesn't have a steep learning curve for those familiar with popular programming languages like Python, C++, and JavaScript, as it borrows much of its syntax from these languages.

Solidity

Concept	Description
Contracts	Basic building blocks of Solidity. They contain code and data (state) that resides at a specific address on Ethereum.
State Variables	Variables that hold data permanently in contract storage.
Functions	Executable units of code within contracts. Functions can modify the state or perform computations.
Mappings	Key-value data structures, similar to hash tables or dictionaries, used for efficient storage and retrieval of data.
Structs	Custom data types that allow grouping of related variables.
Arrays	Ordered collections of elements of the same type. Can be fixed-size or dynamic.
Modifiers	Used to change the behavior of functions in a declarative way (e.g., restricting access).
Events	Mechanism for logging that allows contracts to communicate with external consumers via the Ethereum log system.
Inheritance	Enables contracts to inherit properties and methods from other contracts, promoting code reuse.
Visibility Specifiers	Keywords that define who can call functions or access state variables (e.g., public, private, internal, external).
Require Statement	Used to enforce conditions. If a condition is not met, the transaction is reverted.
Memory vs Storage	Differentiates between temporary (memory) and permanent (storage) data storage locations.
SafeMath Library	Used to prevent overflows and underflows in arithmetic operations.
Constructor Functions	Special functions executed once during contract deployment, used to initialize state variables.
Mappings and Structs in Arrays	Combining mappings and structs within arrays for complex data structures.

Solidity: Memory and ABI

Memory is used for temporary data within functions, whereas storage is for persistent data on the blockchain.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleExample {
    // State variable (stored in storage)
    uint256 public storageData;

    // Function to set the storage data
    function setStorageData(uint256 _value) public {
        storageData = _value;
    }

    // Function to demonstrate memory usage
    function useMemory(uint256 _value) public pure returns (uint256) {
        // Local variable (stored in memory)
        uint256 memoryData = _value;
        memoryData *= 2; // Modify memory data
        return memoryData; // Return modified memory data
    }
}
```

Concept	Description	Details
Memory in Solidity		
Storage	Persistent data storage on the blockchain.	Contract-level variables are stored here.
Memory	Temporary storage used during function execution.	Use the `memory` keyword for temporary data within functions.
Stack	Used for small local variables and function arguments.	Extremely limited in size (1024 elements).
Bits and Bytes		
Bit	The smallest unit of data (0 or 1).	
Byte	A group of 8 bits.	
uint8, uint16, ... uint256	Unsigned integers (8 to 256 bits).	Examples: `uint8` (0 to 255), `uint256` (0 to 2^256-1).
int8, int16, ... int256	Signed integers (8 to 256 bits).	Examples: `int8` (-128 to 127), `int256` (-2^255 to 2^255-1).
bytes1, bytes2, ... bytes32	Fixed-size byte arrays.	Examples: `bytes1` (1 byte), `bytes32` (32 bytes).
bytes	Dynamically-sized byte array.	Length can vary during execution.
address	20-byte Ethereum address.	Typically used to store wallet or contract addresses.

Solidity: Memory and ABI

The Application Binary Interface (ABI) in Solidity defines the standard for encoding and decoding data for interacting with smart contracts, ensuring consistent communication and execution across different Ethereum platforms and applications.

Concept	Description	Details
Application Binary Interface (ABI)		
ABI Overview	Standard way to interact with contracts in the Ethereum ecosystem.	Defines data encoding/decoding for contract calls and transactions.
ABI Encoding	Function signatures hashed using Keccak-256. Parameters encoded based on type.	Fixed-size types have fixed positions; dynamic-size types have offset table.
ABI Decoding	Reverse process of ABI encoding.	Extract and decode data based on the function signature.

REMIX IDE

An open-source integrated development environment designed for Solidity smart contract development.

https://remix.ethereum.org

Key Features

- Solidity Support:

Write, compile, and debug Solidity smart contracts.

- User-Friendly Interface:

Intuitive and accessible for all skill levels.

- Browser-Based:

No installation needed, runs directly in a web browser.

- Integrated Tools:

Includes code editor, Solidity compiler, and debugger.

- Testing and Deployment:

Easily test and deploy contracts on local, testnet, and mainnet networks.

- Plugins and Extensibility:

Modular architecture with plugins for added functionality.

Hands-On with CryptoZombies

#1 Solidity Tutorial & Ethereum Blockchain Programming Course

CryptoZombies is The Most Popular, Interactive Solidity Tutorial That Will Help You Learn Blockchain Programming by Building Your Own Fun Game with Zombies — Master Blockchain Development with Web3, Infura,...

cryptozombies.io

END OF PART ONE :)

Connect with the Devs Here

Join Us on Telegram

@metis_dev

METIS

JOIN THE WEB3 ECONOMY

metis.io