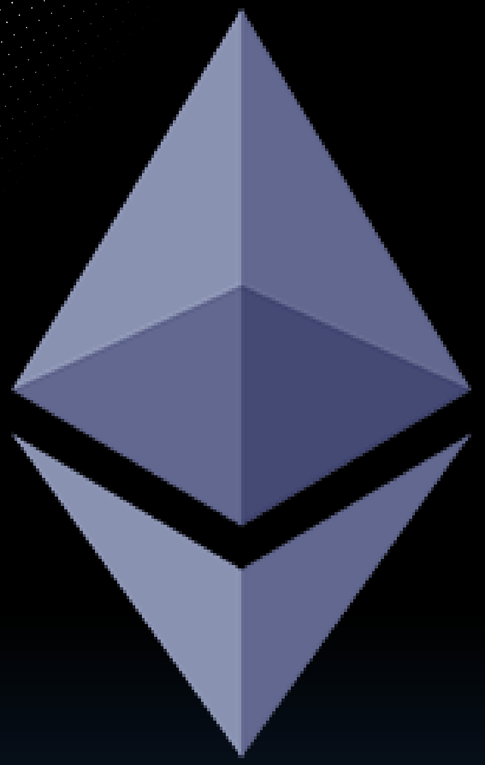
IN5420 – Distributed Blockchain Technologies



Ethereum

Ivan Shirshikov

EXECUTIVE SUMMARY

01

Introduction

The idea behind Ethereum

02

Smart Contracts

Smart Contracts and the Ethereum Software Model with relevant examples.

03

Data model and storage structure

Ethereum has its own accounting model and unique data storage system, which distinguishes it from Bitcoin.

04

Consensus and the Ethereum Network

To maintain the integrity of the network, Ethereum uses unique consensus algorithms and blockchain optimizations.

05

Transition to Proof-of-Stake

Ethereum has switched to Proof-of-Stake, a greener and more sustainable mechanism where security is provided by pledging ETH.

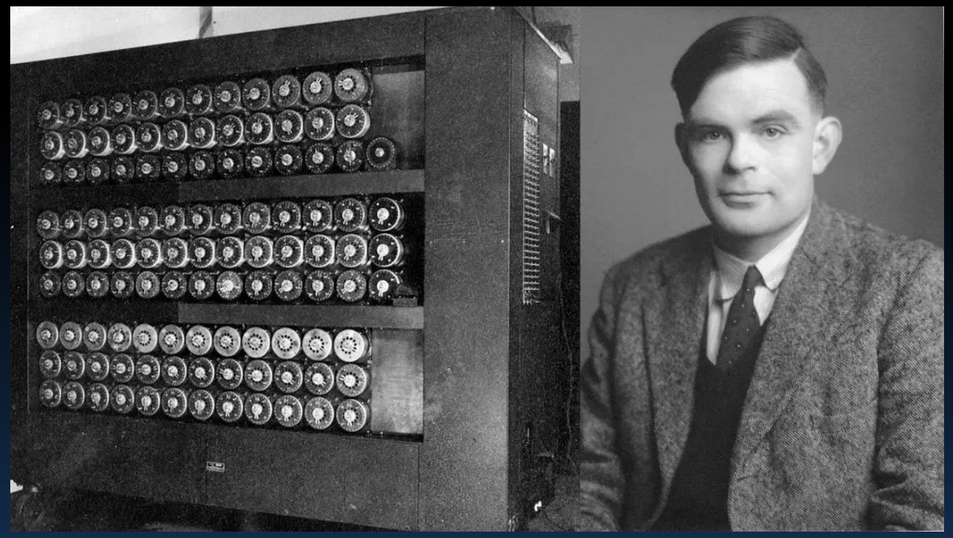
06

Scaling and the Future of Ethereum

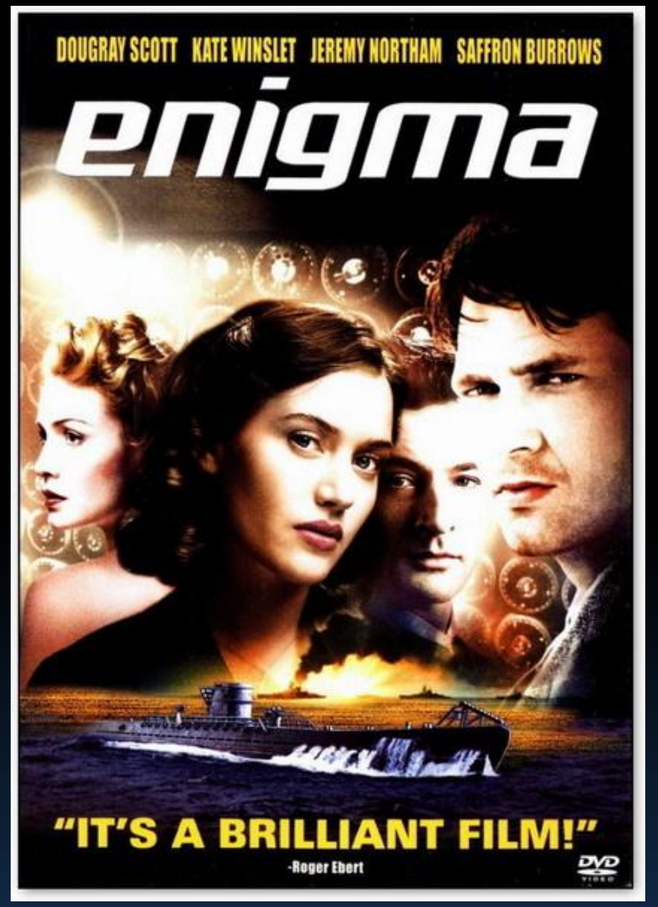
Ethereum is evolving towards scaling to become an accessible platform for everyone. This part will also contain possible future problems.

The idea behind Ethereum

- A Turing Complete system means a system in which a program can be written that will find an answer (although with no guarantees regarding runtime or memory).
- Ethereum is cryptocurrency that are built for any application



Alan Turing and his Turing machine



Film representation of Turing Complete idea

What is Ethereum?

• Ethereum is an ambitious altcoin designed to provide a Turing-complete programming language for building decentralized applications (dApps) and smart contracts

• It acts as a general-purpose decentralized computer, where smart contracts can manage digital assets and complex workflows autonomously.

•Described in late 2013 by Vitalik Buterin, launched as Frontier in 2015. Funded via Bitcoin-based pre-sale (raised funds for Ethereum Foundation).

• Other projects that tried to realise same idea: Counterparty, Nxt, BitShares



Why Ethereum is the most noticable?

- Ethereum has the largest developer ecosystem in the blockchain space.

- Backed by significant venture capital and enterprise support (Consensys, Ethereum Foundation, etc.).

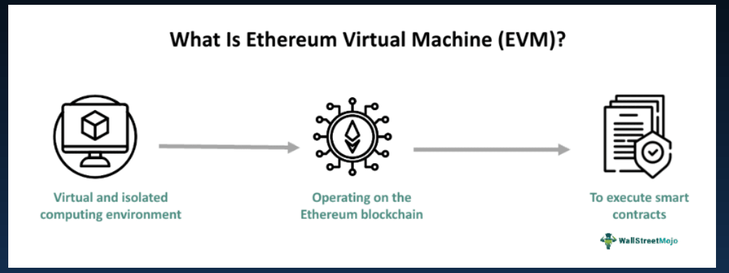
- Several novel technical ideas as EVM, Solidity, gas etc.

- Funded via **Bitcoin-based pre-sale** (raised funds for Ethereum Foundation).



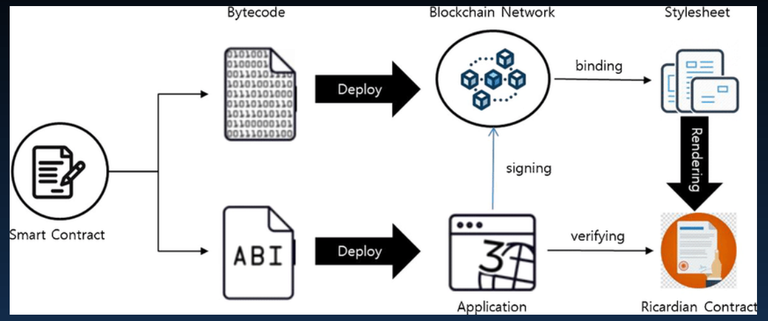


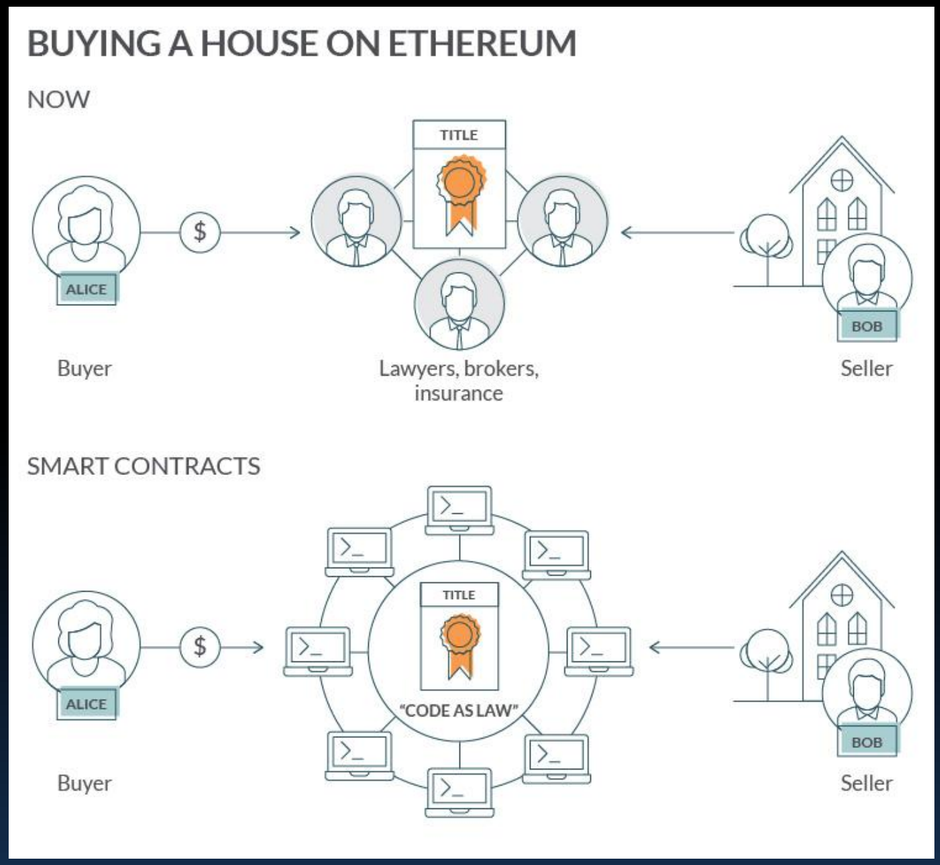
ethereum



Smart Contract Programming Model

- Smart Contract = Program on the Blockchain
- Executed by the EVM
- Autonomous & Persistent
- Owns and Manages Funds
- Public API





A simple example: Namecoin in Ethereum.

```
contract NameRegistry {
      mapping(bytes32 => address) public registryTable;
      function claimName(bytes32 name) {
            if (msg.value < 10) {
                  throw;
            }
            if (registryTable[name] == 0) {
                  registryTable[name] = msg.sender;
            }
      }
}
```

- You can build simple services like DNS or name systems directly in Ethereum with just a few lines of Solidity.

- Like Namecoin, this contract creates a basic **name registration system**, but fully inside Ethereum.



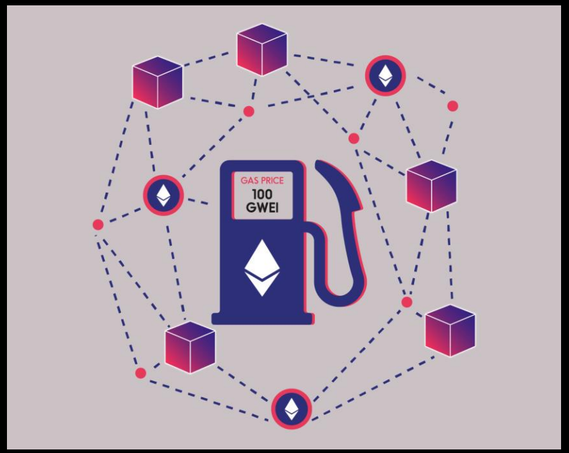
Gas. The first way to Limit and charge for calculations

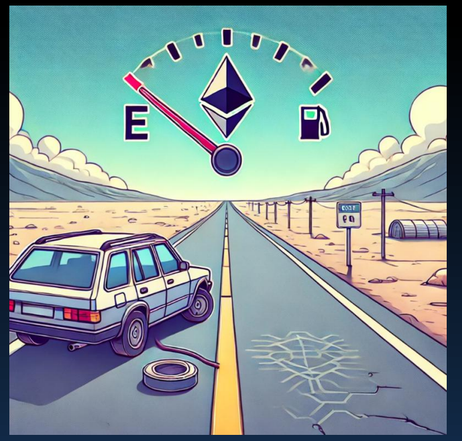
• Gas = Execution Fee for each operation

• Examples of Gas Costs: Simple math (add, compare) = 1 gas, SHA-3 hash = 20 gas, Writing to storage = 100 gas, Sending a transaction = 21,000 gas (base fee)

• In Ethereum, if you run out of gas mid-execution, it’s like your car stalling halfway to the destination — you lose your fuel, and you don’t reach your destination

• What limitations in building apps on Ethereum we have? No big computations





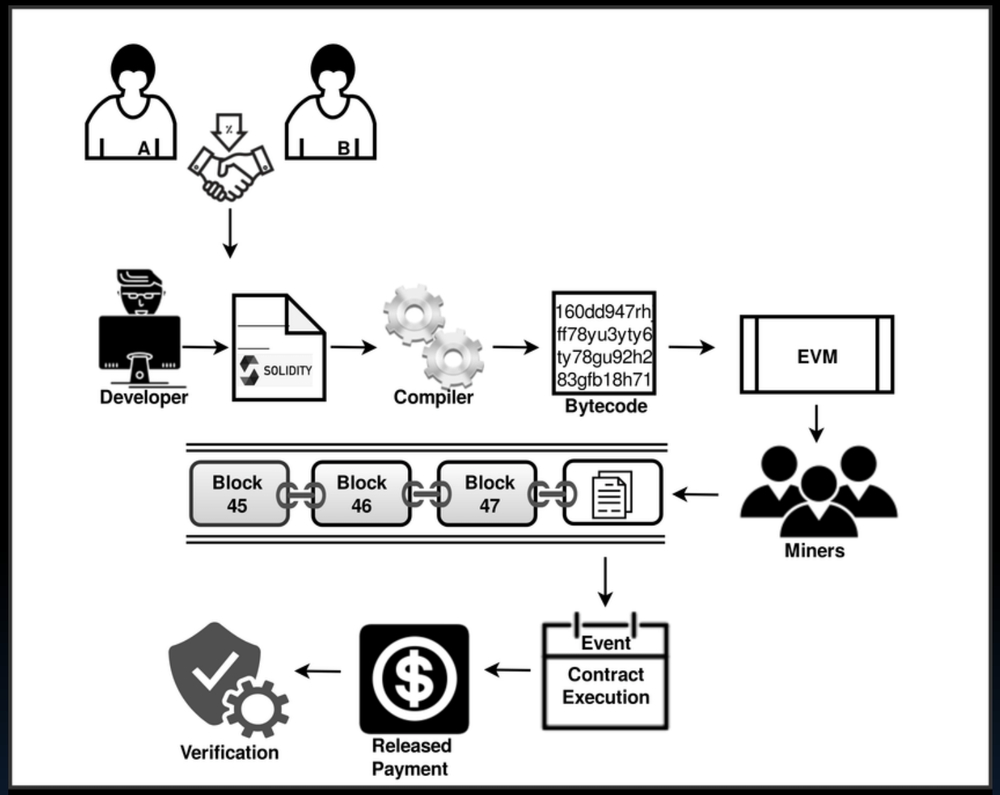
Security and Incentive Concerns.

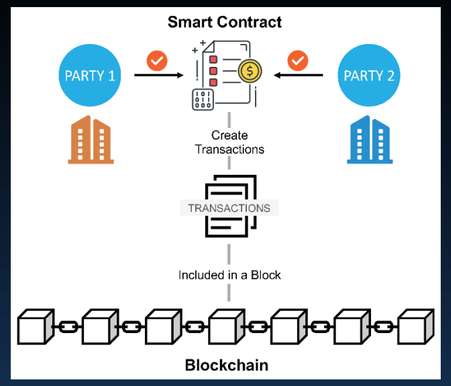
- Higher Complexity
- Less Maturity
- Gas Fees Issue. Gas is paid only to the block proposer
- Validation Risks



Real World Chess example.

- Alice and Bob want to bet on a chess game, but don't trust each other.
- Bob reviews the contract and deposits his matching stake.
- No one can withdraw funds early or cheat the system.
- Key Idea: Ethereum enables trustless games where the rules and funds are enforced directly by the blockchain.





Who plays white? Randomization problem

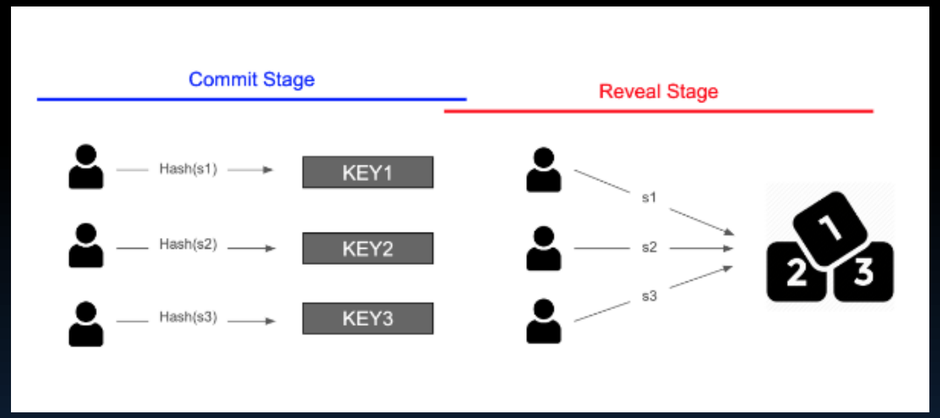
- Ethereum (and other blockchains) do not have a built-in mechanism for generating random numbers, because of determinism of blockhain

- You cannot use a random() function like in other programming languages

- Two ideas to solve it: Block hash randomness

- Commit-reveal scheme

if (block.timestamp % 2 == 0) { Alice plays white } else { Bob plays white }




```
AliceNumber = 1010  (in binary = 10 in decimal)
BobNumber   = 0110  (in binary = 6 in decimal)

XOR:
1 0 1 0
0 1 1 0
---------
1 1 0 0 = 12 (in decimal)
```

Beyond Games: Financial Applications of Ethereum.

Ethereum enables powerful financial applications:

- Escrow Payments — trustless, conditional payments.
- Prediction Markets — decentralized betting on future events.
- Smart Property — programmable ownership of assets (real estate, IP, etc.).
- Micropayment Channels — fast and cheap off-chain transactions.
- Mixing Services — enhance privacy in transactions.
- Decentralized Auctions & Exchanges — on-chain marketplaces.

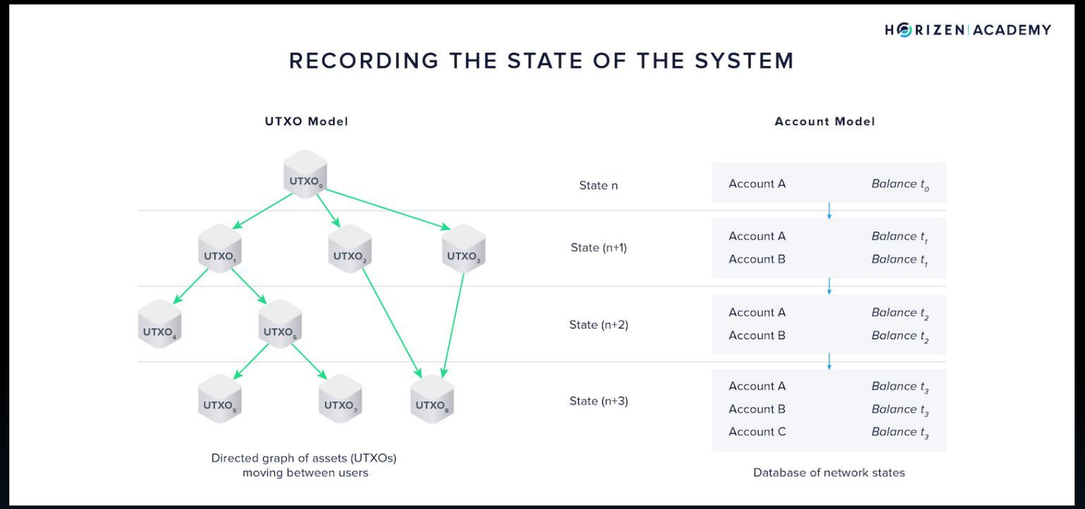






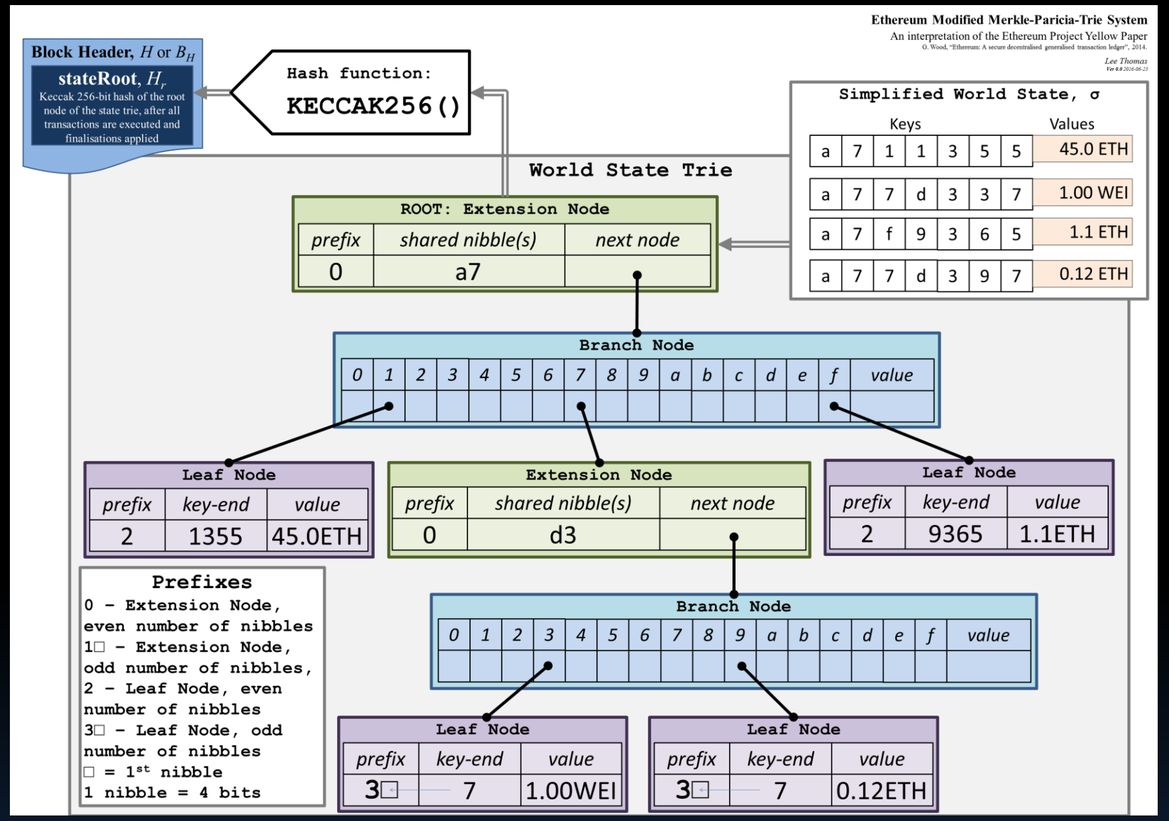
State and account balances in Ethereum.

- Stores balances directly in accounts (like a bank).
- Each account/address has a balance and state.
- Supports both user accounts and contract accounts.
- Simpler for smart contracts and interactions between contracts.



Merkle Patricia tree

- Combines Merkle Tree + Prefix Tree (Trie).
- Efficient proofs of account/contract state.
- Fast lookups and updates.
- Keys = account addresses (usually 160-bit Ethereum addresses)
- Values = account and contract states.



What is stored in the leaves?

- For regular (EOA) accounts: Balance, Nonce and CodeHash
- For contracts: All the same (Balance, Nonce, CodeHash), but also: a Storage root
- Structure of the Transaction:

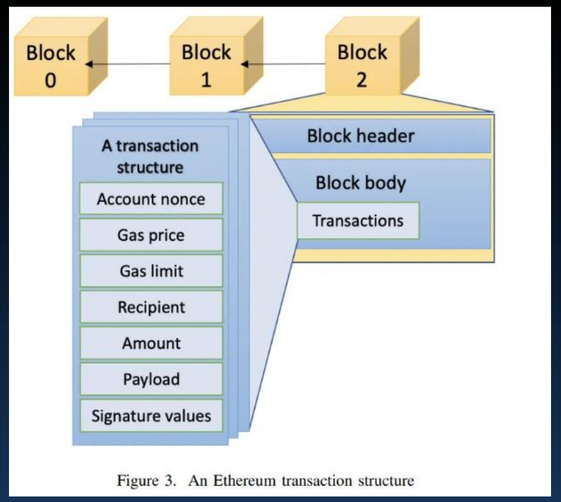
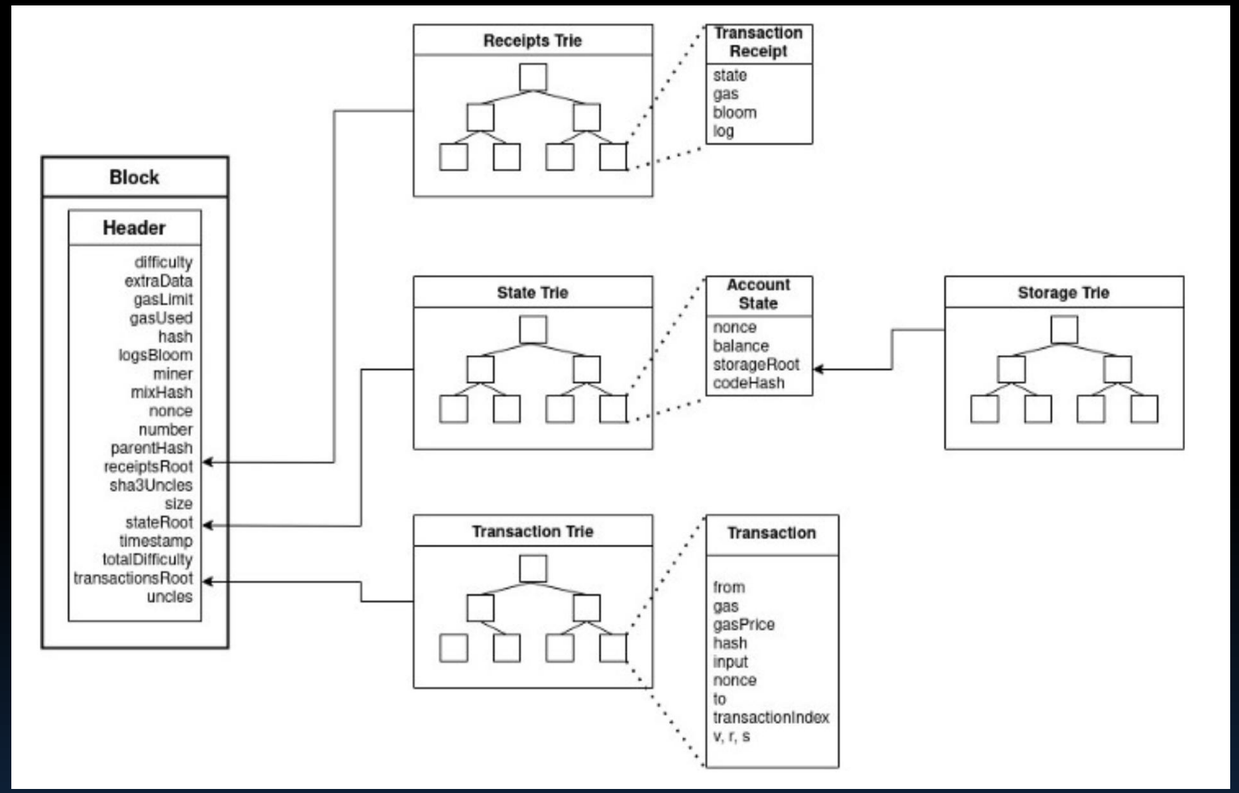


Figure 3. An Ethereum transaction structure



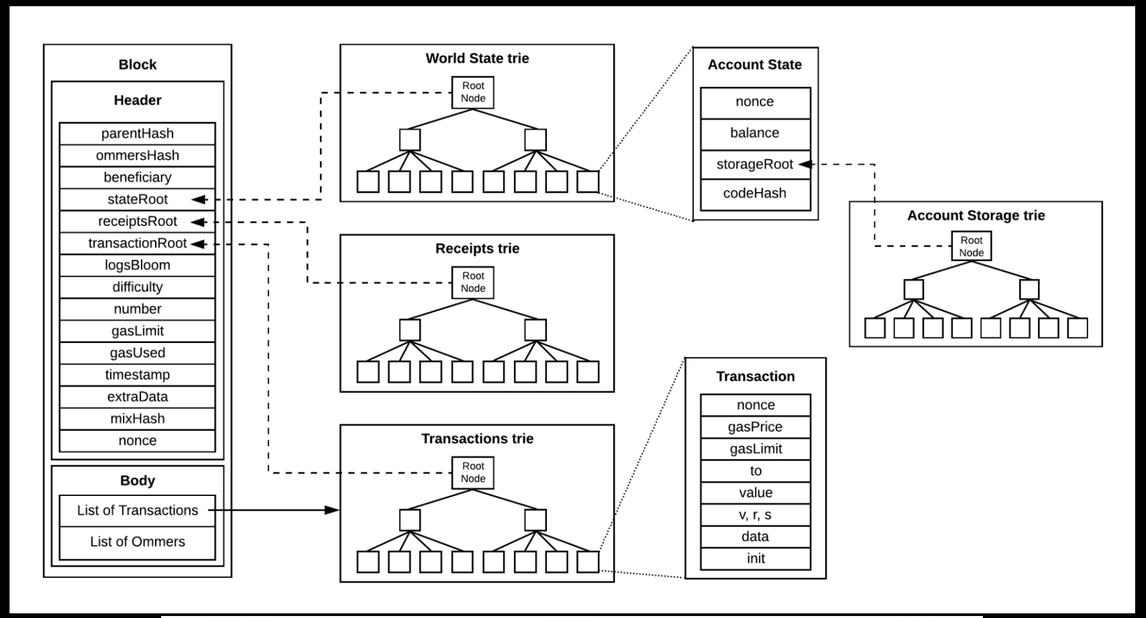
MPT vs Merkle Tree

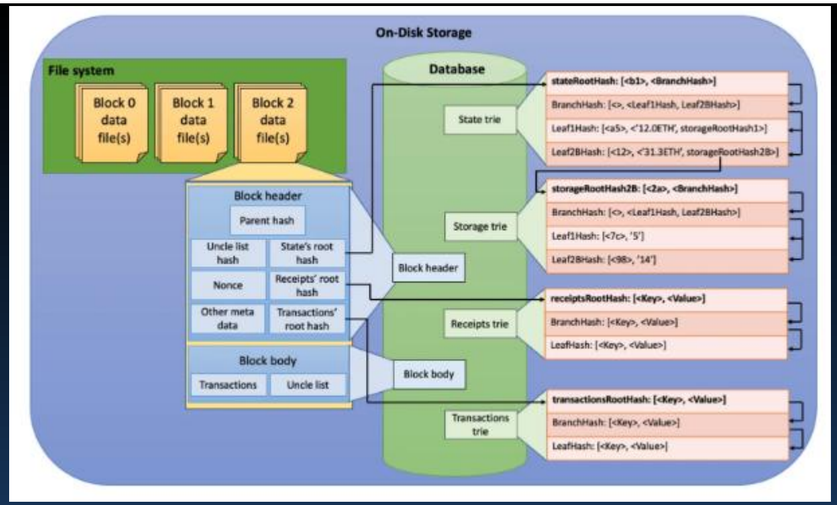
- **Merkle Tree** = Proof of inclusion for transactions.
- **MPT** = Combines proof of inclusion **and** dynamic state updates for accounts/contracts.

Feature	Merkle Tree (Bitcoin)	Merkle Patricia Trie (Ethereum)
Structure Type	Binary Tree (2 children per node)	Prefix Tree (Trie) + Merkle Tree (up to 16 branches)
Purpose	Stores transactions inside a single block	Stores global account and contract states
Leaf Data	Only transaction hashes	Full account data (balance, nonce, codeHash, etc.)
Path Representation	Binary left/right path	Hexadecimal prefixes (0-9, a-f)
Updates	Built anew for each block	Persistent across blocks, supports fast updates
Lookup Efficiency	Used only for transaction proofs	Fast lookups, inserts, and deletes
Block Usage	Merkle root of transactions stored in block header	Multiple roots (state, storage, receipts, txs) stored
Main Use Case	Merkle proof for transaction inclusion	Merkle proof + dynamic state management

Data storage structure

- All states is laying in levelDB on each validator.
- State trie — global account state.
- Storage trie — contract data.
- Transactions trie — block transactions themselves.
- Recipts trie — transaction execution results.

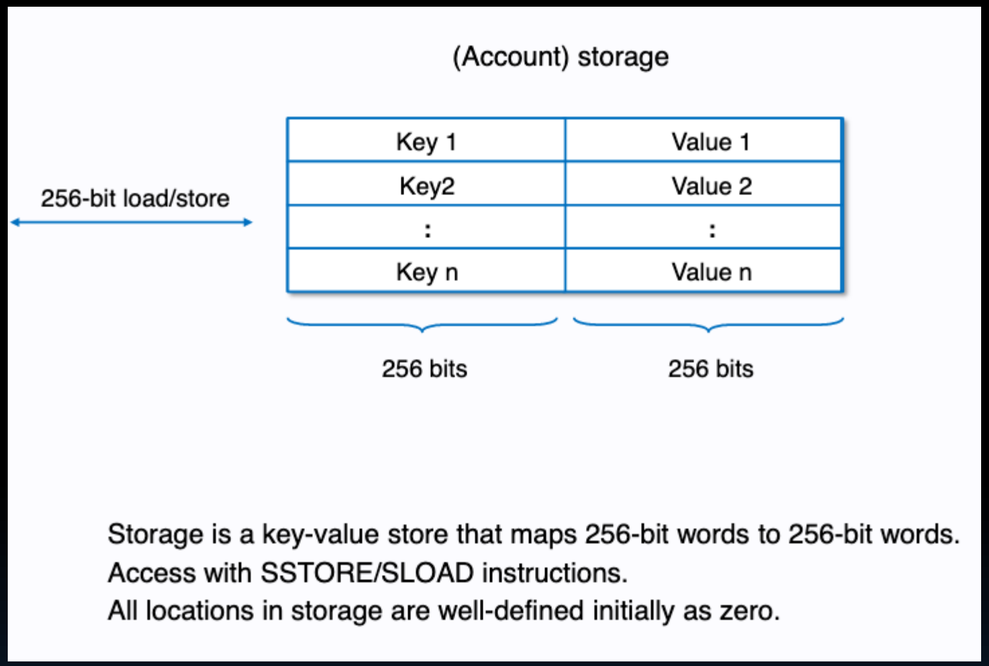




Contract Data Storage in Ethereum

- Maps 256-bit keys ➞ 256-bit values (theoretical space: 2^{264} bytes or 1.7×10^{71} GB). More data - more payment and you can not run into limit of memory, but in economical

- How not to run into limit?

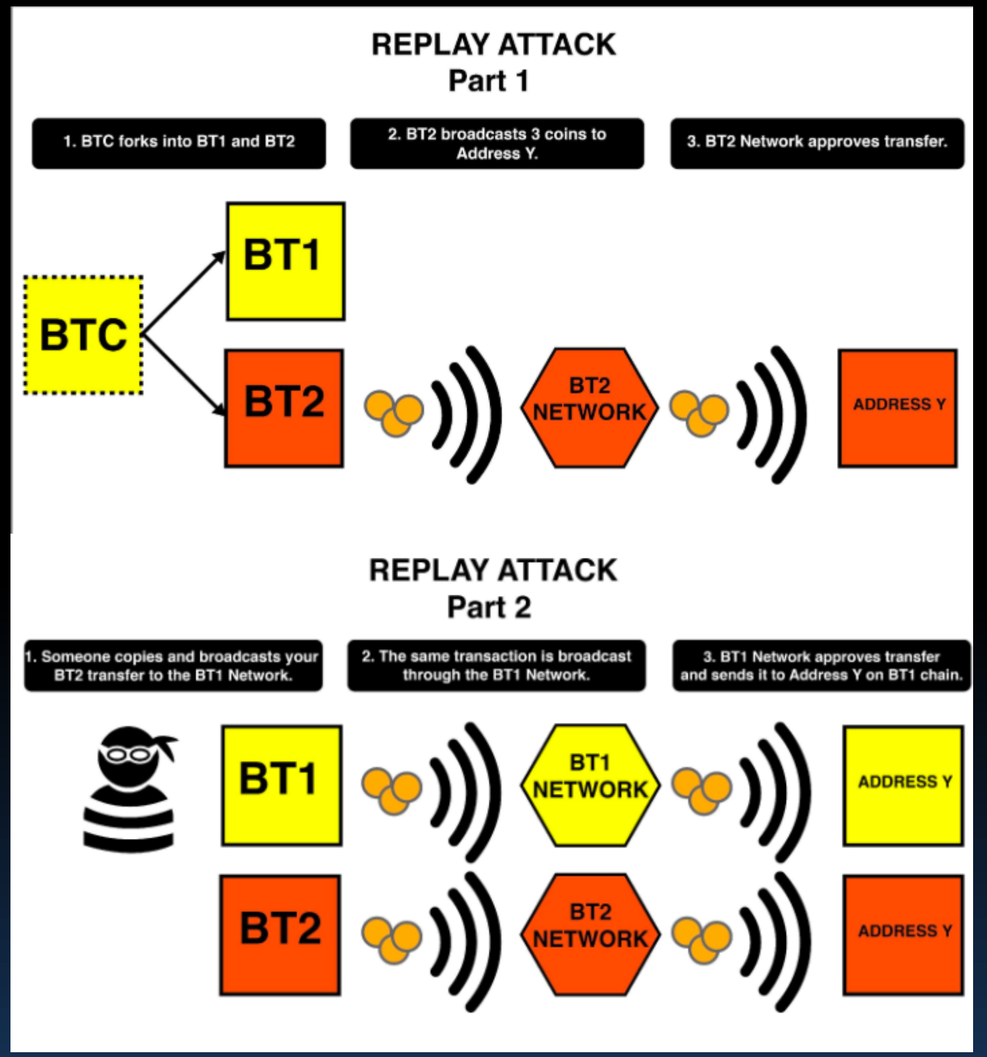


Data Retention Policy in Ethereum

- Permanent Ledger
- State Trie Pruning
- Pruning Window. Delete after 5000 new blocks (~20 hours).
- Goal is to reduce disk space usage while still allowing recent historical queries.

Replay protection

- Each account has a transaction counter (nonce).
- Prevents replaying the same transaction twice.
- “I authorize my nth transaction to be a payment of 1 ether to Bob.”



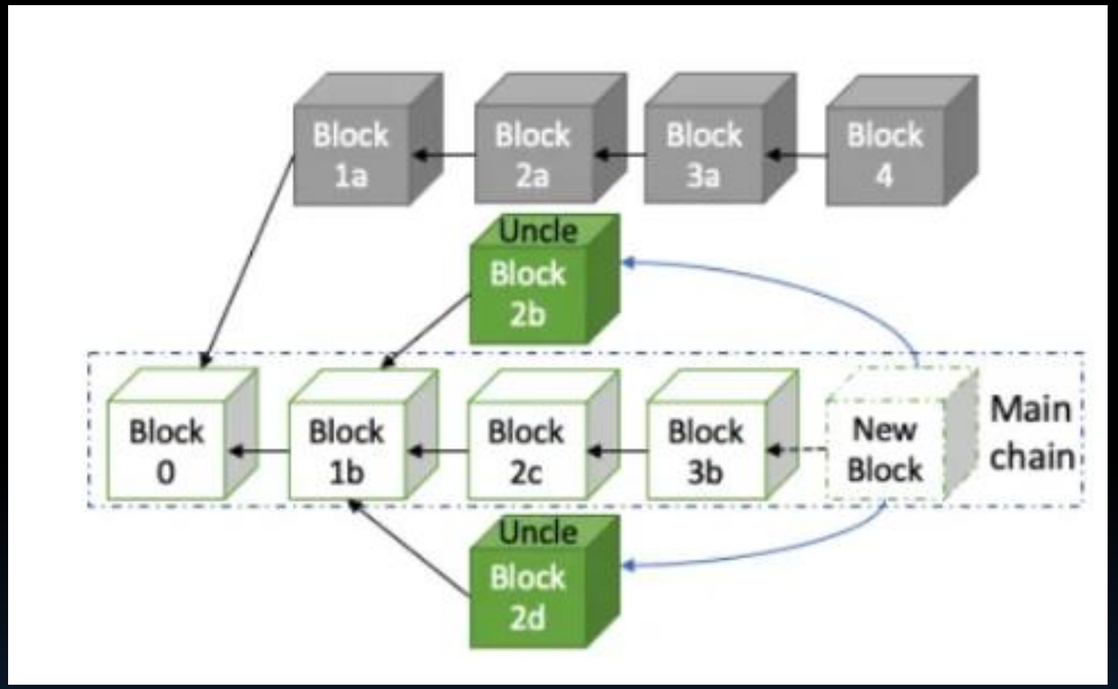
Conflict resolution in Ethereum

• Ethereum uses the **GHOST (Greedy Heaviest Observed Subtree) protocol** to resolve blockchain forks efficiently.

• Chooses the branch with the **heaviest subtree** (most descendant blocks, including uncles).

• GHOST improves Ethereum’s fork resolution by **counting uncles** as part of the chain's total weight, unlike Bitcoin’s simple "longest chain" rule.

• Miners of uncle blocks receive partial rewards.



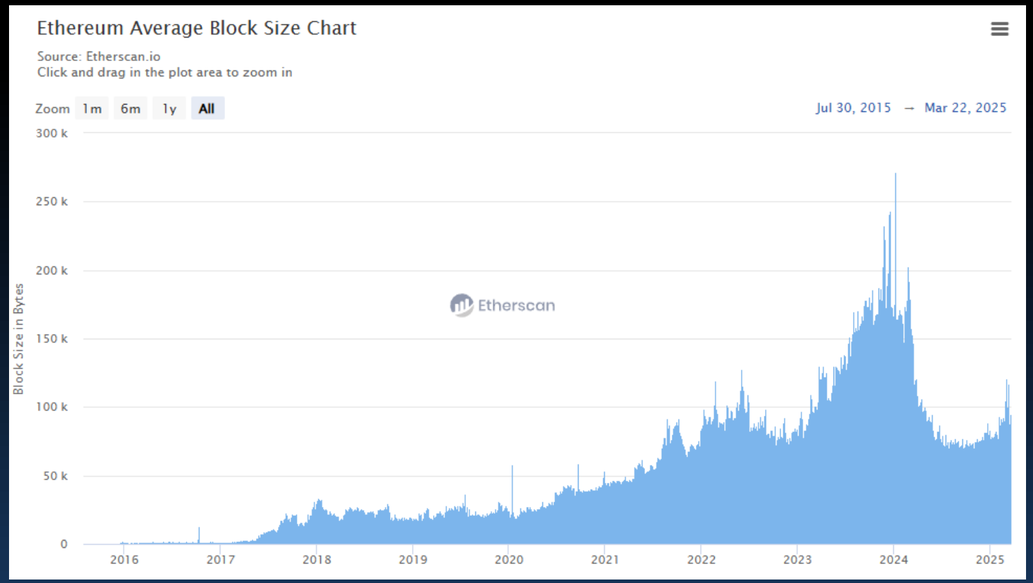
Ethereum block size

• Each new block is created by a miner (before PoS) or a validator (in PoS), and this block creator can change the gas limit within the allowed range.

• Raising or lowering the gas limit by no more than 1/1024 (~0.097%) of the previous value per block.

• This allows the market itself to determine the optimal gas limit

• Validators decision is based on network load and Technical limitations of nodes



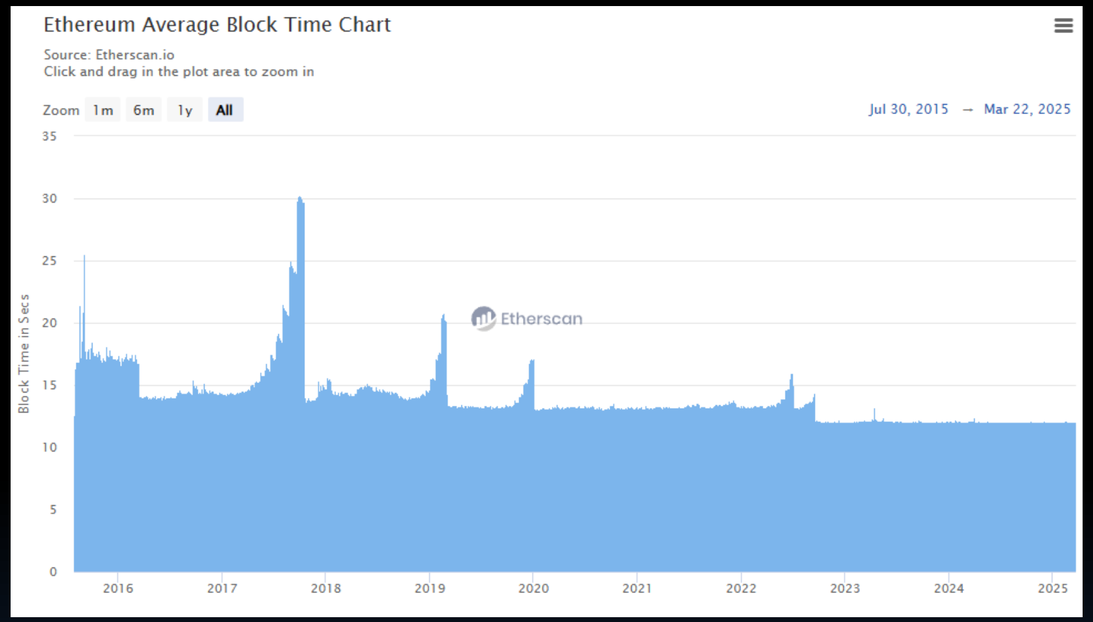
Ethereum block time

- The spikes in Ethereum block time are caused by the built-in Difficulty Bomb.

- A mechanism that exponentially increases mining difficulty over time.

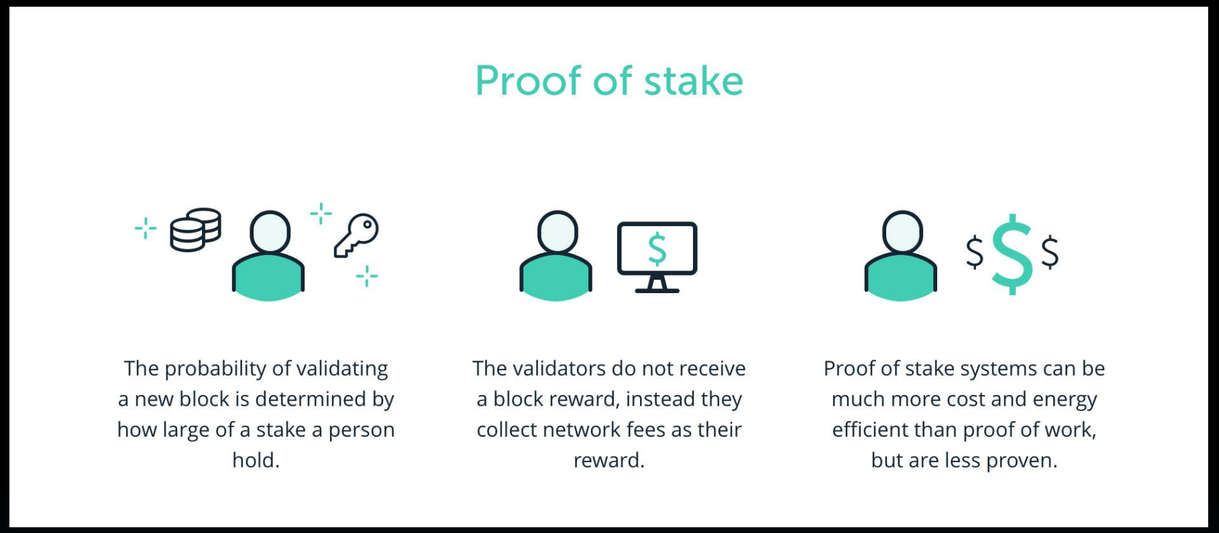
- After major upgrades (e.g., Byzantium, Constantinople, London, The Merge), the bomb is delayed (defused) via hard forks.

- Since Ethereum switched to Proof-of-Stake (2022), the Difficulty Bomb has been deactivated, resulting in stable block times (~12 seconds).



Proof of Stake

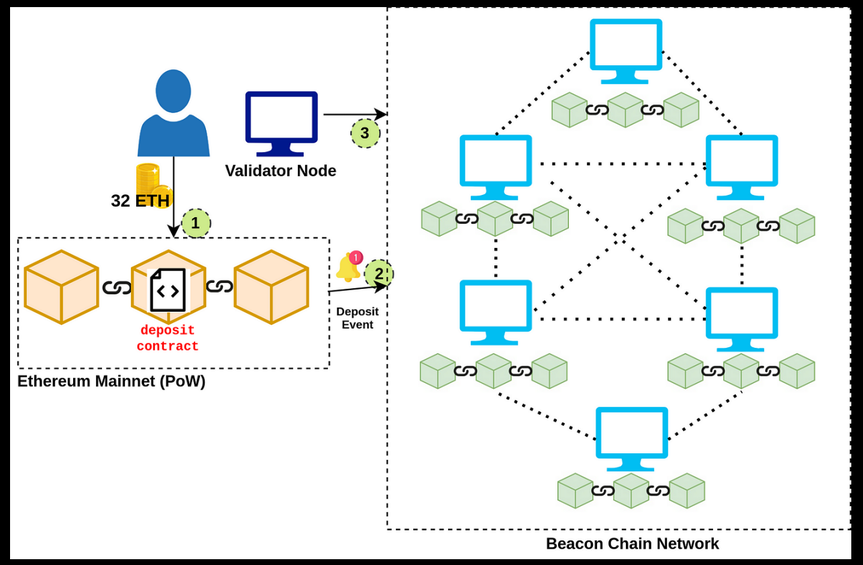
- PoS is a consensus mechanism where validators lock up assets to secure the network.
- In Ethereum, validators stake ETH into a smart contract.
- Verify new blocks. Create & propagate blocks occasionally.
- Misbehavior (e.g., double-signing, sending conflicting data) can result in loss of staked ETH (slashing). Incentivizes honesty and network security.

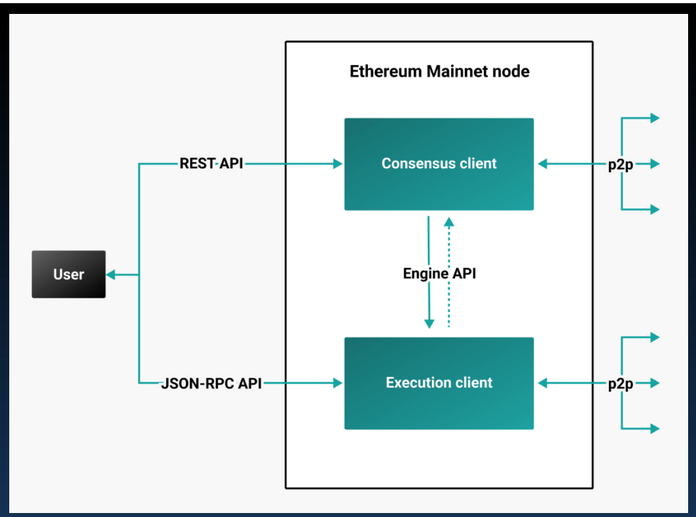


Proof of Stake Validators

- Becoming a Validator:
  - Deposit 32 ETH into the deposit contract
  - Run three software clients:
    - Execution client
    - Consensus client
    - Validator client
  - Join the activation queue

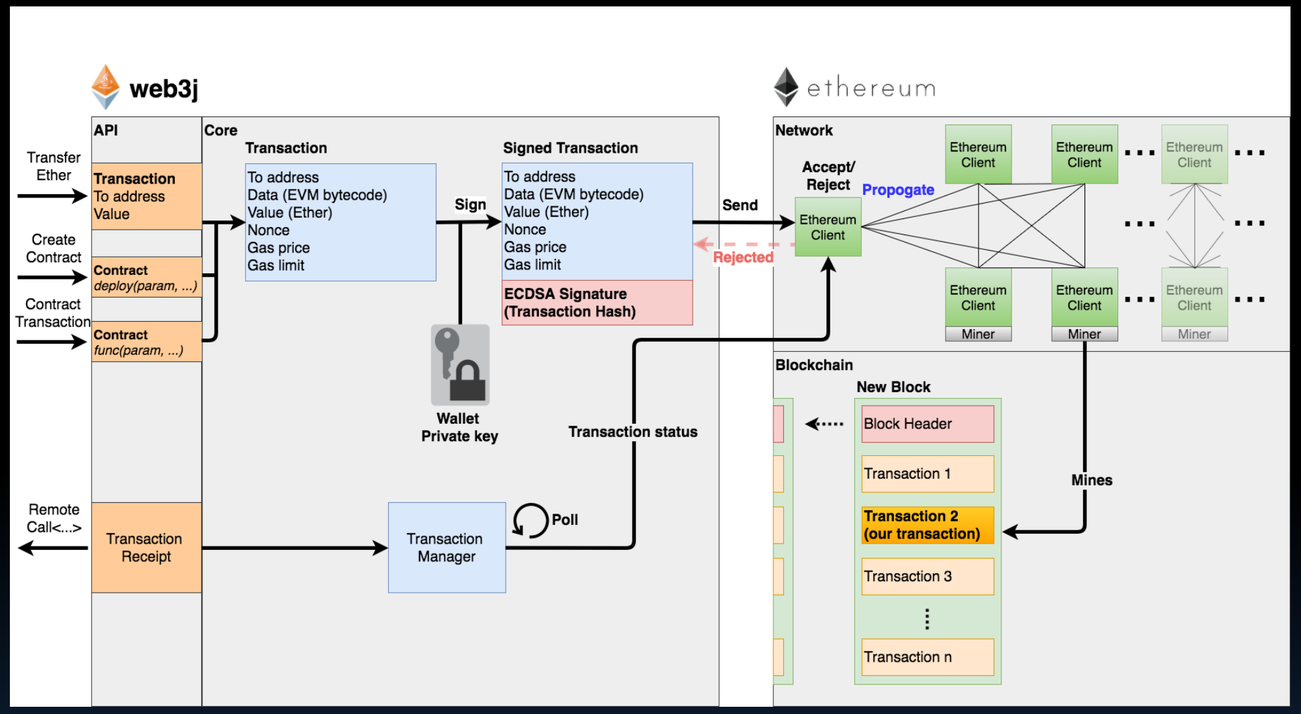
- Validator Duties
  - Receive new blocks
  - Re-execute transactions
  - Verify block signatures.
  - Vote (attest)





How a Transaction Gets Executed in Ethereum PoS

- Transaction Creation
  - Sign using private key
  - Include validator tip + base fee
- Submission & Mempool
  - Sent to execution client -> Added to mempool
- Block Proposal.
  - Validator selected using RANDAO
- Block Propagation
  - Using Gossip protocol
- Finalization

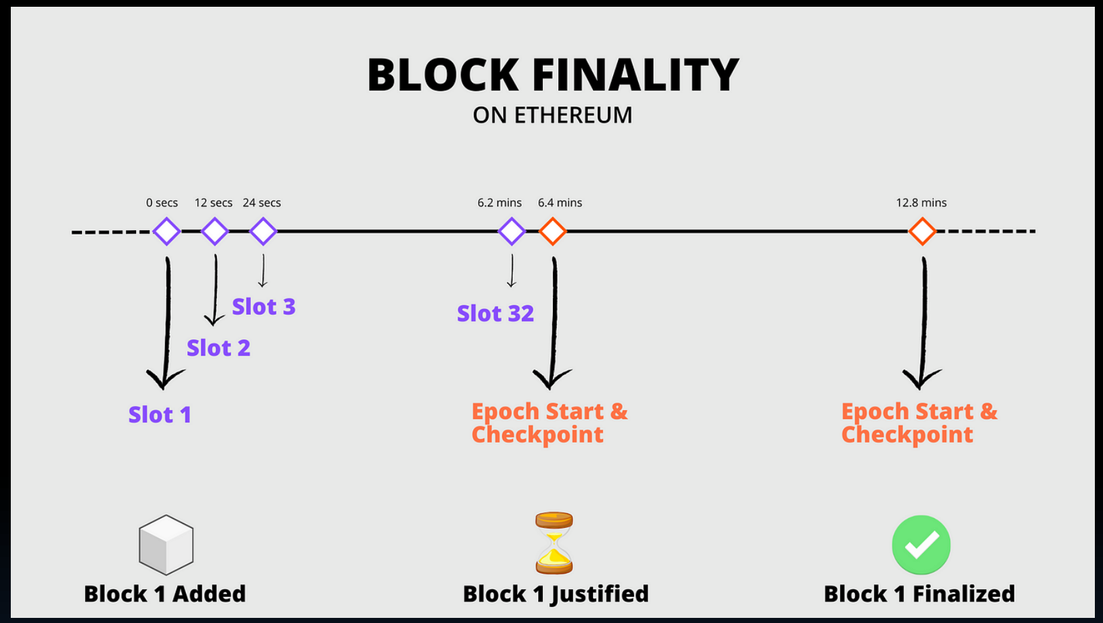


Finality in Ethereum PoS

- The **1st block of every epoch** = a **checkpoint**.
- Validators vote for **checkpoint pairs**.
- If votes represent **≥ 2/3 of total staked ETH**:
  - **Target checkpoint** becomes **justified**.
  - The previous justified checkpoint becomes **finalized**.

Reverting a finalized block requires burning at least **1/3 of all staked ETH**.

If finality fails for **4 epochs**, an **inactivity leak** activates

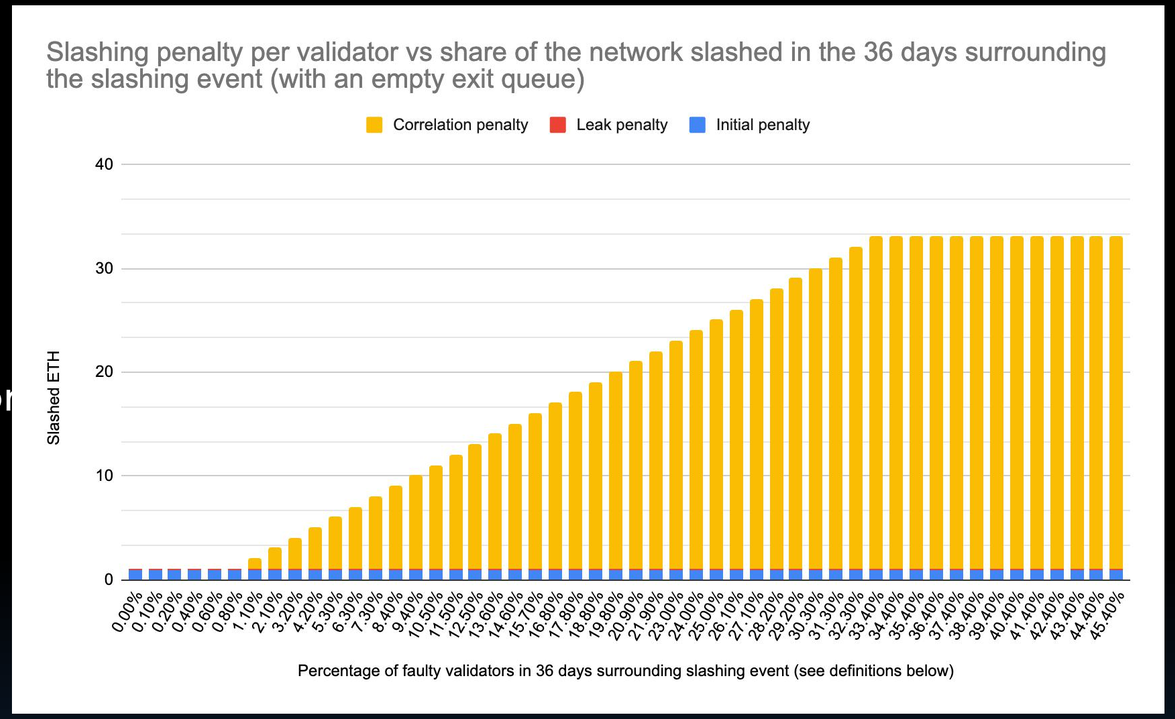


Crypto-Economic Security in Eth PoS

- Missed rewards for inactivity.

- Slashing for:
  - Proposing multiple blocks in one slot (equivocation).
  - Submitting contradictory attestations.

- Correlation Penalty depends on how many validator are slashed together.
  - ~1% stake if slashed individually.
  - Up to 100% stake loss in mass slashing events.



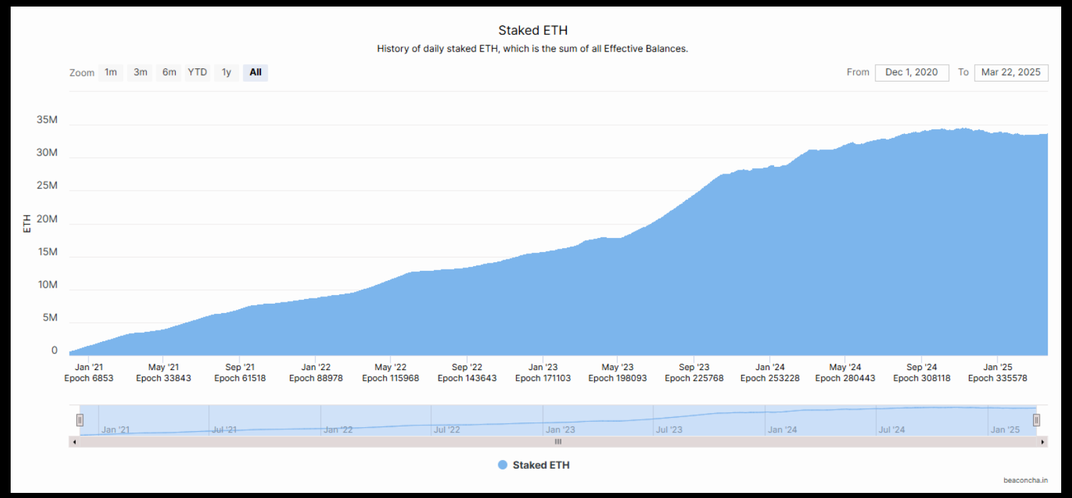
Penalty Timeline:
Day 1: Immediate penalty (up to 1 ETH).
Day 18: Correlation penalty applied.
Day 36: Validator forcibly ejected from network.
Daily: Minor penalties for inactivity.

33% Attack in Ethereum PoS

- a **33% attack** happens when a single entity or group controls **more than 33% of the total staked ETH**.

- **Possible risks:**
  - Can finalize and build invalid blocks.
  - May censor transactions (preventing them from entering blocks).
  - Can create conflicting checkpoints and disrupt finality.

- **Compared to PoW:**
  - In PoS, such an attack would require controlling 33%+ of the staked ETH, making it **extremely capital-intensive**.
  - Slashing mechanisms can severely penalize dishonest behavior, leading to **massive stake loss**.

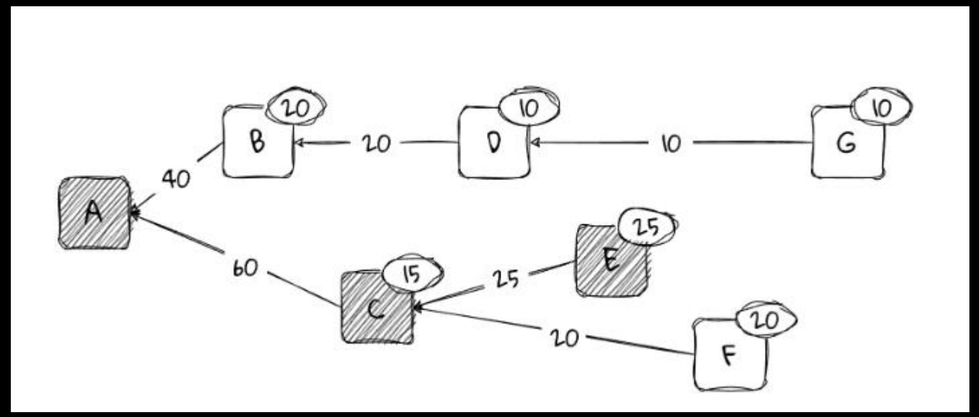


A 33% attack would require over $22.6 billion in ETH stake.
Even if such a sum could be raised, the attack carries the risk of slashing and social forking (where the network could "disconnect" malicious stakes).

From GHOST to LMD-GHOST

How LMD-GHOST Works:

- Chooses the fork (chain) with the **most accumulated attestations** (votes).
- Follows the branch with the **greatest validator weight** supporting it.
- GHOST is based on the number of blocks in subtrees (tree weight).
- LMD-GHOST is based on the fresh votes of validators and their stake-weight (validator weight) when choosing a chain.



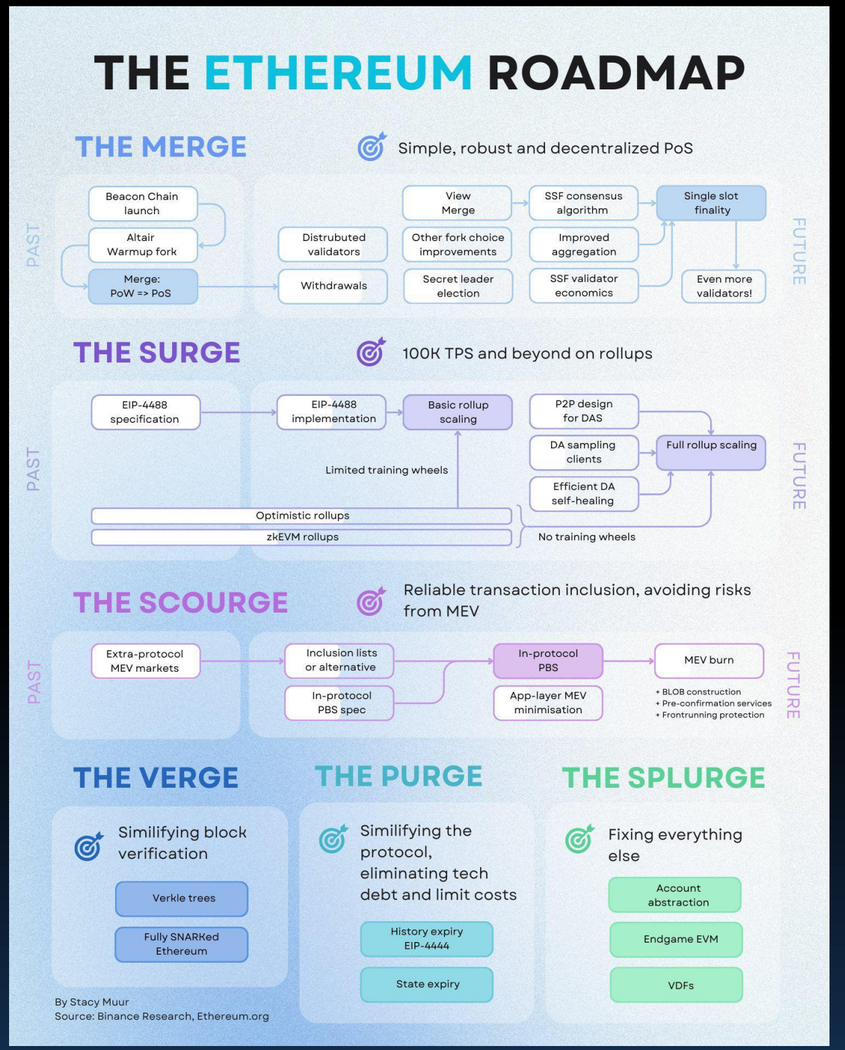
Aspect	GHOST (Ethereum PoW era)	LMD-GHOST (Ethereum PoS era)
Protocol Type	Fork-choice rule for PoW chains	Fork-choice rule for PoS (validators vote on chain head)
Purpose	Selects the heaviest subtree (counts uncles)	Selects the chain with the most recent validator attestations
Block Time Target	~12-15 seconds	Fixed 12-second slots
Finality	Probabilistic (based on number of confirmations)	Economic + deterministic finality (via checkpoints & attestations)
Uncles	Uncle blocks add weight to the chain (partial rewards)	No concept of uncles (validators vote directly)
Security Focus	Resilient to short-term forks due to high block frequency	Faster finality and better fork resolution based on validator votes
Attack Resistance	Depends on hash power distribution	Depends on validator participation and stake distribution

Summary Migration from PoW to PoS

Aspect	Proof-of-Work (PoW)	Proof-of-Stake (PoS)
Consensus Mechanism	Mining via computational work (hashing)	Validators stake coins and attest blocks
Energy Consumption	High (due to mining hardware)	Low (no mining, just validation)
Hardware Requirements	Specialized equipment (ASICs, GPUs)	Standard hardware (laptop, server)
Decentralization Risks	Mining pools can centralize hash power	More decentralized due to lower entry barriers
Block Time (Ethereum)	~12-15 seconds (before The Merge)	Fixed 12-second slots (after The Merge)
Finality	Probabilistic (waiting for 6-12 blocks)	Deterministic (checkpoints + supermajority votes)
Attack Costs	Requires 51% hash power	Requires 33%+ stake but risks slashing
Environmental Impact	High CO_2 emissions (electricity use)	Minimal environmental footprint
Token Issuance	Higher issuance to cover mining incentives	Lower issuance (validators need less compensation)
Recovery After Attack	No native recovery mechanism	Community coordination (e.g., social recovery of chain)

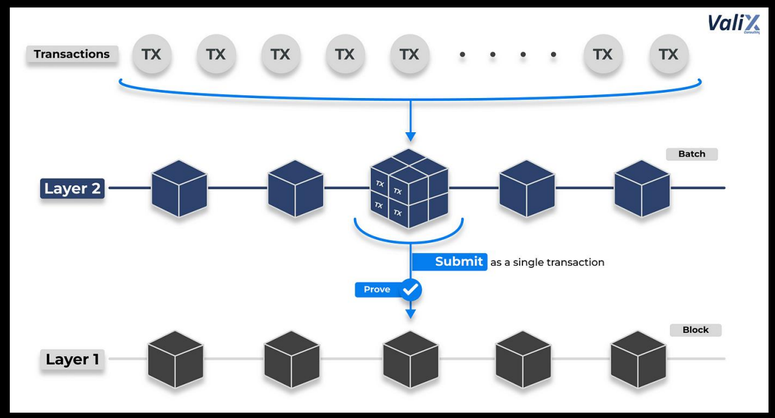
Ethereum roadmap

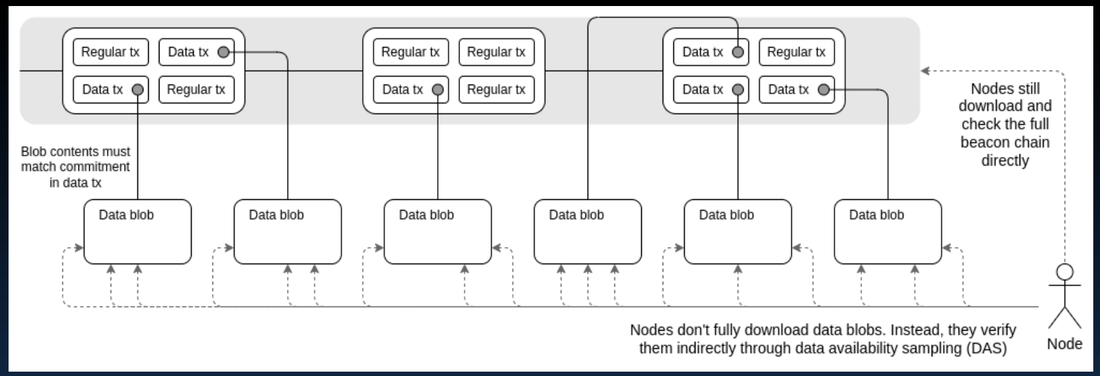
- Developed by Ethereum researchers, developers, and the community.
- Formalized as Ethereum Improvement Proposals (EIPs).
- The roadmap is dynamic — it evolves based on tech progress and network needs.
- Long-term upgrades may shift depending on external events (e.g., quantum computing advancements).



What About Sharding?

- **Old plan**: split Ethereum into shard chains.
- **New plan**: focus on **rollups + Danksharding**.
- **Rollups** batch transactions offchain, reducing costs, but data storage is still expensive.
- **Proto-Danksharding** introduces "blobs" (temporary data storage) to cut rollup costs further.
- **Goal**: Transactions could cost less than **$0.001**.
- **Current status**: Proto-Danksharding launched with the **Dencun upgrade** (March 2024).

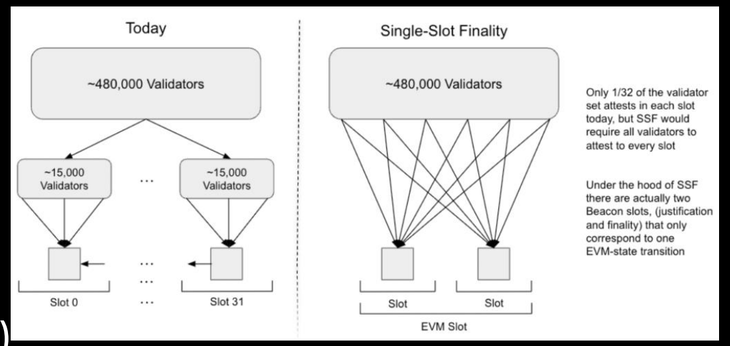


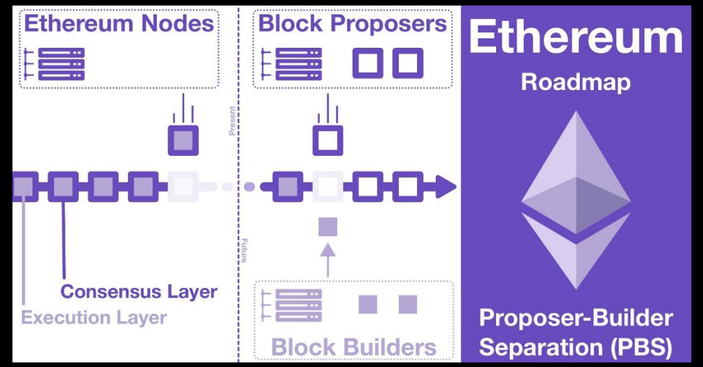


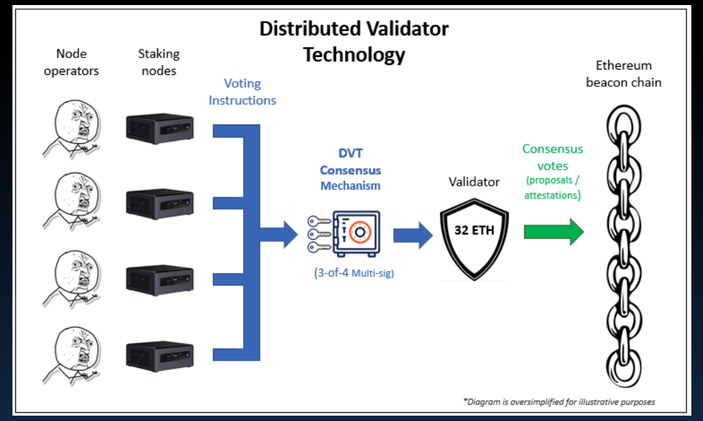
A More Secure Ethereum

Security Upgrades in Roadmap:

- Staking Withdrawals
- Defending Against Attacks:
  - View-Merge
  - Single Slot Finality (SSF)
- Defending Against Censorship:
  - Distributed Validator Tech (DVT)
  - Proposer-Builder Separation (PBS)
- Protecting Validators:
  - Secret Leader Election (SLE)



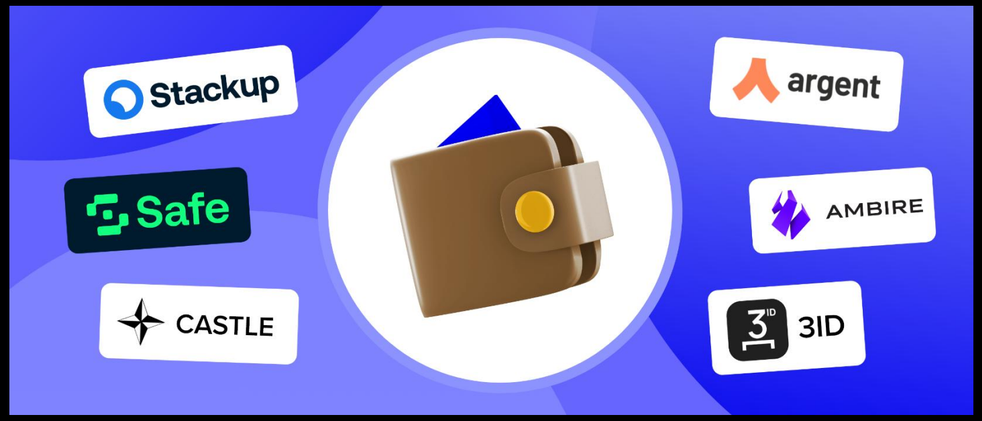




Improving User Experience on Ethereum

- Goal: Make Ethereum as easy to use as traditional web apps
- Beyond Seed Phrases Solution: Smart Contract Wallets + Account Abstraction.
- Nodes for Everyone. Running a node today is resource-heavy
- Future upgrades:

  Verkle Trees
  Statelessness/ Data expiry
  Light Nodes





Future-Proofing Ethereum. Quantum Resistance

• Ethereum must remain secure & reliable even as new threats (e.g., quantum computing) emerge.

• Quantum ResistanceSimplifying the system reduces attack surfaces and boosts performance.

- BLS Signatures
- KZG Commitments

• Replace with quantum-resistant cryptography:
- STARK-based signatures
- Lattice-based signatures



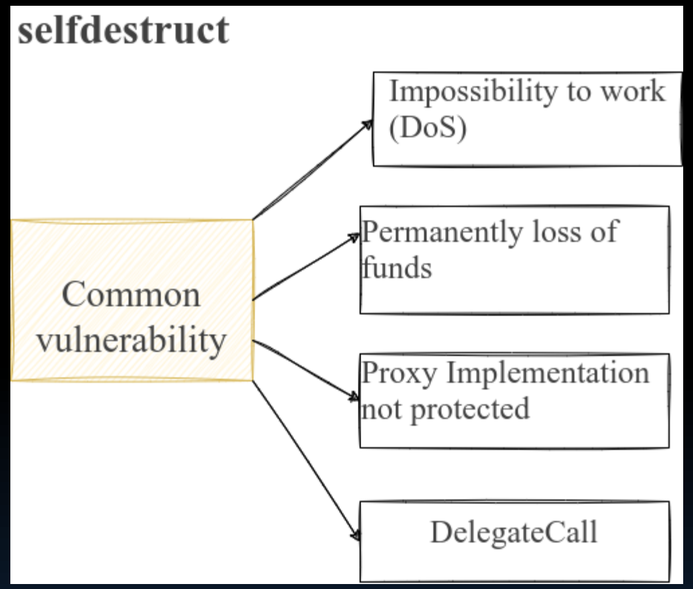
Quantum resistance = longer-term, but essential for Ethereum’s long-term survival.

Future-Proofing Ethereum. Simplifying Ethereum

- Simplifying the system reduces attack surfaces and boosts performance.

Upcoming changes:

- Remove SELFDESTRUCT opcode
- Deprecate old transaction types
- Optimize gas metering
- Unify compression



Thanks for attention

Any questions?

Resources

“Bitcoin and Cryptocurrency Technologies” by Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. Ethereum and Smart Contracts: chapter 10.7
Understanding blockchain: definitions, architecture, design, and system comparison: Sections III.B and IV.D.2, as well as Ethereum-related text in Section IV.B
Ethereum, Ethereum PoS
Ethereum, Ethereum future upgrades
Ethereum, Scaling Ethereum
Ethereum, A more secure Ethereum
Ethereum, Improving user experience
Ethereum, Future-proofing Ethereum
Quora, What is the blocktime of Ethereum?
Etherscan. Analytics of Ethereum 2025
Quora, What is the size of a block in the Ethereum blockchain?
Mirror.xyz, Ethereum replay attack
NikArt, Alan Turing: The Man Who Changed History Twice in a Single Moment
Beaconcha. Chart of staked ether amount
EthRoadmap. Interactive Eth Roadmap.