Lesson 1

Week 1

Lesson 1 - Introduction to Blockchain and Layer 1

Lesson 2 - Why Scalability

Lesson 3 - Introduction to Layer 2

Lesson 4 - Maths and Cryptography

Today's topics

- Decentralised Systems
- Blockchain timeline
- Layer 1 Theory
- Consensus on Ethereum

Course Introduction

Course Plan

Week 1 Background / Scalability

- Intro to L1
- Why Scalability
- Intro to L2s
- Maths and Cryptography

Week 2 -

- Criteria for Understanding and Analysing L2s
- Agnostic L2 Transaction Lifecycle
- Optimistic Rollups v ZK Rollups
- What's next in L2s (1) - Decentralized Sequencers

Week 3

- What's next in L2s (2) - Privacy, L3s/Hyperchains
- Privacy in L2s
- What are ZK EVMs 1 - overview
- What are ZK EVMs 2 - universal circuits/circuit compiler

Week 4

- What are ZK EVMs 3 - proving system: zk algorithm/ASIC optimisation
- zkEVM security
- Overview of Proving Systems
- SNARK implementation

Week 5

- STARK implementation
- Plonk part 1 / Linea
- Plonk part 2 / Boojum
- Advanced Plonk / Scroll

Week 6

- Formal Verification
- Risc Zero
- ZK Machine Learning
- Review

Practical Details

The format will usually be 45 mins of theory followed by 45 mins practical

You do not need to submit the homeworks

You can work in teams in breakout rooms
We use Sli.do to provide Q&A and polls : link

About Extropy

Free Developer Workshops

- Basic
- Enterprise
- Advanced EVM
- Zero Knowledge Proofs

Business Workshops

Website : https://extropy.io

Email : info@extropy.io

Twitter : @extropy

Social Media : @Extropy

https://discord.gg/XzZS3FBx

General Resources

Zero Knowledge Podcast

Decentralised Systems

Problems with centralised systems

Monetary System

- Bank closure / insufficient capital reserves
- greek debt crisis in 2015 ? banks closed and people lost savings, insurance schemes meant nothing, lead to an increase in Bitcoin use in Greece
- Availability of banks
- Inflation - money supply controlled by central authority
- Merchant accounts may be shut down
- Control of money for political reasons - wikileaks funding shutdown

There are layers of access control built into our banking systems to prevent fraudulent transactions, effectively security is achieved by closing the network.

Goals of decentralisation

- Participation
- Diversity
- Conflict resolution
- Flexibility
- Moving power to the edge (user)

Blockchain Timeline

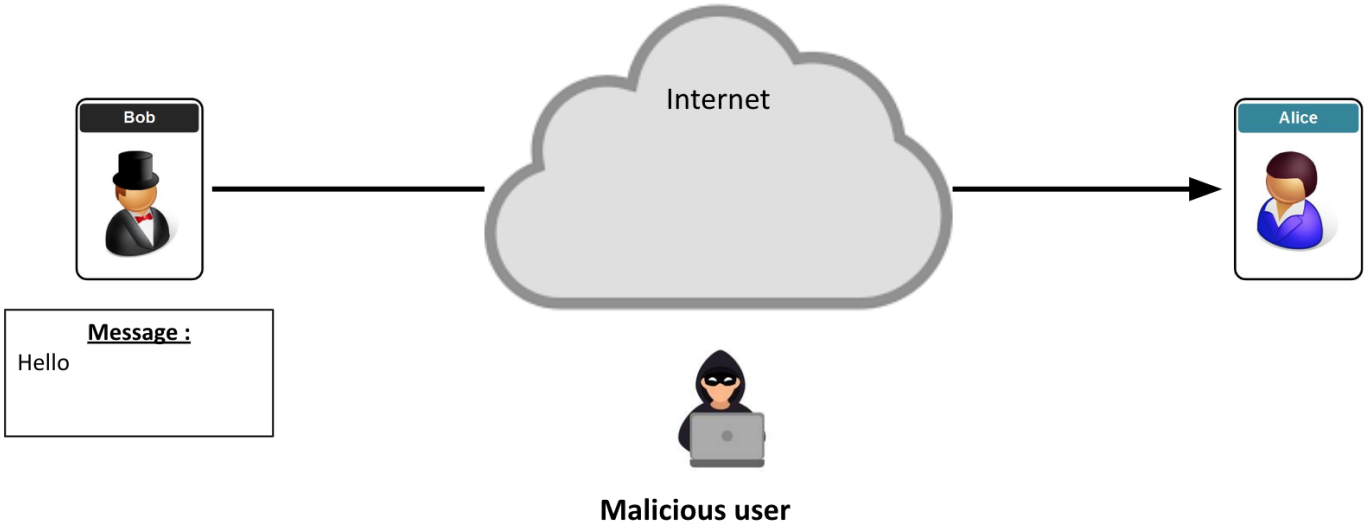
1970s

Problem = Security !

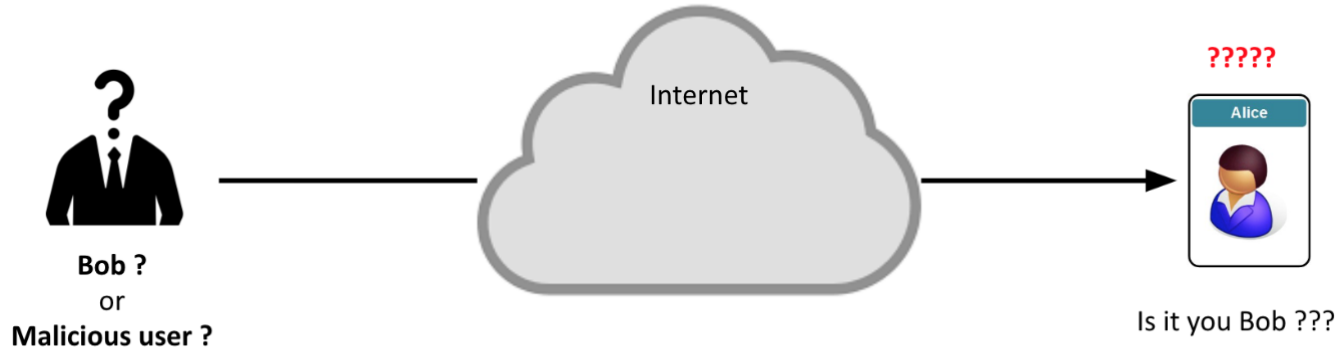
- How do I ensure that my message has not been modified ?
- How do I ensure that the message comes from a legitimate person ?

Secure Communication over Insecure Channel

Problem 1 : How do I ensure that my message has not been modified ?



Problem 2 : How do I ensure that the message comes from a legitimate person ?



pre 1970s solution : Symmetric Cryptography !

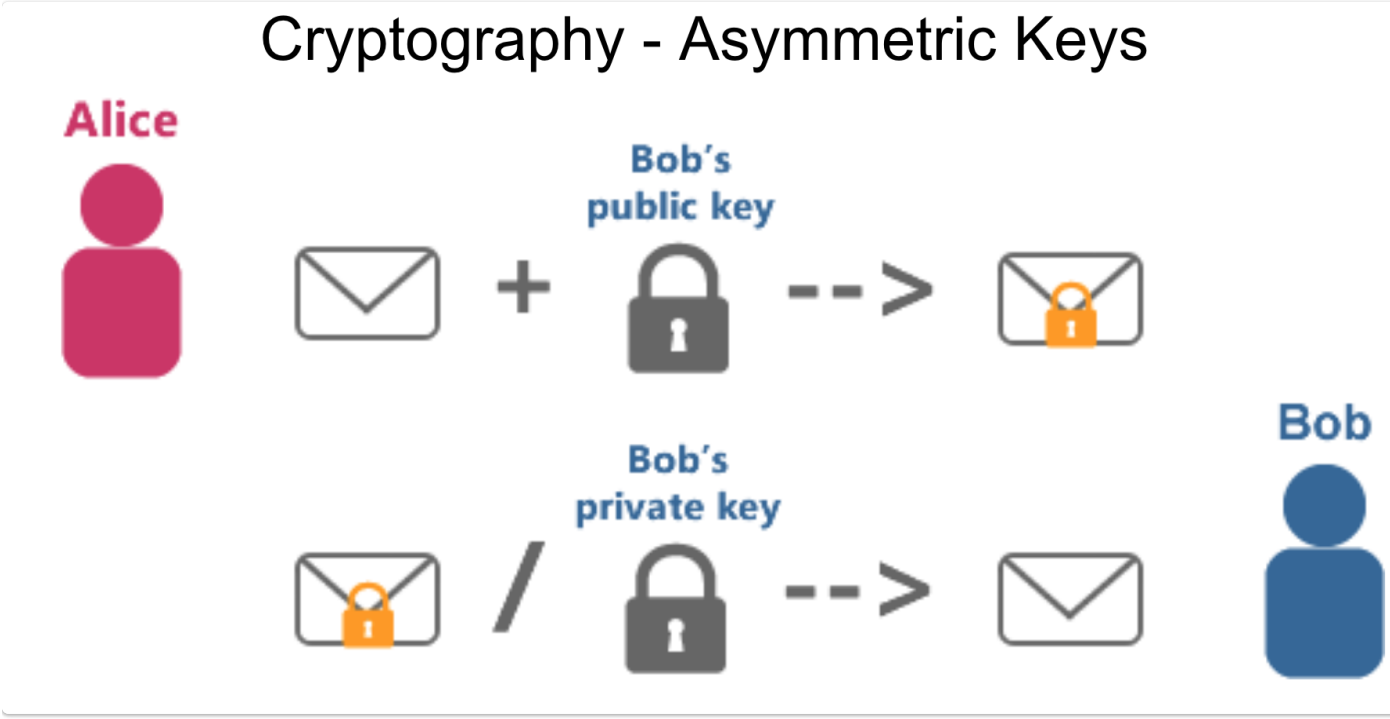
- Alice and Bob share the same key.
- One key for both encryption and decryption of messages

But what about key management ?

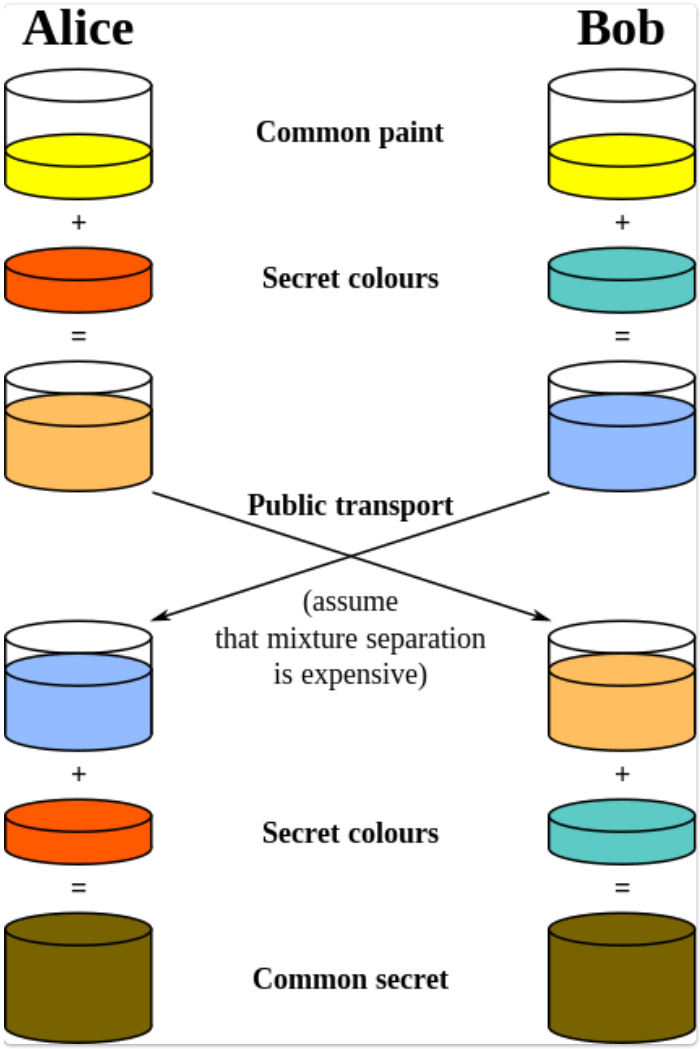
Can Alice and Bob share a key

- Without meeting
- Across a potentially hostile network

In the 1970s there was a major step forward with the invention of asymmetric cryptography.
Here 2 paired keys are used, one for encryption and one for decryption.



A method was also designed to allow the production of a shared secret across an insecure channel, via Diffie Helman Key Exchange. An analogy of the process is provided by thinking of colours being mixed. See [Guide] (https://github.com/archit-p/simplest-oblivious-transfer)



Public Key Encryption solves :

Problem 1 : Key Management

- If I use a 3rd party to share my key, do I trust him ?

Problem 2 : Integrity

- How do I ensure that my message has not been modified ?

Problem 3 : Authenticity

- How do I ensure that the message comes from a legitimate person ?

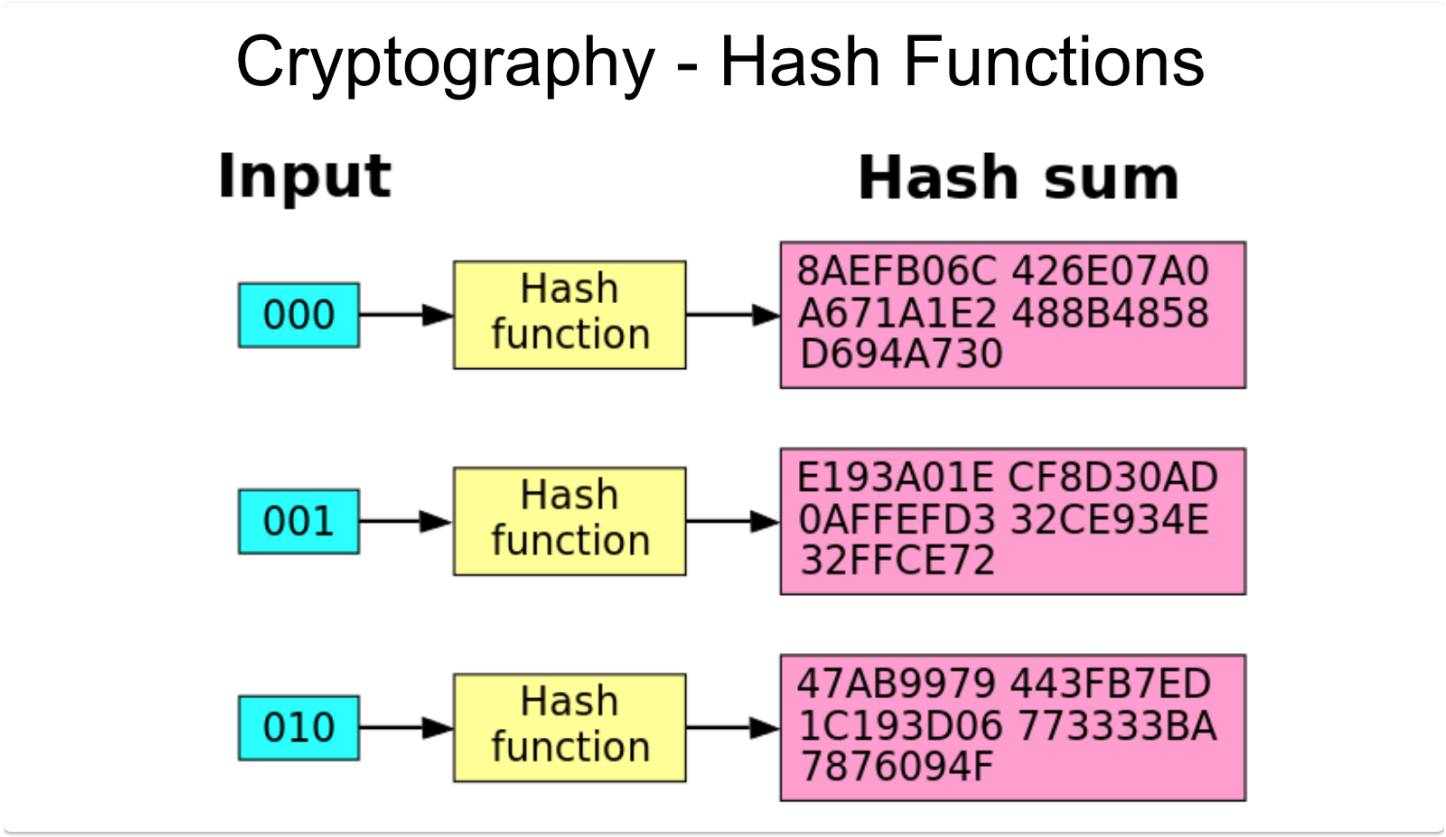
This final problem was solved by the invention of digital signatures.

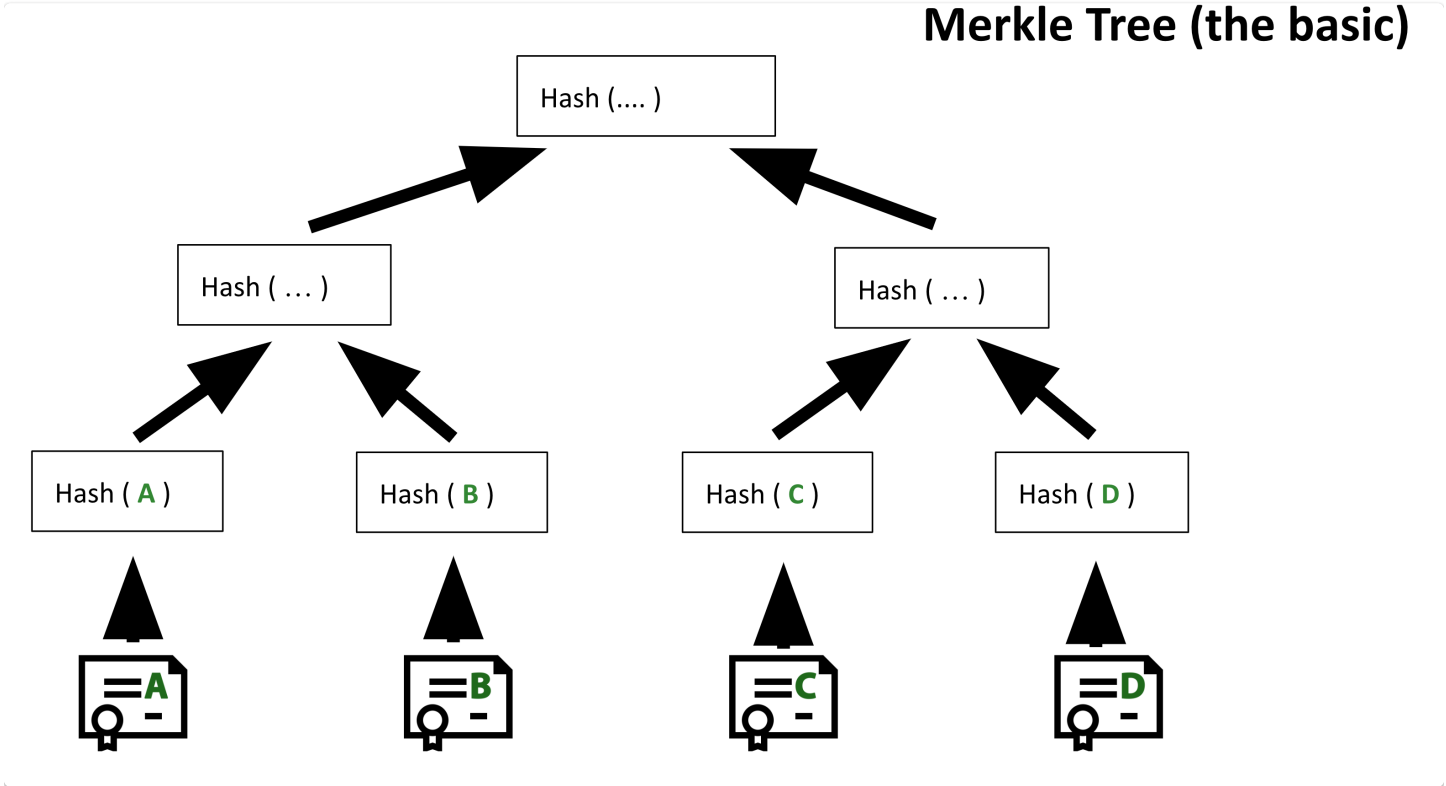
Digital Signatures

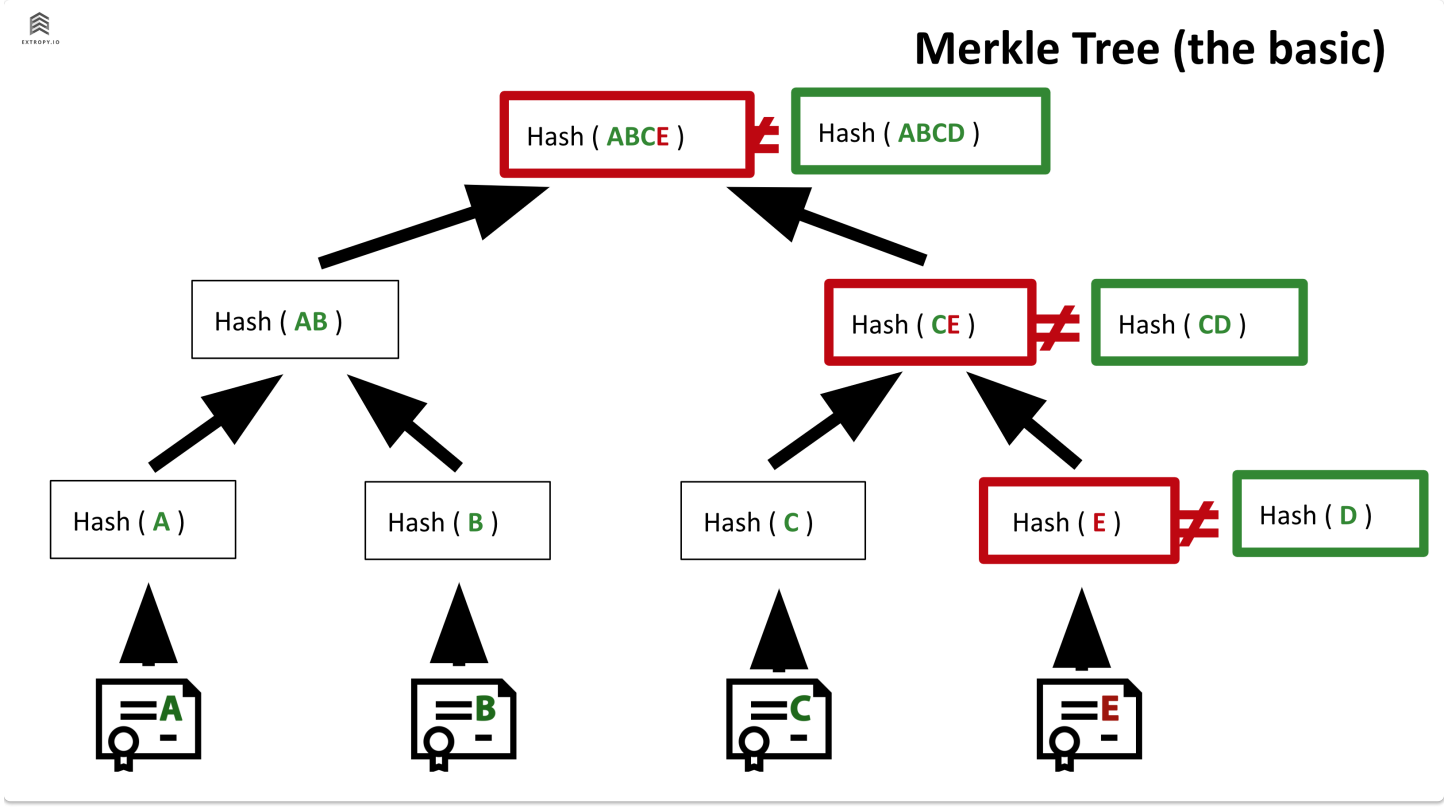
We use digital signatures as a way to show that a message came from a particular person (or holder of a key)
For example imagine Bob is signing a document and sending it to Alice.

A digital signature will have 4 properties

- Authenticity : when Alice verifies the message with Bob's public key, she knows that he signed the message.
- Unforgeable : only Bob knows his private key.
- Not reusable : the signature is tightly bound with the document, it cannot be transferred to any other document.
- Unalterable : If there is any alteration to the document, it will no longer be verifiable with Bob's public key.


Cryptography - Hash Functions
Input
Hash sum
000
Hash function
8AEFB06C 426E07A0 A671A1E2 488B4858 D694A730
001
Hash function
E193A01E CF8D30AD 0AFFEFD3 32CE934E 32FFCE72
010
Hash function
47AB9979 443FB7ED 1C193D06 773333BA 7876094F


Merkle Tree (the basic)
Hash (....)
Hash (…)
Hash (…)
Hash (A)
Hash (B)
Hash (C)
Hash (D)
A
B
C
D


Merkle Tree (the basic)
Hash (ABCE)
≠
Hash (ABCD)
Hash (AB)
Hash (CE)
≠
Hash (CD)
Hash (A)
Hash (B)
Hash (C)
Hash (E)
≠
Hash (D)
A
B
C
E

That is the cryptographic background, how did people try to use this technology ?

The development of

- Electronic cash
- Timestamping
- P2P Systems
- Consensus systems

1980s

David Chaum - Blind Signatures
David Chaum - Digicash

1990s

Timestamping records
Adam Back - HashCash
Wei Dai - B-Money

2000s

Peer to peer networks

- Freenet / Gnutella / Bit Torrent

Further Attempts at Electronic Cash

"the one thing that's missing is a reliable e-cash, whereby on the internet you can transfer funds from A to B without A knowing B or B knowing A" - Milton Friedman 1999

1998 - b-money - Wei Dai (http://www.weidai.com/bmoney.txt)
1998 - Bit Gold - Nick Szabo (https://nakamotoinstitute.org/bit-gold/)

Bitcoin

Satoshi Nakamoto is the name used by the presumed pseudonymous person or persons who developed bitcoin, authored the bitcoin white paper, and created and deployed bitcoin's original reference implementation

Early Bitcoin History

August 2008 - the domain name bitcoin.org registered
October 2008 - A Peer-to-Peer Electronic Cash System posted to a cryptography mailing list
January 2009 - Software implementation released as open source

2010, the first known commercial transaction using bitcoin occurred when programmer Laszlo Hanyecz bought two Papa John's pizzas for 10,000 BTC

General Blockchain events since 2009

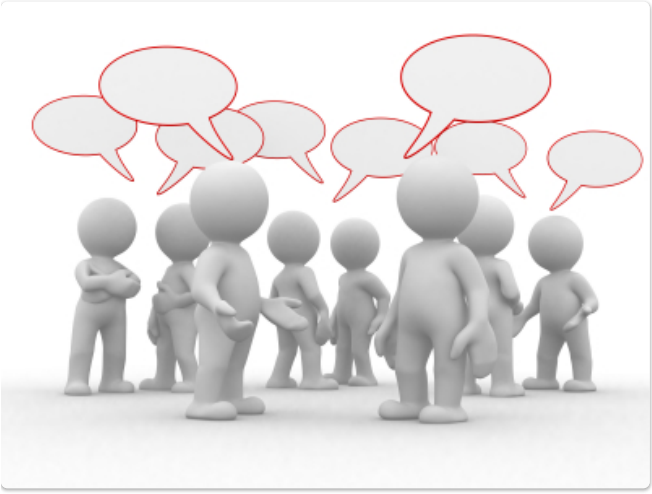
2014 - Ethereum created
2017 - ICO Boom / Alternatives to Ethereum
2018 - Crypto winter
2020 - DeFi summer
2021 - Rise of NFTs / Gaming
2022 - Ethereum Merge
2022 - Another crypto winter

Layer1Theory

A simplistic / traditional view of blockchains

Typical Blockchain components

Gossip network



Shared public ledger

**

Cryptography

**

These components give the blockchain

- Transparency and verifiable state based on consensus
- Resilience

- Censorship resistance
- Tamper proof interactions

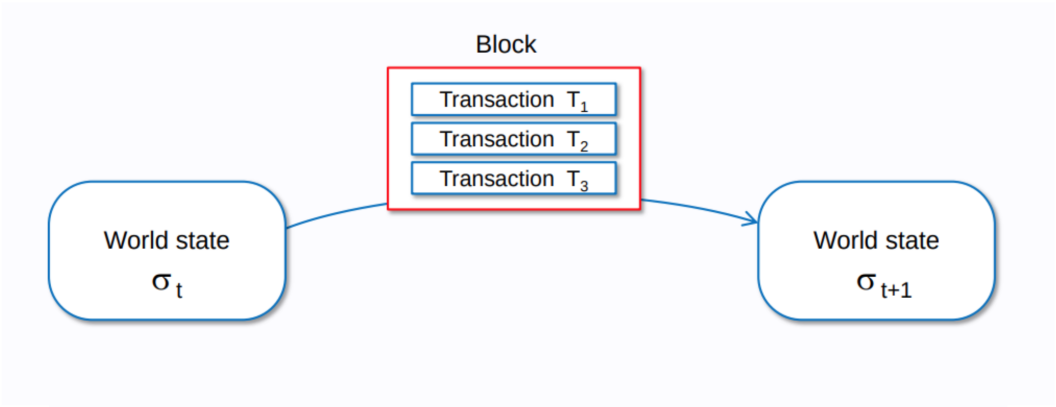
Blockchain components (Bitcoin / Ethereum) in more detail

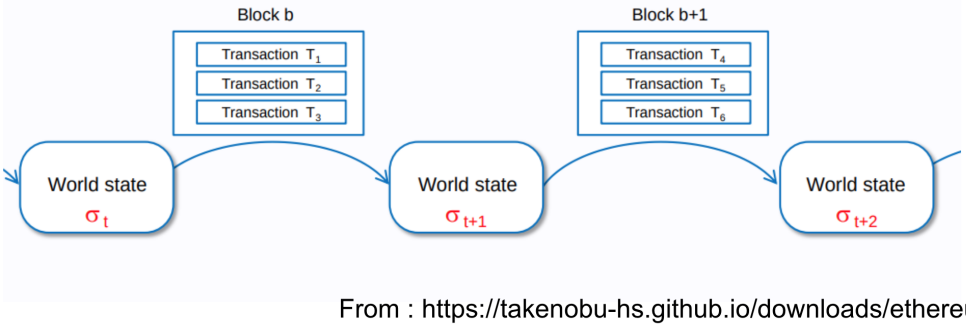
- A peer-to-peer (P2P) network connecting participants and propagating transactions and blocks of verified transactions, based on a standardised "gossip" protocol
- Messages, in the form of transactions, representing state transitions
- A set of consensus rules, governing what constitutes a transaction and what makes for a valid state transition
- A state machine that processes transactions according to the consensus rules
- A chain of cryptographically secured blocks that acts as a journal of all the verified and accepted state transitions
- A consensus algorithm that decentralizes control over the blockchain, by forcing participants to cooperate in the enforcement of the consensus rules
- A game-theoretically sound incentivization scheme (e.g., proof-of-work costs plus block rewards) to economically secure the state machine in an open environment
- One or more open source software implementations of the above ("clients")

State Machine

A useful way to think of the interaction between blocks, transactions and blockchain state is to model the blockchain as a state machine.
The state transition is caused by the execution of a set of transactions gathered in a block.
The use of a block is helpful to delineate the state transition, but is it not universally used.
Aptos / Sui and to some extend Solana focus more on the transactions.

Blockchain as a state machine





From : https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

Traditional Blockchain architecture

Network

From : Bitcoin book

Bitcoin is structured as a peer-to-peer network architecture on top of the internet. The term peer-to-peer, or P2P, means that the computers that participate in the network are peers to each other, that they are all equal, that there are no "special" nodes, and that all nodes share the burden of providing network services.
The network nodes interconnect in a mesh network with a "flat" topology. There is no server, no centralised service, and no hierarchy within the network.

Blockchain nodes

Nodes will typically

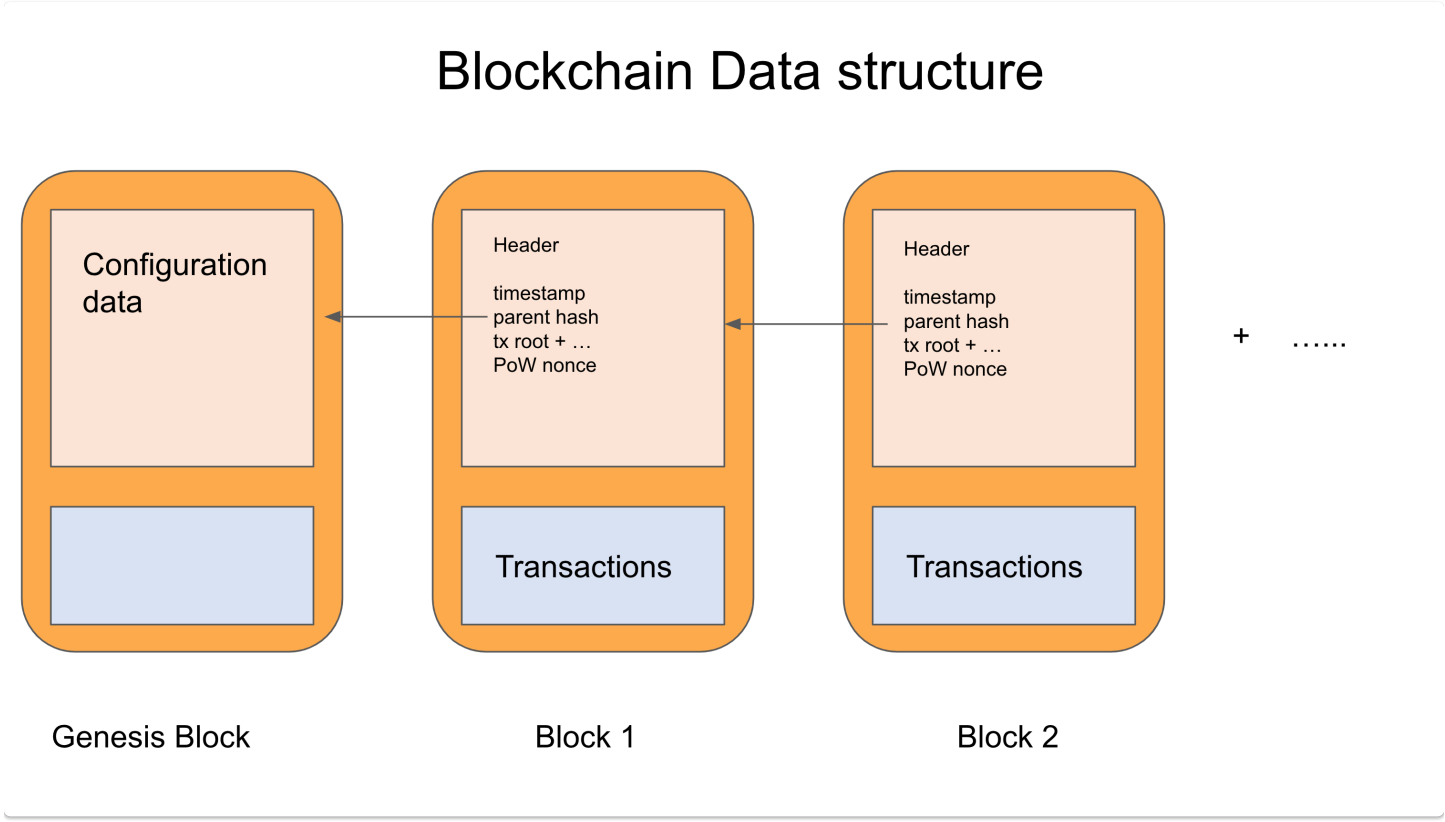
- Accept and transmit transactions (if valid)
- Keep a mempool of pending transactions
- Provide network discovery and routing functions
- Drop connections to misbehaving nodes

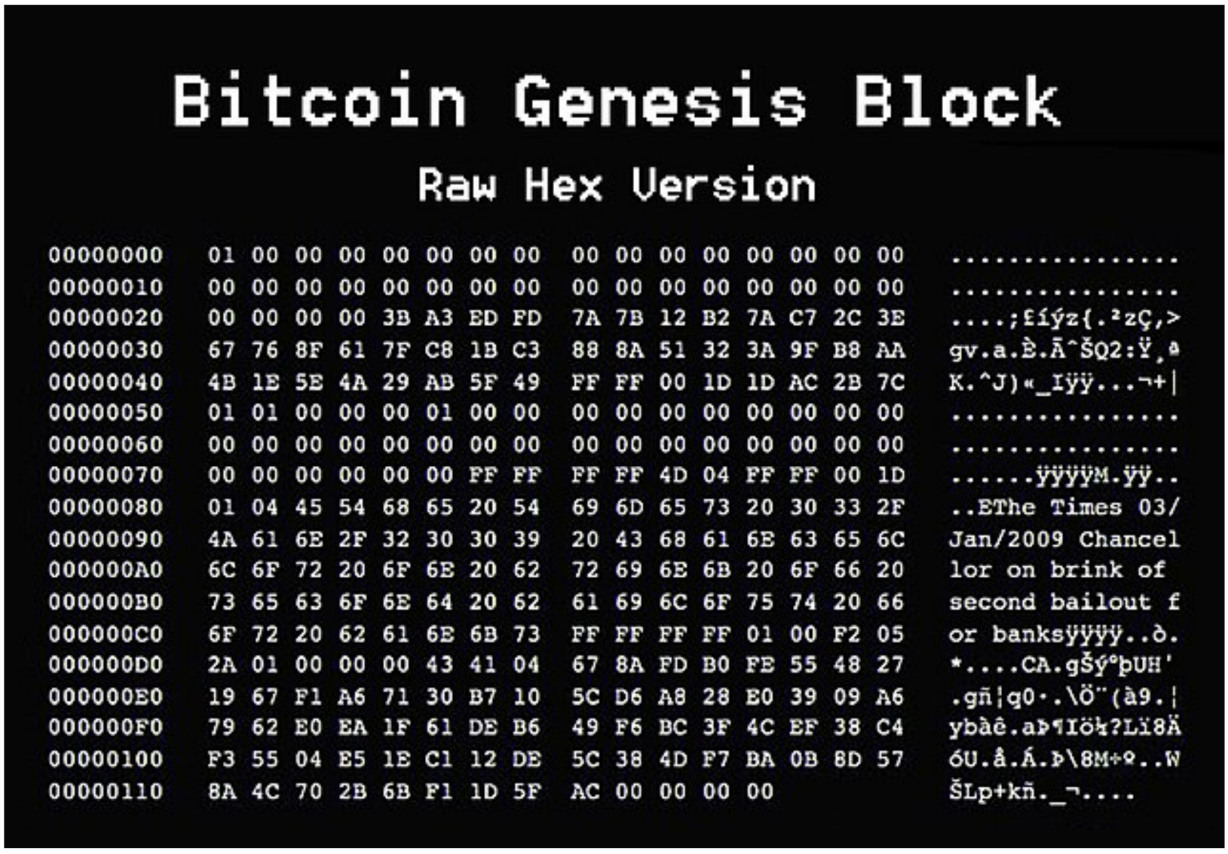
- Accept blocks and update their local ledger

The connections are not based on geographical proximity but proximity in a hash table
Node discovery is via

- DNS seed nodes
- A locally stored list

Block data structures





By MikeG001 - Own work, CC BY-SA 4.0,

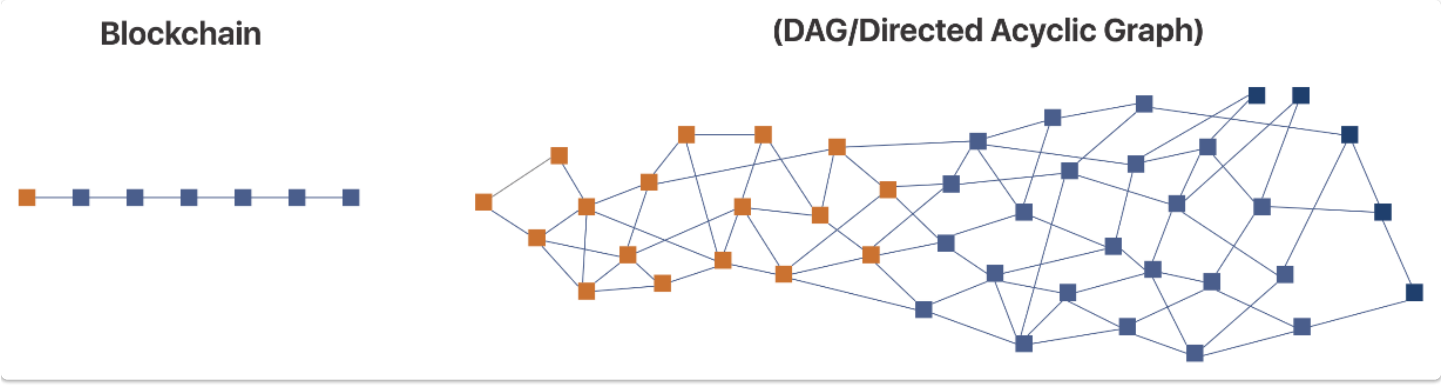
Storage data structures

Merkle trees or variants such as Merkle Patricia trees are widely used to organise storage, though this can be an abstraction, the physical storage of items, including blocks and blockchain state is up to the design of the node software.
In Ethereum roadmap Merkle trees are planned to be replaced by Verkle trees for better performance.

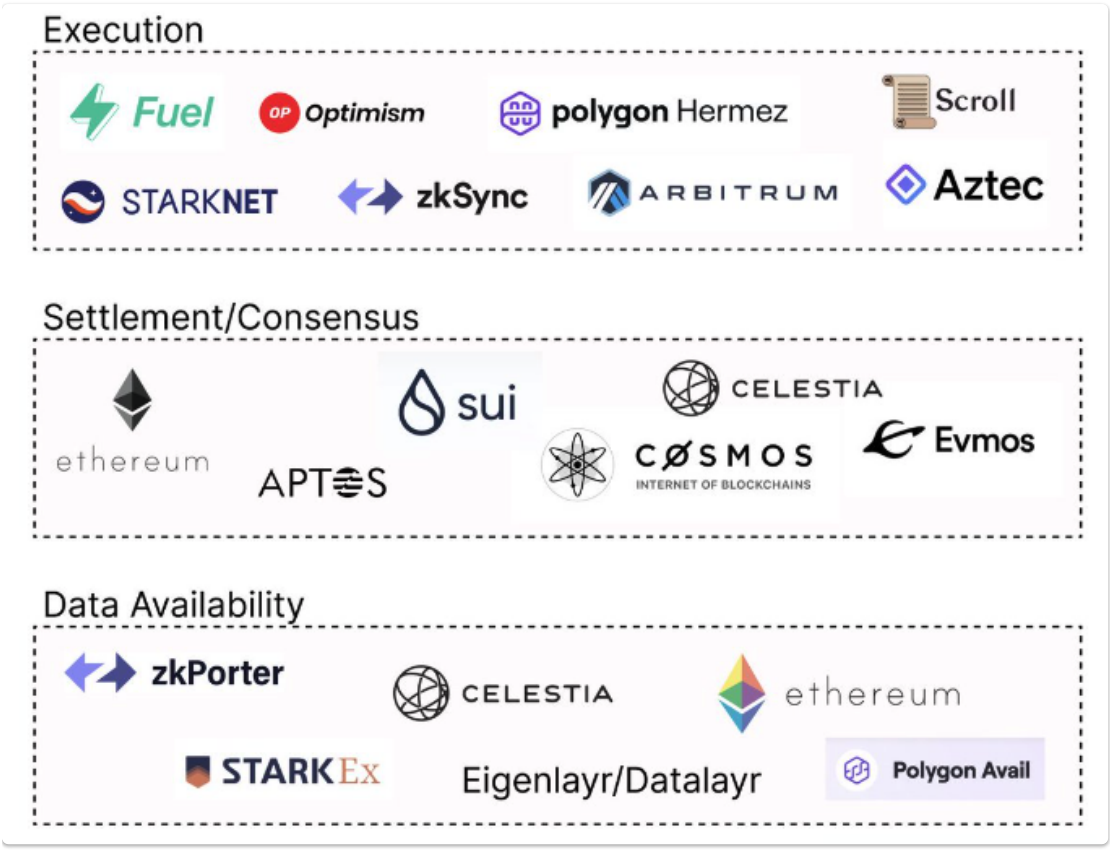
Alternative data structures

Some chains such as Sui and IOTA use a Directed Acyclic Graph (DAG) rather than a linear list of blocks.

DAGs are a structure where there are no loops, and the items are linked with directed links.
Traditional blockchains can be seen as a specific case of a DAG in which each block only has one parent, DAGs in general allow multiple parents.



Modular Blockchains



If we follow the principle of separation of concerns, we can use a combination of blockchains to provide the functionality of a L1 and increase scalability.
For further details see Volt article

The Blockspace Market

See article

Demand Side

The demand side is formed from the collection of transactions supplied by users (via DApps)

Supply Side

The supply side is provided by the validators proposing and voting on blocks, adding to the Ethereum blockchain.
The 2 sides are mediated by congestion and fees.
The mechanics are complex and give rise to features such as

- MEV
- ASIC development

We can see from this, that the functionality we need to provide falls into 3 categories

- Execution
- Settlement / Consensus
- Data availability

Execution

This is where smart contracts or programs implement the business logic.
We can design the environment to optimise the execution, which means we can move away from the EVM if desired.
Some execution layers such as Starknet use a completely different execution environment, and smart contract language to Ethereum L1.

Settlement Consensus

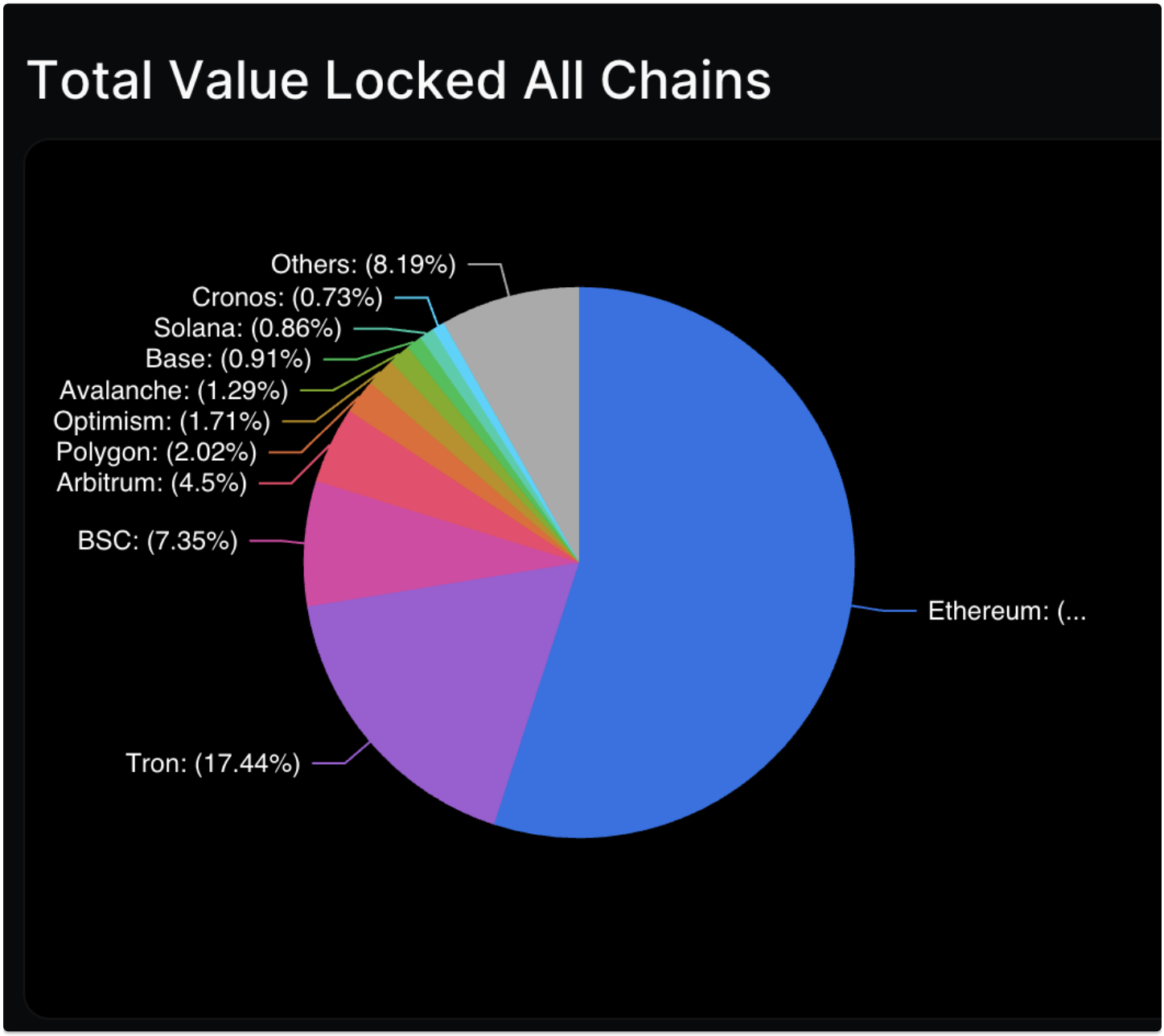
In this layer we come to agreement about the state transitions, using consensus mechanisms for security and decentralisation.
The execution layer will send updates to the settlement layer, which once accepted are seen as securing the updates to the execution layer.

Data Availability

Data availability is the ability to verify that the information for a proposed block was actually published on the blockchain network. A block is "unavailable" if the producer publishes the block itself but withholds data used to create the block.

Chains overview

From Defi Llama



Consensus on Ethereum

From Ethereum developer documentation
"Now technically, proof-of-work and proof-of-stake are not consensus protocols by themselves, but they are often referred to as such for simplicity. They are actually Sybil resistance mechanisms and block author selectors; they are a way to decide who is the author of the latest block. It is this Sybil resistance mechanism combined with a chain selection rule that makes up a true consensus mechanism."

There are 2 parts to block addition :

- block producer selection
- block acceptance

From yellow paper
" Since the system is decentralised and all parties have an opportunity to create a new block on some older pre-existing block, the resultant structure is necessarily a tree of blocks. In order to form a consensus as to which path, from root (the genesis block) to leaf (the block containing the most recent transactions) through this tree structure, known as the blockchain, there must be an agreed-upon scheme. If there is ever a disagreement between nodes as to which root-to-leaf path down the block tree is the 'best' blockchain, then a fork occurs."

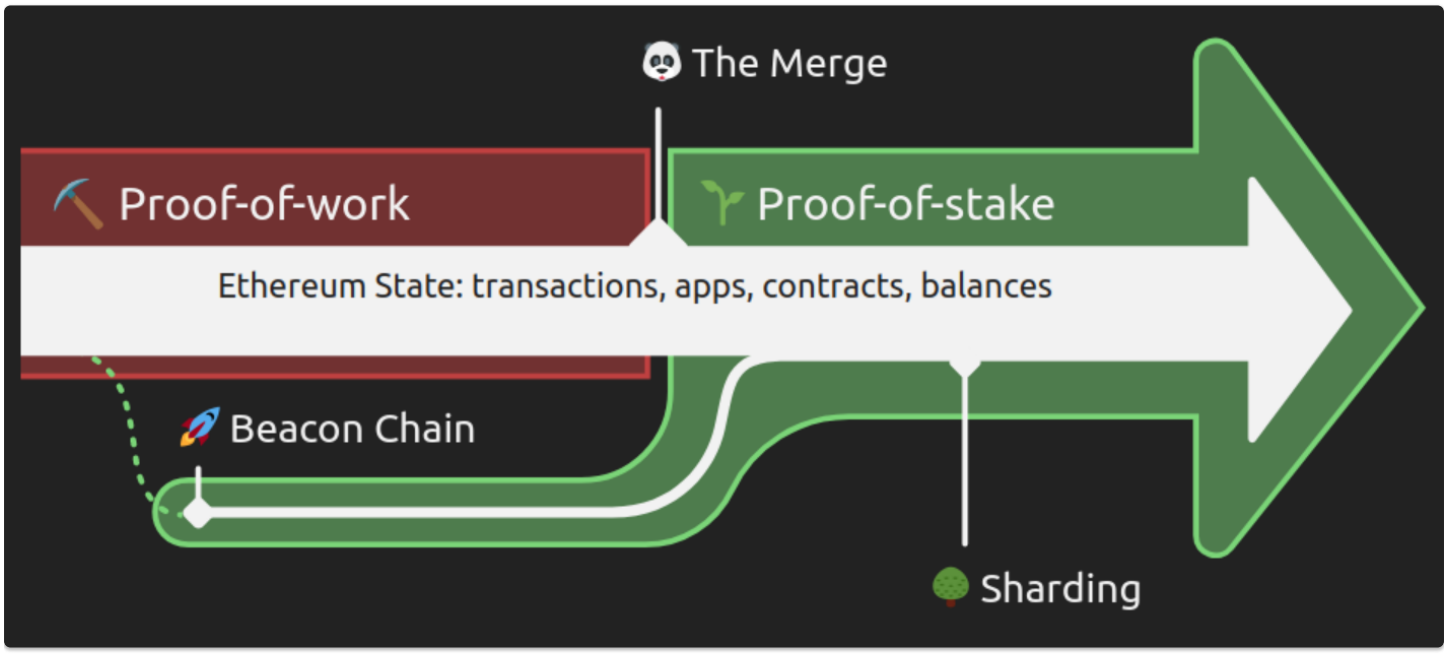
Consensus Mechanism Families

- Nakamoto style - Bitcoin, Ethereum pre merge
- BFT style - Ethereum post merge
- Avalanche Snow

The Merge - Proof of stake update

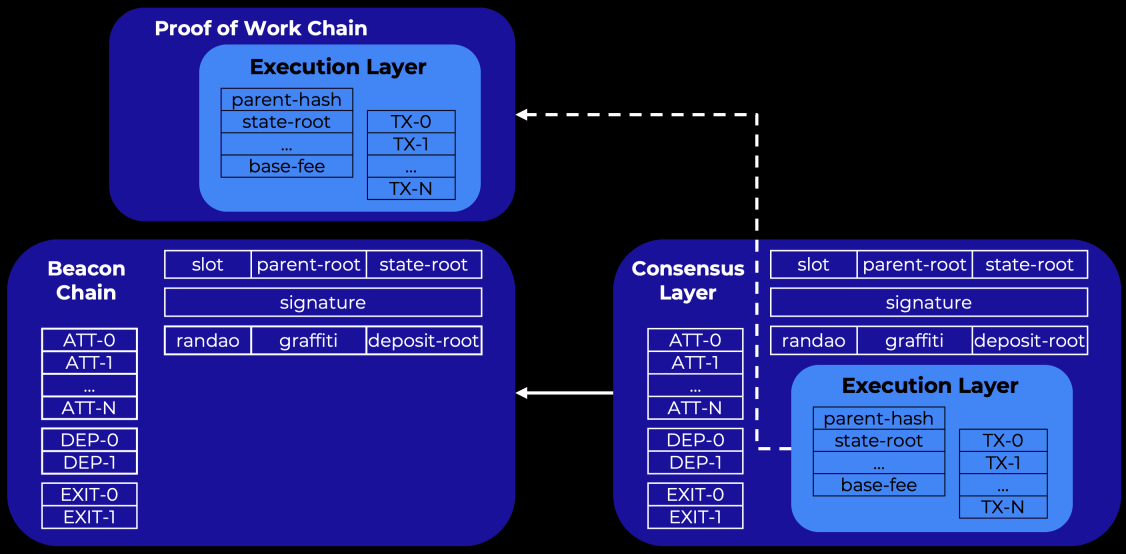
- Replaced proof of work with the proof of stake beacon chain.
  i.e. merging existing beacon chain into ethereum.
- Pre merge the Beacon Chain had not been processing Mainnet transactions. Instead, it has been reaching consensus on its own state by agreeing on active validators and their account balances.
- POS specs: https://github.com/ethereum/consensus-specs#phase-0

"The number `58750000000000000000000` joins the list of the most important integers of our time. The moment Ethereum's proof of work chain accumulates that much difficulty (read, hashes done), the entire network will switch over to proof of stake"



The Merge

Plugging the Execution Layer into the Beacon Chain Consensus Layer

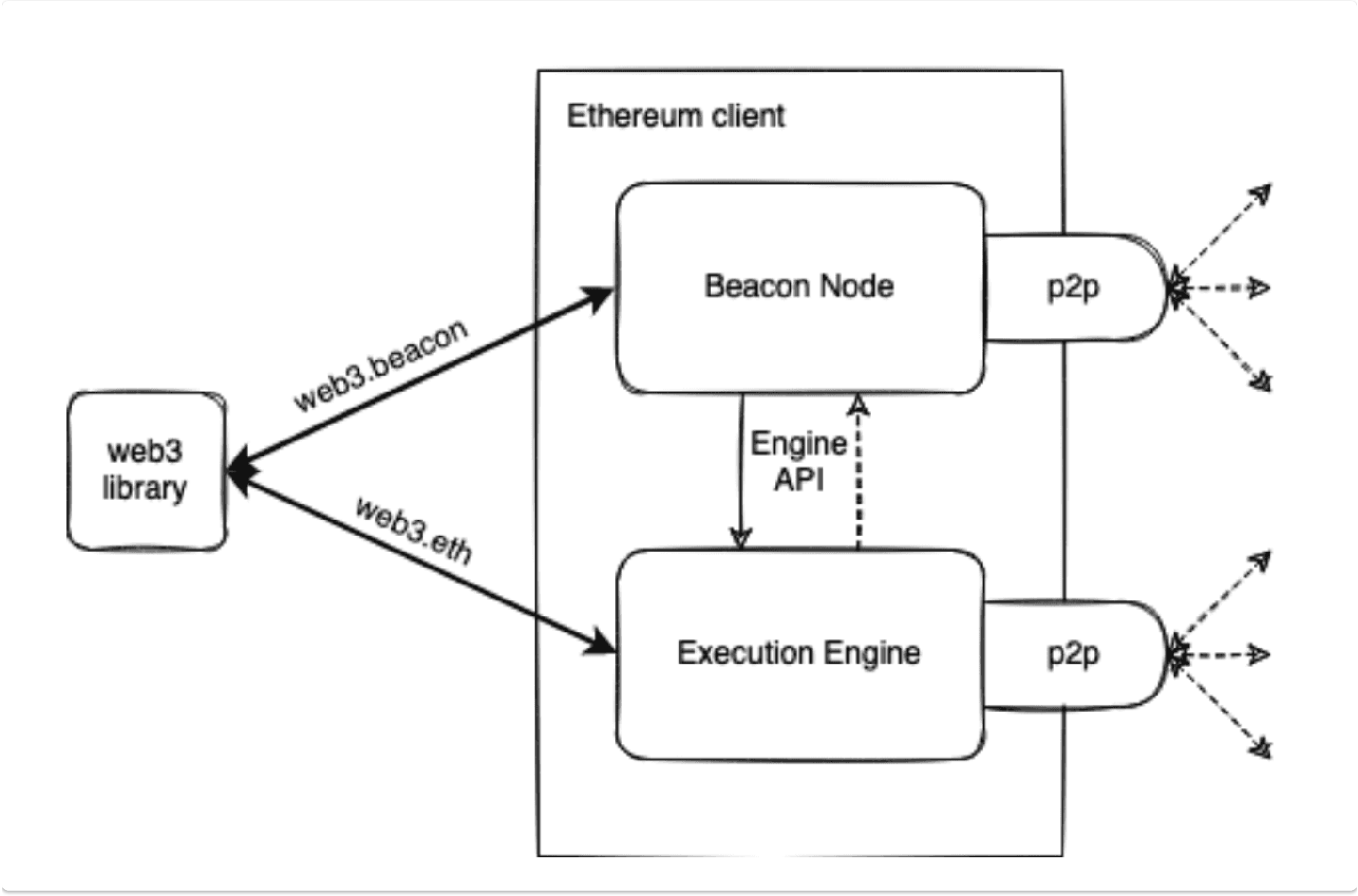


Source: Tim Beiko, Ethereum Foundation

DELPHI DIGITAL

Ethereum clients after the merge

- Eth 1.0 clients continue to handle execution. They process blocks, maintain mempools, and manage and sync state. The PoW stuff has been taken out.
- Consensus client – Current Beacon Chain clients continue to handle PoS consensus. They track the chain's head, gossip and attest to blocks, and receive validator rewards.



Execution Clients

See Docs

Client	Language	Operating Systems	Networks
Geth	Go	Linux, Windows, macOS	Mainnet, Sepolia, Goerli
Nethermind	C#, .NET	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more
Besu	Java	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more
Erigon	Go	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more
Reth	Rust	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more

Consensus Clients

See Docs

Client	Language	Operating systems	Networks
Lighthouse	Rust	Linux, Windows, macOS	Beacon Chain, Goerli, Pyrmont, Sepolia, Ropsten, and more
Lodestar	TypeScript	Linux, Windows, macOS	Beacon Chain, Goerli, Sepolia, Ropsten, and more
Nimbus	Nim	Linux, Windows, macOS	Beacon Chain, Goerli, Sepolia, Ropsten, and more
Prysm	Go	Linux, Windows, macOS	Beacon Chain, Gnosis, Goerli, Pyrmont, Sepolia, Ropsten, and more
Teku	Java	Linux, Windows, macOS	Beacon Chain, Gnosis, Goerli, Sepolia, Ropsten, and more

A number of different synchronisation modes are available see docs

Client diversity in the network is very much encouraged

Consensus after the Merge

Ethereum has moved to Gasper (Casper FFG + LMD GHOST (Latest Message Driven Greediest Heaviest Observed SubTree))

Consensus relies on both LMD-GHOST – which adds new blocks and decides what the head of the chain is – and Casper FFG which makes the final decision on which blocks *are* and *are not* a part of the chain.
GHOST's favourable liveness properties allow new blocks to quickly and efficiently be added to the chain, while FFG follows behind to provide safety by finalising epochs.

The two protocols are merged by running GHOST from the last finalised block as decided upon by FFG.
By construction, the last finalised block is always a part of the chain which means GHOST doesn't need to consider earlier blocks.

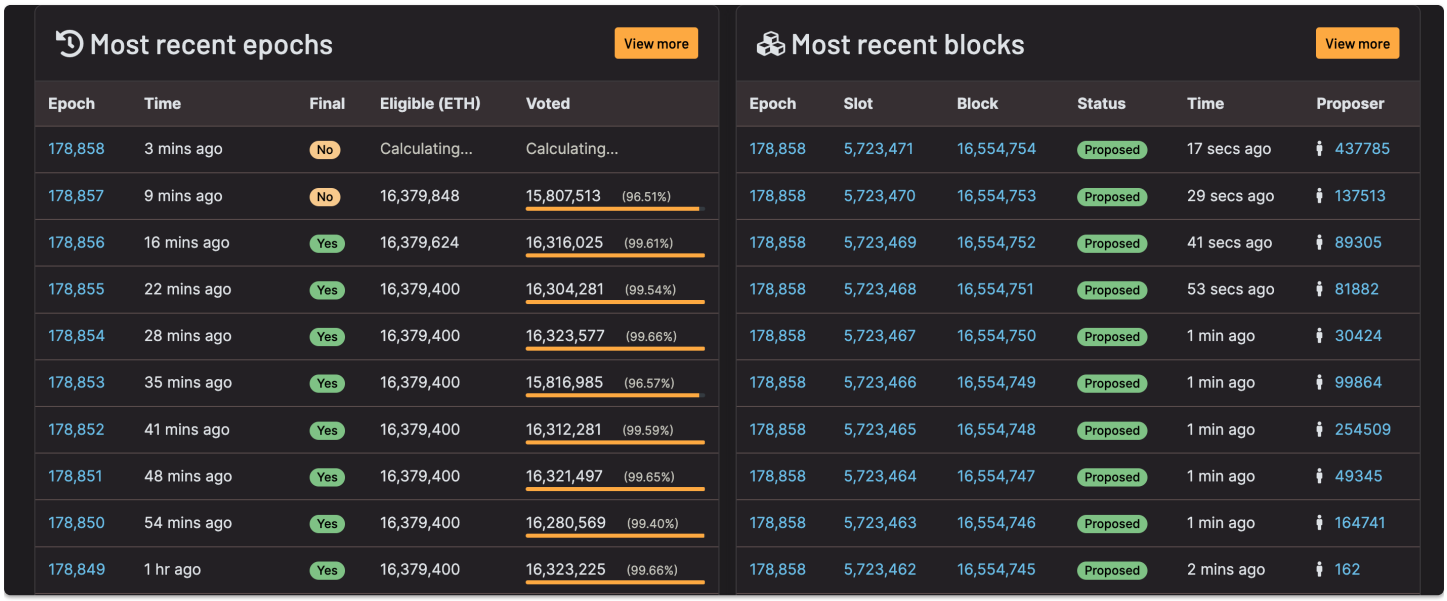
Safety favouring protocols such as Tendermint can halt, if they don't get enough votes.
Liveness favouring protocols such as Nakamoto continue to add blocks, but they may not come to finality.
Ethereum will achieve finality by checkpointing

Epochs and checkpointing

Epochs of about 6 mins have 32 slots with all validators attesting to one slot (~12K attestations per block)
The fork-choice rule LMD GHOST then determines the current head of chain based on these attestations.
Finality is achieved when sufficient votes are reached, generally after 2 epochs.

Validator Selection and consensus in more detail

You can find stats about the validators here



Most recent epochs — View more

Epoch	Time	Final	Eligible (ETH)	Voted
178,858	3 mins ago	No	Calculating...	Calculating...
178,857	9 mins ago	No	16,379,848	15,807,513 (96.51%)
178,856	16 mins ago	Yes	16,379,624	16,316,025 (99.61%)
178,855	22 mins ago	Yes	16,379,400	16,304,281 (99.54%)
178,854	28 mins ago	Yes	16,379,400	16,323,577 (99.66%)
178,853	35 mins ago	Yes	16,379,400	15,816,985 (96.57%)
178,852	41 mins ago	Yes	16,379,400	16,312,281 (99.59%)
178,851	48 mins ago	Yes	16,379,400	16,321,497 (99.65%)
178,850	54 mins ago	Yes	16,379,400	16,280,569 (99.40%)
178,849	1 hr ago	Yes	16,379,400	16,323,225 (99.66%)

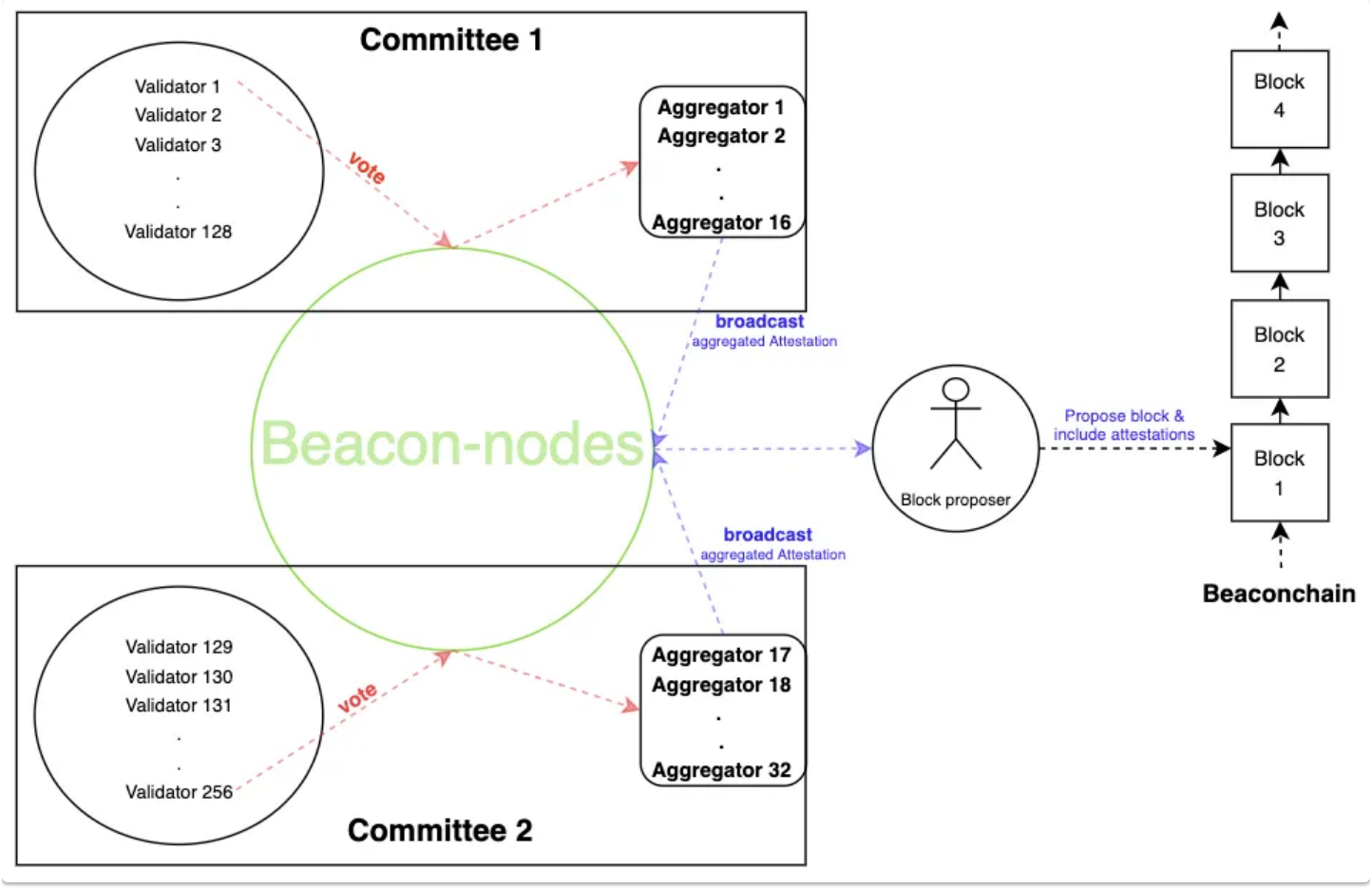
Most recent blocks — View more

Epoch	Slot	Block	Status	Time	Proposer
178,858	5,723,471	16,554,754	Proposed	17 secs ago	437785
178,858	5,723,470	16,554,753	Proposed	29 secs ago	137513
178,858	5,723,469	16,554,752	Proposed	41 secs ago	89305
178,858	5,723,468	16,554,751	Proposed	53 secs ago	81882
178,858	5,723,467	16,554,750	Proposed	1 min ago	30424
178,858	5,723,466	16,554,749	Proposed	1 min ago	99864
178,858	5,723,465	16,554,748	Proposed	1 min ago	254509
178,858	5,723,464	16,554,747	Proposed	1 min ago	49345
178,858	5,723,463	16,554,746	Proposed	1 min ago	164741
178,858	5,723,462	16,554,745	Proposed	2 mins ago	162

A slot occurs every 12s and one validator is chosen to submit a block within that slot.
If the validator fails to do so, the slot is let empty.
The first block within an epoch (~6.4 mins) is a checkpoint block.

Coming to consensus about the block

If a validator is not chosen to produce a block, it will instead vote on what it regards as the current head of the chain and the checkpoint block. Within an epoch a validator will only vote once.



Validators are grouped into committees at random, their votes are aggregated and published in the block header.

This article gives an in depth view of validator rewards

Fork choice rule

When faced with a potential fork, we choose the fork that has the most votes, but when counting the votes, we only include the last one from any validator.

Voting rules

The validator must vote for the chain they consider to be correct
The validator cannot vote for 2 blocks in any one time slot.

Finalisation

Validators vote on a pair of checkpoint blocks, to indicate that they are valid.
Once a checkpoint block gets sufficient votes, it is regarded as 'justified'
Once its child checkpoint block becomes justified, then the parent is regarded as final.