Blockchain

Tecnologie Internet

a.a. 2022/2023

Outline

- Definition of Blockchain
- Coin vs. Token
- Transactions
- Bitcoin
- Smart Contract
- Ethereum
- Consensus Models

Definition of Blockchain

A **distributed ledger** is a database that is replicated across several nodes or computing devices.

Each node has an identical copy of the ledger.

Each participant node of the network updates itself independently.

The ledger is not maintained by any central authority.

Blockchains are *one form* of distributed ledger technology.

Definition of Blockchain

A blockchain is

- a system that acts as a trusted and reliable third party, not centralized, always online, to preserve a shared state, mediate exchanges and provide secure computations

- a **distributed ledger**, storing **transaction data**, grouped into **blocks** that constitute a **growing and inalterable** linked list

- managed by a large group of networked servers

Definition of Blockchain

- **full node**: stores a copy of the whole blockchain, may create blocks

- **consensus** between full nodes: they have the same blocks

- **wallet**: software for making transactions and checking their validity (using asymmetric encryption)

Definition of Blockchain

- **public = permissionless**: anyone can
  - be a user (by means of a wallet) or participate with a full node
  - execute transactions
  - participate to the consensus process that defines the evolution of the blockchain

examples: Bitcoin, Ethereum, Algorand

- **private = permissioned**: blockchain operated by a recognized entity; identifiable full nodes; rules for data access

  - **consortium**: special case of permissioned blockchain, operated by multiple entities

P2P Network

Full nodes collaboratively maintain a peer-to-peer network for block and transaction exchange. Many wallets also use this protocol to connect to full nodes.

Consensus rules do not cover networking, so full nodes may use alternative networks and protocols, such as the **high-speed block relay network** used by some miners and the **dedicated transaction information servers** used by some wallet programs.

When started for the first time, programs do not know the IP addresses of any active full nodes. In order to discover some IP addresses, they may query specific servers called **DNS seeds**. The response to the lookup includes one or more **DNS A records** with the IP addresses of full nodes that may accept new incoming connections.

http://tools.ietf.org/html/rfc1035#section-3.2.2

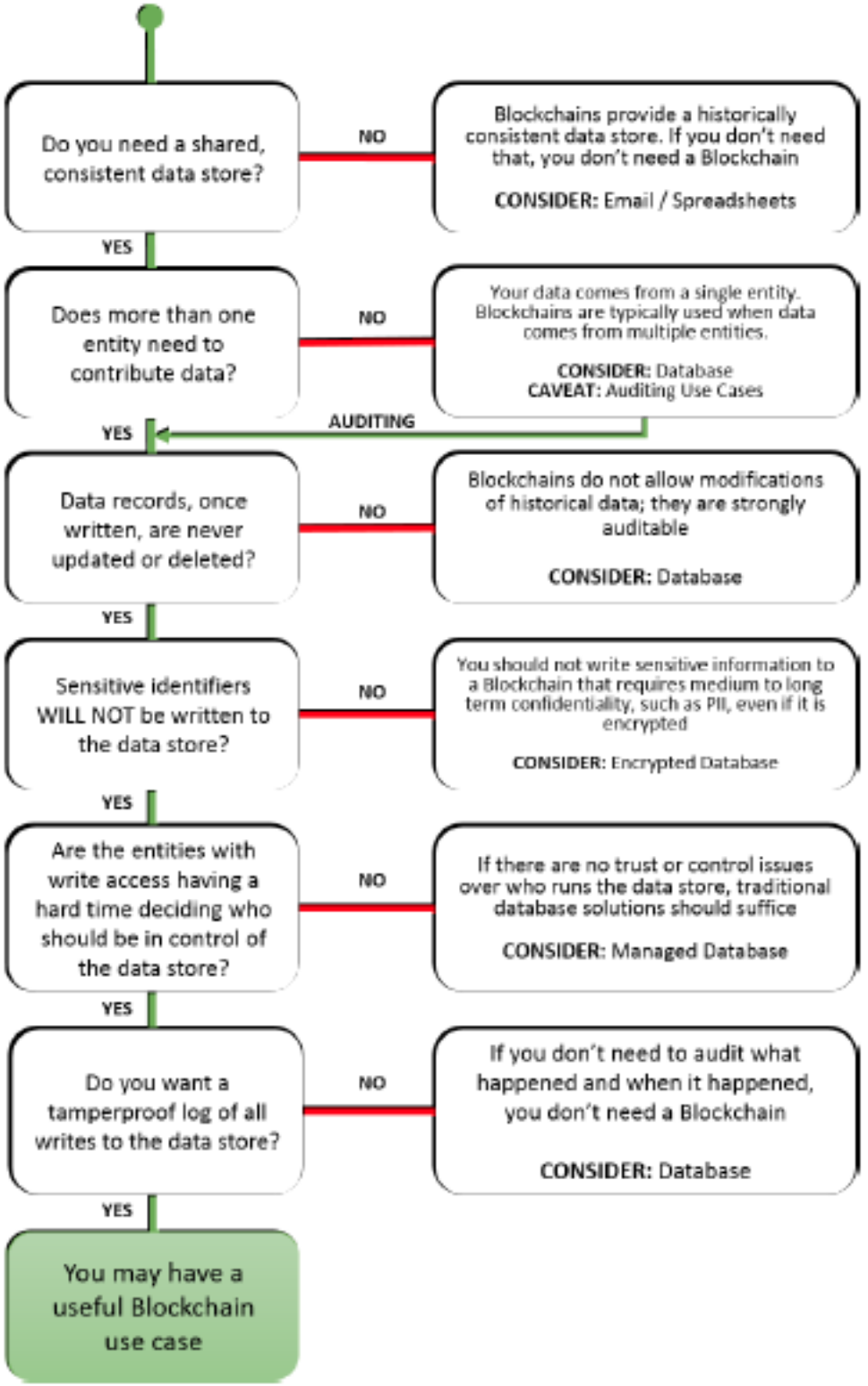
P2P Network

Before a full node can validate unconfirmed transactions and recently-mined blocks, it must download and validate all blocks from block 1 (the block after the hardcoded genesis block) to the current tip of the best blockchain. This is the **Initial Block Download (IBD)** or initial sync.

Block Broadcasting: when a miner discovers a new block, it broadcasts the new block to its neighbor peers.

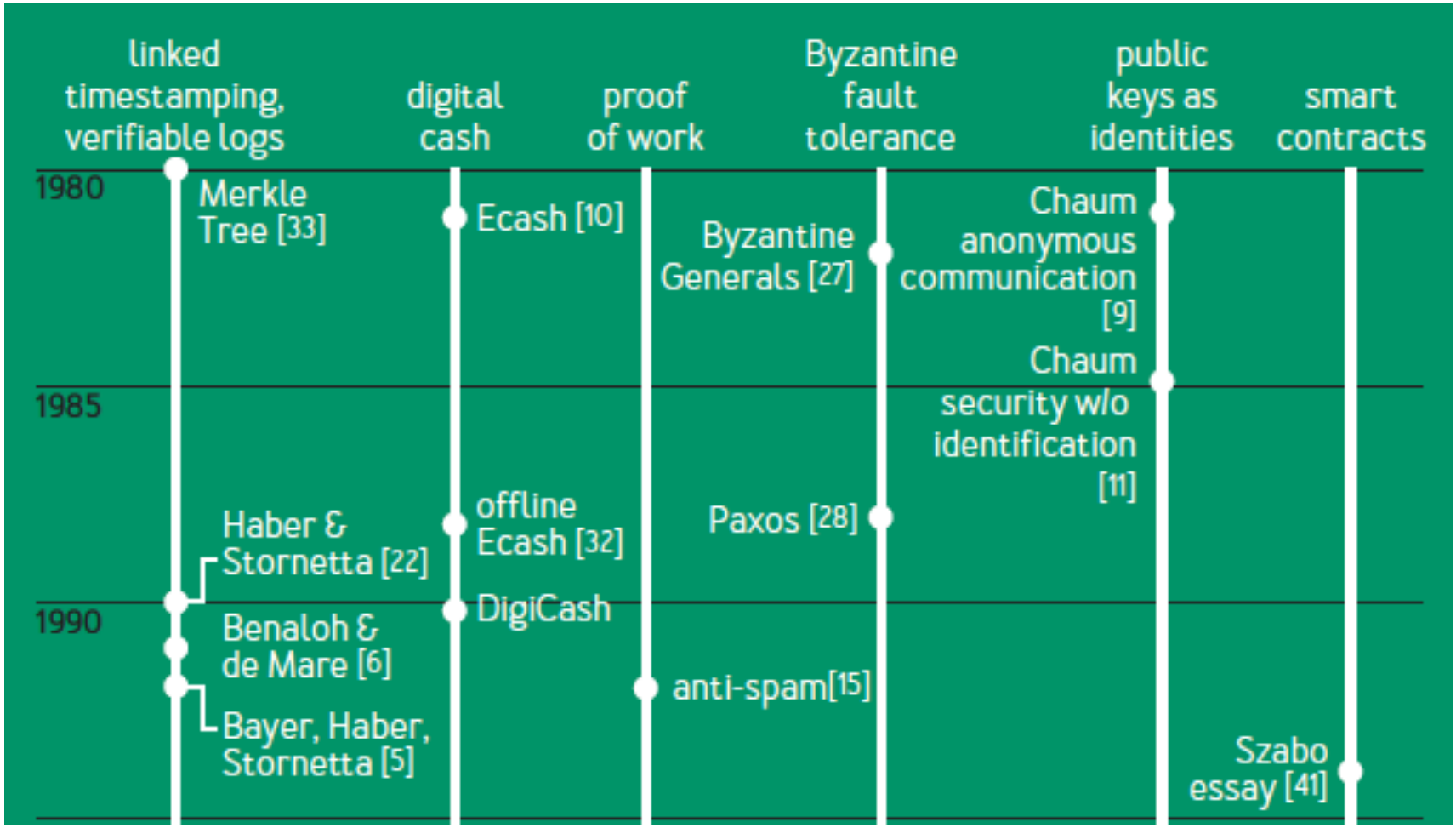
Transaction Broadcasting is similar to block broadcasting, but concerning transactions.

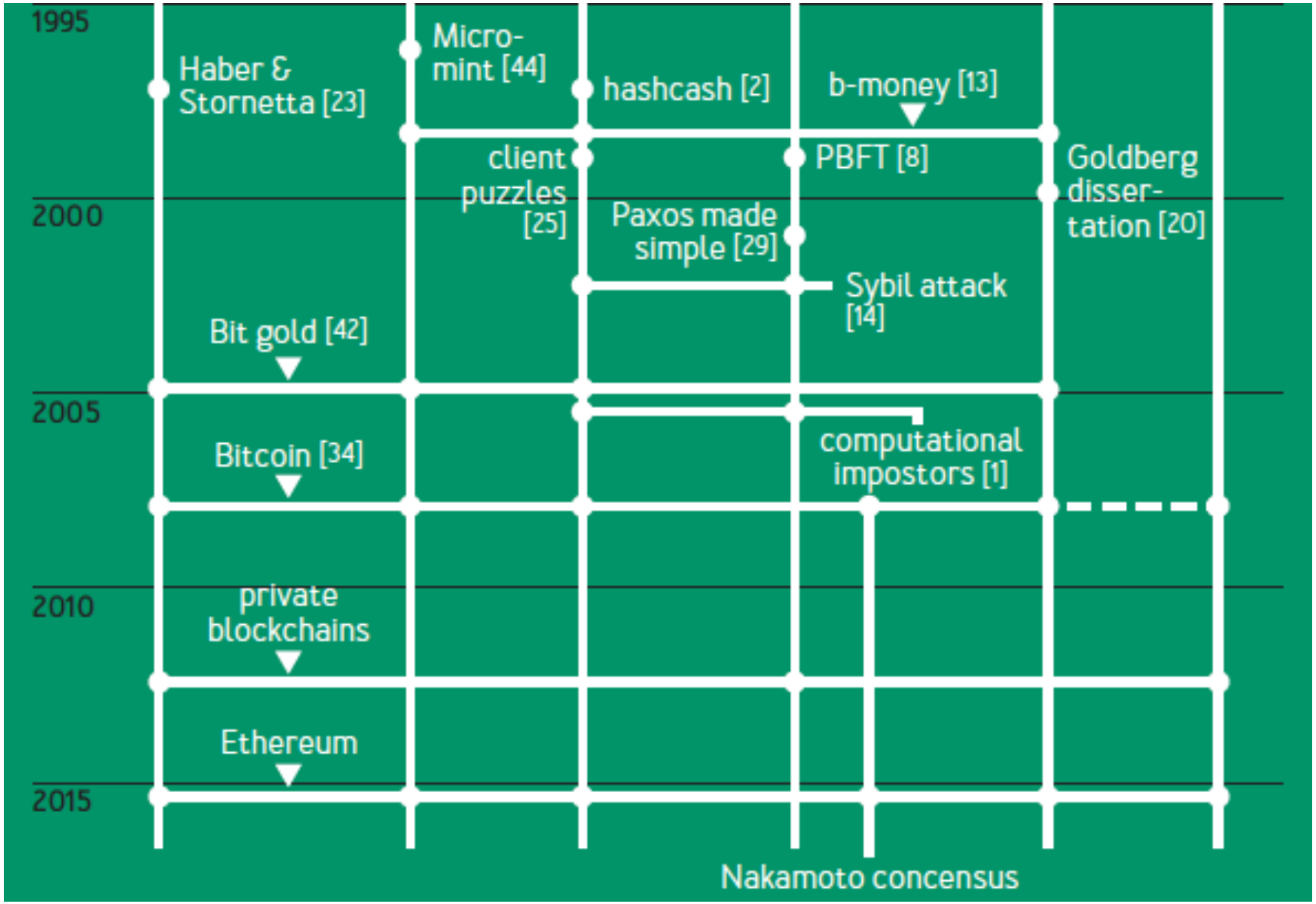
Why should I use a blockchain?



Source:
Blockchain Technology Review,
NIST, October 2018

Academic pedigree




1995
2000
2005
2010
2015
Haber & Stornetta [23]
Micro-mint [44]
hashcash [2]
b-money [13]
client puzzles [25]
PBFT [8]
Goldberg disser-tation [20]
Paxos made simple [29]
Sybil attack [14]
Bit gold [42]
Bitcoin [34]
computational impostors [1]
private blockchains
Ethereum
Nakamoto concensus

Coin vs. Token

- Speaking about **cryptocurrency** associated to a blockchain, we need to discern between:

  - **coin**: the blockchain's own units of virtual currency, for transaction purposes

  - **token**: secondary units that reside in a blockchain, with various purposes

- There are two types of tokens:

  - **utility token**: represents the right to get products/services from the token emitter

  - **security token**: digital asset whose value derives from a tradable asset; subsequently, it is liable to government laws

Security Token

A security token represents traditional, private security interest. It could represent a share in a company, an LP interest in a fund or a trust, a member share in an LLC. Essentially, you're taking something that today you have on paper and you're putting an electronic wrapper around it.

Joshua Stein, CEO of tokenized securities startup Harbor

LP = Limited Partnership
LLC = Limited Liability Company

Transactions

- transactions describe payments for goods and services, made by means of **coins**

- parties are identified by their public keys

- every payment must be digitally signed

Bitcoin

Genesis block: 3 January 2009
Developers: "Satoshi Nakamoto", Gavin Andresen, Wladimir van der Laan

- **1 BTC = 10^8 satoshi**

The Bitcoin Core software includes a transaction verification engine and connects to the network as a full node.

https://github.com/bitcoin/bitcoin/

https://bitcoin.org
https://bitcoin.org/bitcoin.pdf
https://en.bitcoin.it/wiki/Bitcoin_Improvement_Proposals
https://charts.bitcoin.com/bch/

Bitcoin

Blockchain-based virtual currency is usually generated in a decentralized and competitive process (i.e., the **mining**).

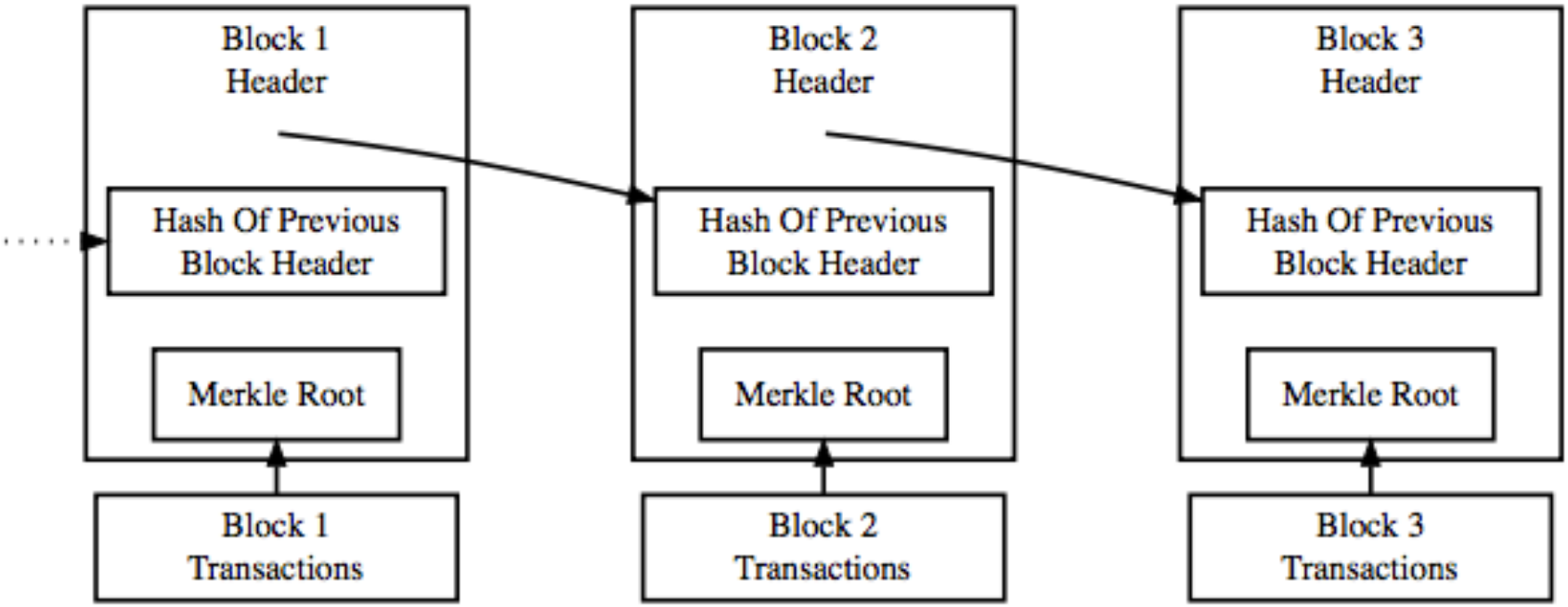
Bitcoin miners are full nodes that process transactions and secure the network using specialized hardware. They collect new bitcoins in exchange.

Bitcoin

https://bitcoin.org/en/faq#economy

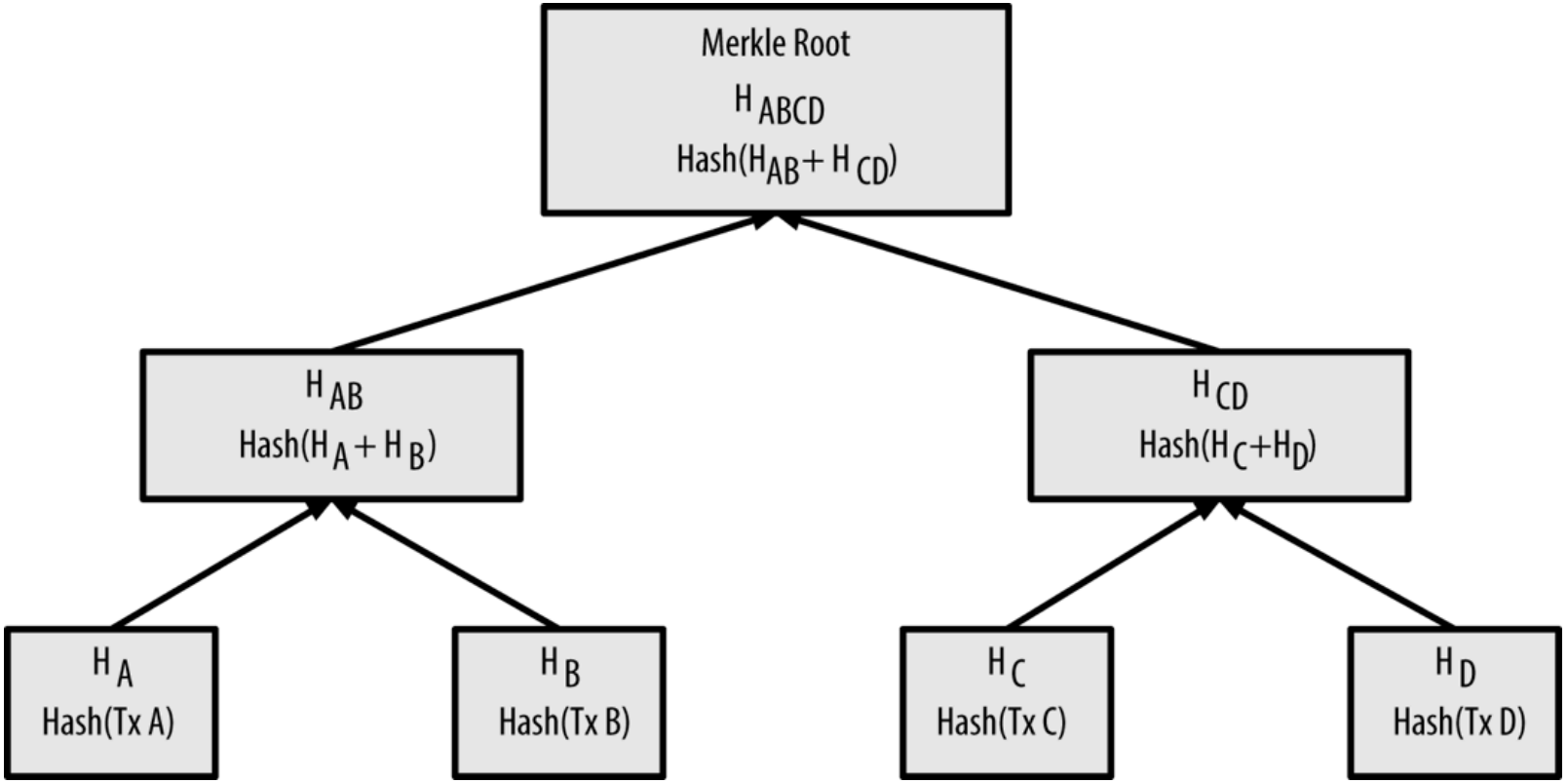
There is only **a limited number of bitcoins in circulation** and new bitcoins are created at a predictable and decreasing rate, which means that demand must follow this level of inflation to keep the price stable.
Because Bitcoin is still a relatively small market compared to what it could be, it does not take significant amounts of money to move the market price up or down, and thus **the price of a bitcoin is still very volatile**.

Bitcoin



Each block's transactions are hashed in a **Merkle Tree**, until a unique hash (the **Merkle Root**) is produced and stored in the block's header, together with the hash of the previous block header.

Merkle Tree



A **Merkle Tree** is constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains, the **Merkle Root**.

Bitcoin transactions

- bitcoin transactions are linked together
- **input:** reference to an output of a previous transaction
- **output:** amount of virtual currency to be transferred
- each one has one or more inputs and one or more outputs
- if output > total input, the transaction is not valid
- if input = 50 but the payer wants to send only 25 to the receiver, there will be two outputs worth 25 (payment + “change” for the payer)

Bitcoin transactions

- the receiver sends his/her public key to the payer

- the payer creates the transaction, specifying the output and the "signature script" that includes the hash of the public key of the receiver

- the payer signs the transaction with its private key

- the transaction is sent by the payer to the network of full nodes

- the full nodes verify and add the transaction, if valid, to the blockchain

Smart Contract

- concept introduced by Nick Szabo in 1994

- **program stored into the blockchain** (created with special transactions; provided with a unique address)

- expression of a **contractual logic**

- can implement **any algorithm**

- deterministic behavior
(but pay attention to concurrent executions!)

- can interact with other smart contracts

Smart Contract

- exposes an **interface**
- may **return data** or **store data**
- interaction based on **message passing**
- messages may represent events of interest for the contract (e.g., the car instalment has been paid)
- interactions with smart contract are stored into the blockchain like transactions, thus they can be **traced**

Smart Contract

- **DApp** (Decentralized Application): frontend + decentralized logic

- **ICO** (Initial Coin Offering): crowdfunding for a single project, in exchange for ICO-specific tokens

- **DAO** (Decentralized Autonomous Organization): autonomous organization ruled by a smart contract

Concurrency and Smart Contracts

There are remarkable similarities between multi-transactional behaviors of smart contracts in cryptocurrencies such as Ethereum and classical problems of shared-memory concurrency.

- ***contracts-as-concurrent-objects* analogy**: accounts using smart contracts in a blockchain are like threads using concurrent objects in shared memory

- relation between observable contract behaviors and well-studied concurrency topics, such as atomicity, interference, synchronization, and resource ownership

I. Sergey, A. Hobor, *A Concurrent Perspective on Smart Contracts*, Financial Cryptography and Data Security, ed. Springer, 2017

Ethereum

First version: 22 July 2014
Main designer: Vitalik Buterin
https://www.ethereum.org

Objective: provide a flexible, shared and secure *World Computer*.

Ethereum allows developers to program and deploy **smart contracts** on the blockchain, to enable complex transactions that go far beyond the mere transfer of virtual currency.

1 ETH = 10^9 GWei = 10^{18} Wei
https://ethereumprice.org

Parity: most used wallet

Geth: Go implementation of an Ethereum client
https://github.com/ethereum
https://geth.ethereum.org/docs/interface/merge

Ethereum

In Ethereum, the total supply of ether and its rate of issuance was decided by the donations gathered on the 2014 presale. The results were roughly:

- **60 million ether created to contributors of the presale**
- 12 Million (20% of the above) were created to the development fund, most of it going to early contributors and developers and the remaining to the Ethereum Foundation
- The current **mining reward** is **2 ether per block** plus **all transaction fees of the transactions contained in the block**.

Ethereum

Evolution of ether price:

Ethereum accounts

There are two types of accounts in Ethereum:

- *normal accounts*, controlled by public-private key pairs (i.e., human users)
- *contract accounts*, controlled by their own internal code

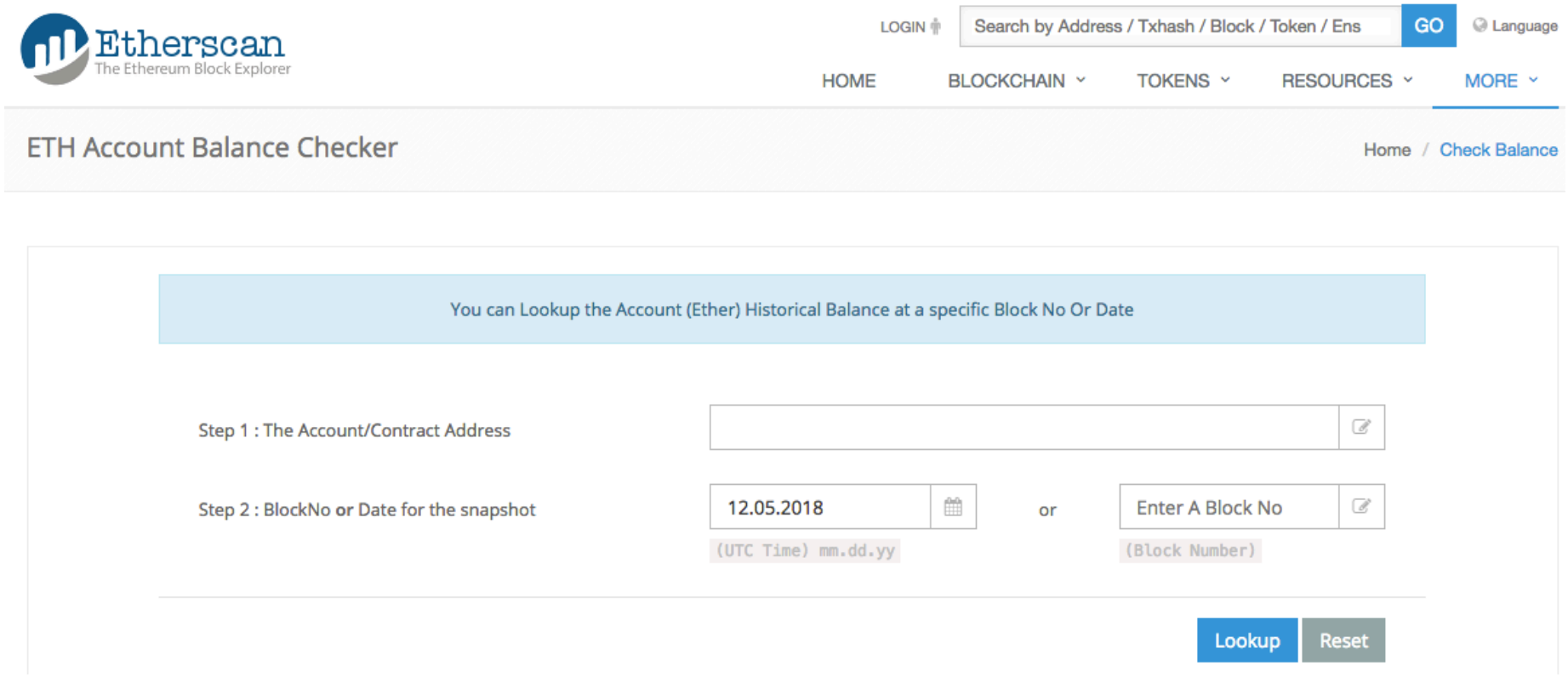
Ethereum uses the **Elliptic Curve Digital Signature Algorithm (ECDSA)**, with elliptic curve **secp256k1**, to validate the origin and integrity of messages.

Private key (256 bit) and public key (512 bit) are generated when a new blockchain account is created.
Best practice: hardware wallet to securely store the private key.

Account = last 20 bytes of hash **Keccak-256**(public key)

Ethereum account balance

Ethereum has accounts that have balances.
Bitcoin just has transactions.



Smart Contracts in Ethereum

Each smart contract has a **unique address**, assigned by means of a **special transaction**.

After such a transaction, the smart contract becomes an immutable part of the blockchain.

Interaction: signed messages from an account to the address of the smart contract, and vice versa.

Smart contracts are stateful entities, with **storage** capabilities.

Smart Contracts in Ethereum

Source code written with **Solidity** (similar to JavaScript)
--> compiled to a **bytecode** of low-level instructions (**opcodes**)
--> executed on the **Ethereum Virtual Machine (EVM)**

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.

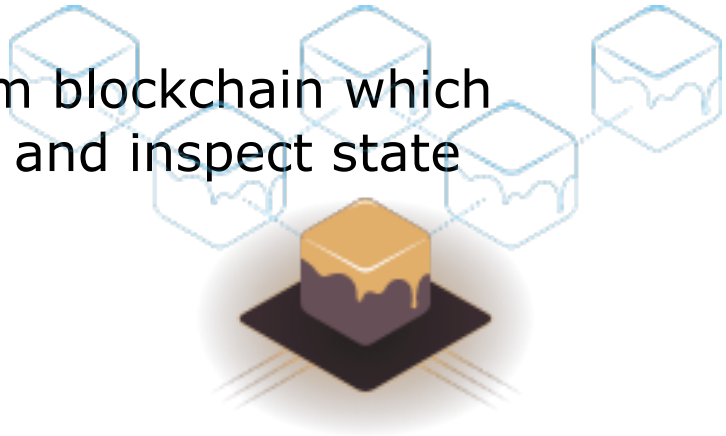
Smart Contracts in Ethereum

Kovan, Ropsten: testnets

Truffle: development framework, allowing the developer to test smart contracts on a local EVM and to release them on the real Ethereum blockchain, on an online testnet, or on a local one

https://www.trufflesuite.com/truffle

Ganache: Quickly fire up a personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates.



https://www.trufflesuite.com/ganache

Other tools: https://ethereum.org/developers/#other-tools

Smart Contracts in Ethereum

Example: **ballot box**

The owner initializes the contract with a list of proposals and gives the right to vote to a set of Ethereum addresses.

Voters cast their votes for a particular proposal, which they may do only once.

```
1  contract Ballot {
2    mapping(address => Voter) public voters;
3    // more state definitions
4    function vote(uint proposal) {
5      Voter sender = voters[msg.sender];
6      if (sender.voted)
7        throw;
8      sender.voted = true;
9      sender.vote = proposal;
10     proposals[proposal].voteCount += sender.weight;
11   }
12   // more operation definitions
13 }
```

https://solidity.readthedocs.io/en/develop/solidity-by-example.html

Smart Contracts in Ethereum

The DAO bug: in The DAO's smart contract, the splitDAO() function was vulnerable to the **recursive send pattern**.

```
1010    // Burn DAO Tokens
1011    Transfer(msg.sender, 0, balances[msg.sender]);
1012    withdrawRewardFor(msg.sender); // be nice, and get his rewards
1013    totalSupply -= balances[msg.sender];
1014    balances[msg.sender] = 0;
1015    paidOut[msg.sender] = 0;
1016    return true;
1017  }
```

On June 17, 2016, a hacker used this loophole to drain funds from The DAO. In the first few hours of the attack, 3.6 million ETH were stolen, the equivalent of $70 million at the time. Once the hacker had done the damage he intended, he stopped the attack.

http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/

Transaction Fees - Type 0

TX Fee = Gas Used * Gas Price

Example transactions for a smart contract execution:

	Gas consumed by operation	Gas remaining
Start of transaction		150
STORE 31	45	105
ADD 2 numbers	10	95
STORE sum	45	50
End of transaction	The number 31 and the sum is stored and written to the blockchain.	

Transaction Fees - Type 0

- **Gas Used** = how many units of Gas have been used for the transaction
- **Gas Limit** = how many units of Gas the user is willing to pay for
- **Gas Price** = price of a Gas unit, decided by the user; the higher the Gas Price, the sooner the transaction will be included in a block by the miners

The current average Gas Price changes continuously.

If Gas Limit is low, risk of running **Out of Gas**.

Block Gas Limit (BGL) = max amount of Gas a block can include. Current BGL is 30M.

Exceeding Gas is refunded.

Transaction Fees - Type 0

Miners seek to **maximize their profit** within each block.

Miners are able to determine the actual gas each tx will consume, i.e., the **Gas Used**.

Miners compute **Gas Price * Gas Used** for each given transaction and ranks the transactions based on their value.

Miners also perform several controls on each transaction, such as:

- Gas Used ≤ Gas Limit
- Gas Limit < BGL

Transaction Fees - Type 2

https://www.blocknative.com/blog/eip-1559-fees

The Ethereum Improvement Proposal **EIP-1559** (introduced in 2021) changed how Ethereum transaction fees are calculated and where fees go.

Instead of a singular Gas Price, there are:

- The Base Fee, which is adjusted up and down by the protocol based on how congested the network is. And is subsequently burned.
- A Max Priority Fee, which is optional, determined by the user, and is paid directly to miners.
- The Max Fee Per Gas, which is the absolute maximum the user is willing to pay per unit of gas to get the transaction included in a block. For brevity and clarity, we will refer to this as the Max Fee.

Transaction Fees - Type 2

The Base Fee targets 50% full blocks and is based upon the contents of the most recent confirmed block. Depending on how full that new block is, the Base Fee is automatically increased or decreased.

Example:

- If the last block was exactly 50% full, the Base Fee will remain unchanged.
- If the last block was 100% full, the Base Fee will *increase* by the maximum 12.5% for the next block.
- If the last block was more than 50% full but less than 100% full, the Base Fee will *increase* by less than 12.5%.
- If the last block was 0% full – that is, empty – the Base Fee will *decrease* the maximum 12.5% for the next block.
- If the last block was more than 0% full but less than 50% full, the Base Fee will *decrease* by less than 12.5%

Transaction Fees - Type 2

The minimum gas price for a transaction is the current Base Fee.

However, what if the Base Fee increases while the transaction is pending?

The transaction will become underpriced and be at risk of becoming stuck. Or failing. Or getting dropped.

Transaction Fees - Type 2

Regarding for predictable transaction settlement under EIP-1559, it is currently considered best practice to set a Max Fee that anticipates such an increase in the Base Fee.

But by how much? And why?

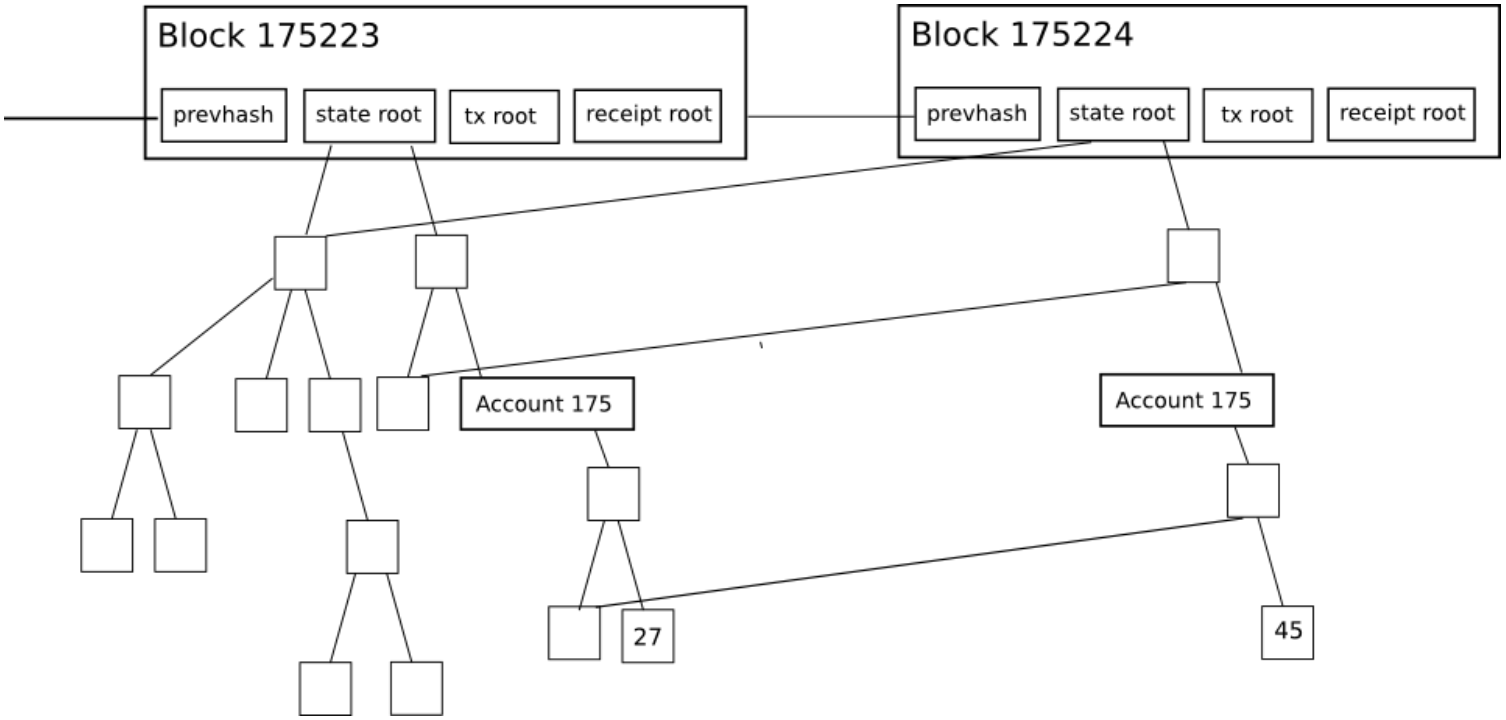
Blocknative's ETH gas fee estimator *currently* uses the following simple heuristic to calculate the recommended Max Fee for any given Base Fee and Max Priority Fee combination:

Max Fee = (2*Base Fee) + Max Priority Fee

Doubling the Base Fee when calculating the Max Fee ensures that your transaction will remain marketable for six consecutive 100% full blocks.

Merkle proofs in Ethereum

In Ethereum, the state is an enormous data structure called a **modified Merkle Patricia Trie**. Every block header contains not just one Merkle tree, but three trees (for transactions, receipts, state).



Merkle proofs in Ethereum

Bitcoin uses binary Merkle Trees, which are very good data structures for authenticating information that is in a "list" format; essentially, a series of chunks one after the other.

In Ethereum, more complex information has to be stored.

Transaction tree: records transaction requests.

Receipt tree: records transaction outcomes.

State tree: records the state of the accounts.

The state in Ethereum essentially consists of a key-value map, where the keys are addresses and the values are account declarations, listing the balance, nonce, code and storage for each account (where the storage is itself a tree).

Merkle proofs in Ethereum

The modified Merkle Patricia Trie is a data structure where

- we can quickly calculate the new tree root after an insert, update edit or delete operation, without recomputing the entire tree

- the depth of the tree is bounded

- the root of the tree depends only on the data, not on the order in which updates are made

Consensus models

A key aspect of blockchain technology is **determining which user publishes the next block**. This is solved through implementing one of many possible consensus models.

For permissionless blockchain networks there are generally many publishing nodes competing at the same time to publish the next block.

They usually do this **to win cryptocurrency**. They are generally **mutually distrusting users** that may only know each other by their public addresses.

Each publishing node is likely motivated by a desire for **financial gain**, not the well-being of the other publishing nodes or even the network itself.

Consensus models

The following properties are then in place:

- The initial state of the system is agreed upon (e.g., the **genesis block**).

- Users agree to the consensus model by which blocks are added to the system.

- Every block is linked to the previous block by including the previous block header's hash digest (except for the first 'genesis' block, which has no previous block and for which the hash of the previous block header is usually set to all zeros).

- Users can verify every block independently.

Consensus models

Most common consensus models:

- Proof of Work
- Proof of Stake
- Round Robin
- Proof of Authority / Proof of Identity
- Proof of Elapsed Time

Proof of Work (PoW)

- full nodes compete to create next stable block of the blockchain

- execution of a moderately complex job (feasible, anyway), easily verifiable

- PoW requires specialized hardware (**ASIC**: Application-Specific Integrated Circuit) https://www.bitmain.com

Proof of Work (PoW)

- calculate the **hash** of some data (including a pseudorandom component)

- repeat until the result is less or equal a certain **threshold**

- the full node that completes first, proposes its block for being added to the blockchain

- for each consolidated block, the full node receives a **reward** in terms of coins

- the threshold is periodically recomputed to guarantee that the required computing time remains almost constant (a few minutes)

Proof of Work (PoW)

In **Bitcoin**, a **SHA-256 hash of merkle root + previous block header + random nonce** is repeatedly computed, until the result does not exceed a certain threshold.

New blocks will only be added to the blockchain if their hash is at least as challenging as a difficulty value expected by the consensus protocol.

Every N blocks (N=2016 in Bitcoin), the network computes the number of seconds elapsed between generation of the first and last of those last N blocks. The threshold is then adjusted based on the elapsed time.

Proof of Work (PoW)

In **Ethereum 1.0** (used until 15/9/2022), the PoW algorithm is called **Ethash**.

The mining function combines and hashes (with Keccak-256) a random nonce, a hash of the block header and random slices from the data set.

The function loops until a value that satisfies the difficulty threshold is found.

https://ethdocs.org/en/latest/mining.html

Proof of Work (PoW)

The miner who successfully hashes a block header to a value satisfying the difficulty threshold gets the fee, adds the entire block to the blockchain and spreads it to other full nodes.

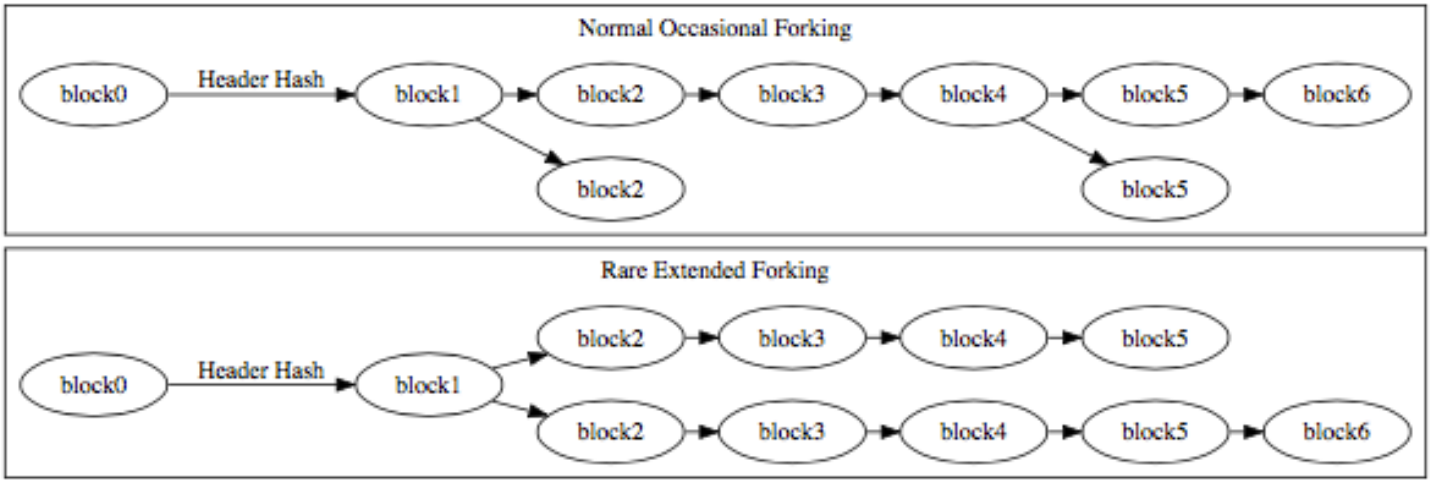
It is noteworthy that including a large volume of transaction data in a block does not slow down hashing with extra I/O.

51% Attack: the ability of someone controlling a majority of network **hash rate** to revise the transaction history and prevent new transactions from confirming.

Proof of Work (PoW)

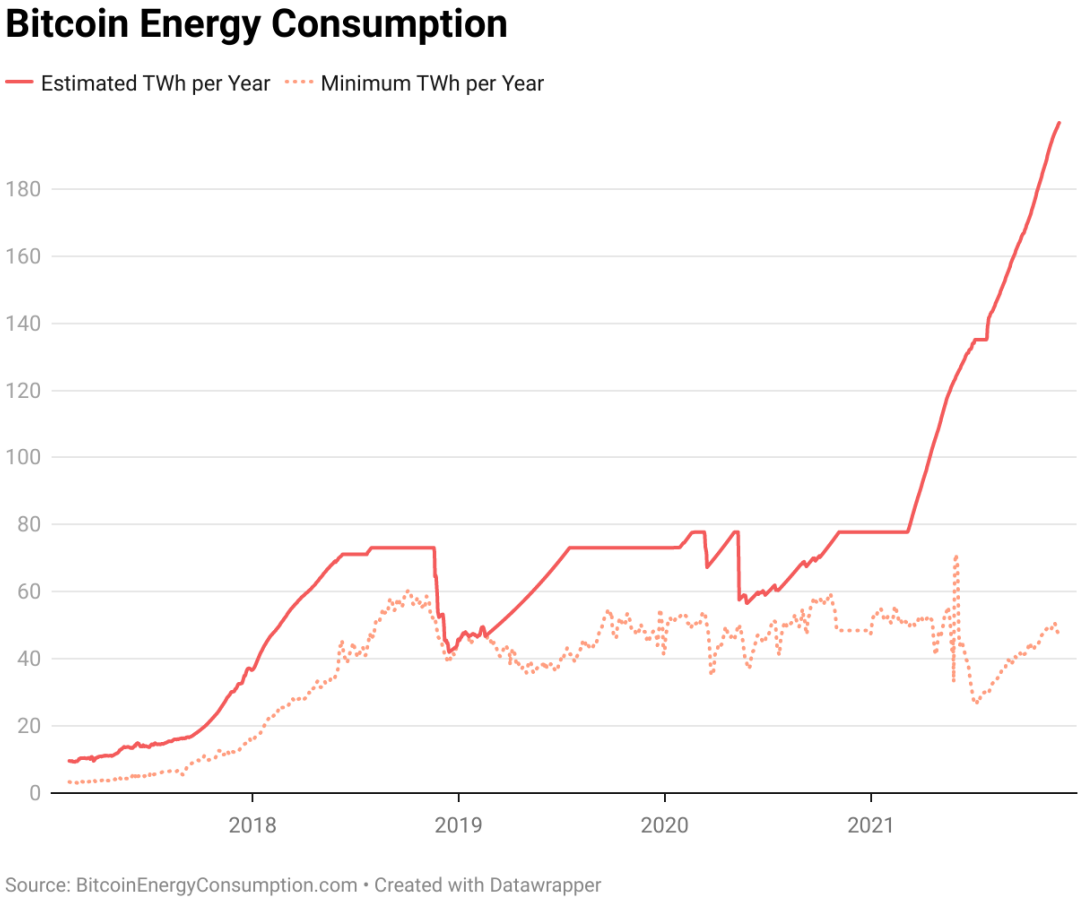
When two or more miners each produce a block at roughly the same time. This creates an apparent **fork** in the blockchain.

Eventually a miner produces another block which attaches to only one of the competing simultaneously-mined blocks. This makes that side of the fork stronger than the other side. Assuming a fork only contains valid blocks, normal peers always follow the most difficult chain to recreate and throw away **stale blocks** belonging to shorter forks.



Proof of Work (PoW)

PoW is very expensive in terms of energy.



Proof of Stake (PoS)

With PoS, the blockchain maintains ownership of a population of coins staked by participants.

To add a new block, a participant is **elected**. The chance of being elected is proportional to the size of the **stake**.

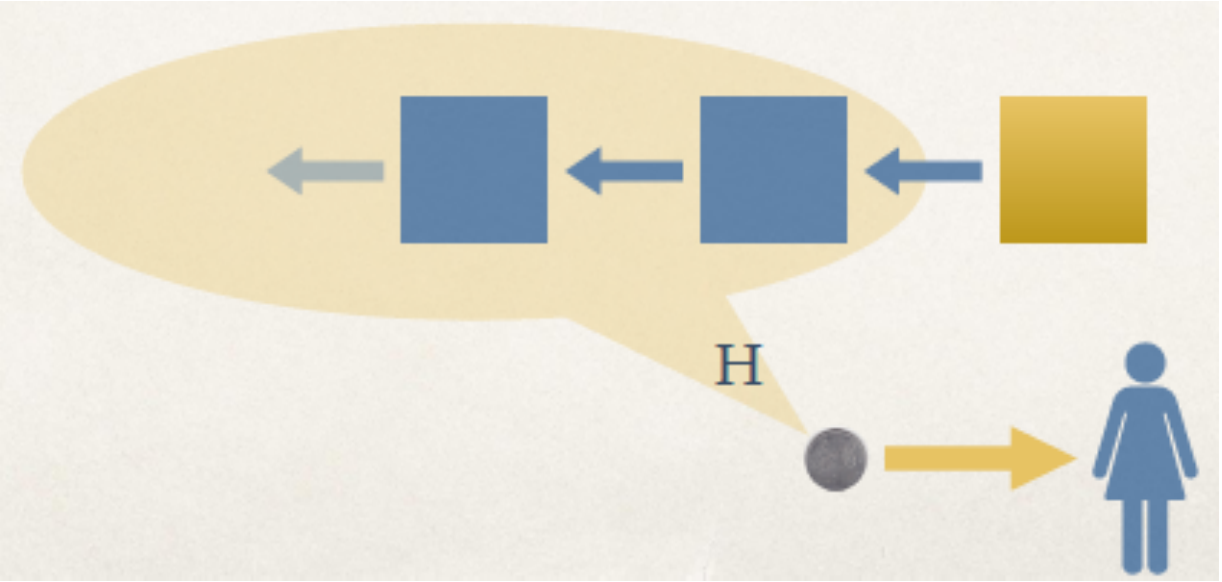
Advantages:
- reduced risk of centralization
- energy efficiency

Problems:
- electing a participant requires **randomness**
- possible security breaches (e.g., *nothing at stake*)

Proof of Stake (PoS)

An idea:

- the blockchain already contains random data (e.g., block hashes)
- obvious approach --> just hash the current blockchain!

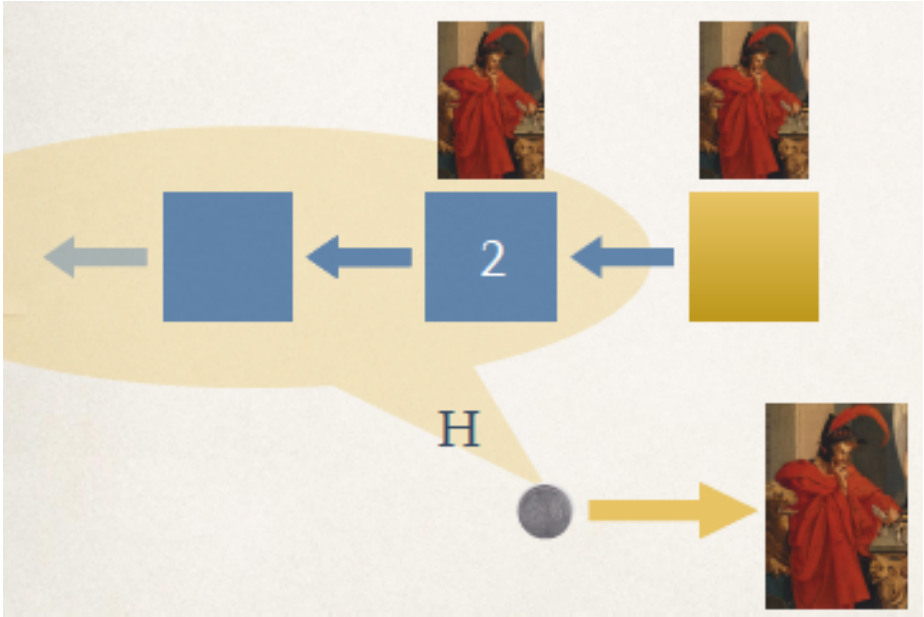


Proof of Stake (PoS)

It turns out that this is a bad idea!

- an adversary could use **rejection sampling** (a basic technique used to generate observations from a distribution)

- if allowed to create a block, the adversary can (privately) make several attempts to create a block, until it finds the one that guarantees, by hashing the updated blockchain, that next elected will be the adversary itself, again



"grinding" attack

Eventual-consensus PoS protocols

Eventual-consensus protocols apply some form of a longest-chain rule to the blockchain. The immutability of a block increases gradually with the number of blocks created on top of it.

Examples:

(with rigorous guarantees)

- **Ouroboros**
- Snow White
- **Ouroboros Praos**
- Ouroboros Genesis

(in the wild)

- Peercoin https://peercoin.net/
- NXT https://nxtplatform.org/

Blockwise-BA PoS protocols

Blockwise-BA protocols achieve the immutability of every single block via a full execution of a *Byzantine Agreement (BA)* protocol before moving on to production of any subsequent block.

Examples:

(with rigorous guarantees)

- **Algorand** https://www.algorand.com/

(in the wild)

- **Casper** https://casperlabs.io/
- **Ethereum** after "The Merge"

Algorand

Testnet launch: 20 July 2018
Founder: Silvio Micali
Chief Scientist: Jing Chen

https://www.algorand.com/
https://developer.algorand.org/

Each new block is generated via the ***BA**** protocol, which is a simple, highly efficient *Byzantine Agreement* protocol.

The players of *BA** are chosen to be a much smaller subset of all nodes. Each of them proposed a block of transactions. Then they converge to one block.

Algorand has **smart contracts**!

J. Chen, S. Micali, ALGORAND, https://arxiv.org/abs/1607.01341, v9 (2017)
Y. Gilad et al., Algorand: Scaling Byzantine Agreements for Cryptocurrencies (2017)

Ethereum after "The Merge"

The Merge:15 September 2022
Main designer: Vitalik Buterin
https://www.ethereum.org

Validators opt to run validator nodes by staking 32 ETH or sending the crypto in a staking wallet.

The algorithm will choose, at random, who should create a block and check and confirm transactions of a given block.

Obviously, the randomness favors those with the most amount of ETH. The validators propose blocks and those are then attested by other validators.

https://ethereum.org/en/upgrades/merge/

Nothing at Stake problem

- a situation where someone loses nothing when behaving badly, but stands to gain everything

- in PoS, validators compete to propose next block; leader election is based on stake

- a **fork** in the chain may occur in case of
  - malicious attack
  - two winners are chosen

- optimal strategy for validators: compete on both chains, for building next block

- an adversary may intentionally fork the chain, to perform **double-spending**

Long-range attacks

- related to the concept of **"costless simulation"** (ability to create up-to-date branches without effort)

- refers to the ability of a minority set of stakeholders to execute the blockchain from the genesis block and **produce a valid alternative history of the system**

- two classes of long-range attacks:

  - **posterior corruption** (the adversary steals the secret keys corresponding to low-stake accounts; the **density** of the alternative blockchain could be indistinguishable from the honestly generated one)

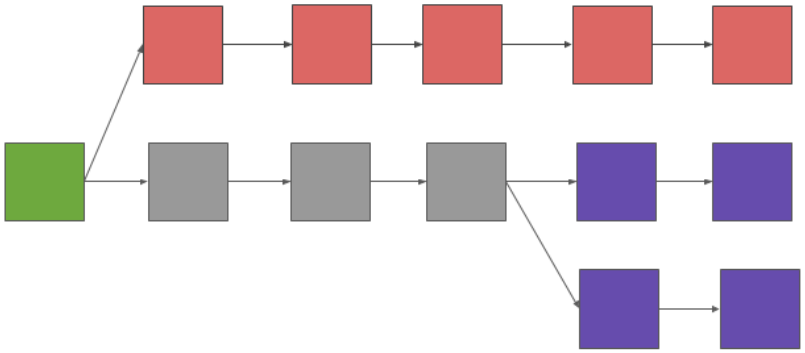
  - **stake-bleeding** (the adversary mounts the attack leveraging transaction fees used as rewards for running the PoS protocol)

How to mitigate long-range attacks

- **checkpoints**:

only the latest K blocks can be reorganized

reorganization of the chain will occur when another branch of the chain becomes longer than the main chain



- **key-evolving cryptography**
- strict **chain density statistics**

P. Gazi, A. Kiayias, A. Russell, Stake-Bleeding Attacks on proof-of-Stake Blockchains, Crypto Valley Conference, Zug, Switzerland (2018)

Blockchains ≈ Merkle hash functions

The structure of a blockchain is very much the same as the Merkle scheme for hash functions, where a compression function is iterated over a message that is processed in blocks:

$m = \{m_1,..,m_n\}$

$H(m) = h_n = H(m_n,h_{n-1})$
..
$h_1 = H(m_1)$

K. Halunen et al., *On the Similarities Between Blockchains and Merkle-Damgard Hash Functions*, IEEE CRS 2018, Lisbon, Portugal

Blockchains ≈ Merkle hash functions

Merkle hash functions are required to satisfy three different high level security properties:

- preimage resistance (given h, hard to find x s.t. $H(x) = h$)
- second preimage resistance (given x_1, hard to find x_2 s.t. $h_1=h_2$)
- collision resistance (hard to find x_1 and x_2 s.t. $h_1=h_2$)

Second preimages are something that blockchain should not have.

However, checkpoints are valuable targets for second preimage attacks.

In **very long timescales** such weaknesses should be considered.

Eclipse attacks

E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on bitcoin's peer-to-peer network, USENIX Security Symposium, Austin, TX, USA (2015)

Y. Marcus, E. Heilman, S. Goldberg, Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network, IACR Cryptology ePrint Archive, (2018)

Eclipse attacks operate at the **peer-to-peer network** layer.

A node under attack obtains a distorted view of the blockchain, as the adversary controls communication channels and applies filters to the messages exchanged with the other nodes.

Summary of possible security breaches

PoW:
- 51% attack

PoS:
- Nothing at stake
- Long-range attacks

General problems:
- Second preimage attacks
- Eclipse attacks

Round Robin consensus model

- used by some permissioned networks
- nodes **take turn** in creating blocks
- block publication **timeout**, to prevent unavailable nodes from halting the blockchain growth process
- lacks cryptographic puzzles
- has low power requirements

Proof of Authority / Proof of Identity consensus model

- publishing nodes must have their **identities proven and verifiable** within the blockchain network

- publishing nodes have a **reputation** to preserve / increase

- the lower the reputation, the less likelihood of being able to publish a block

- applies to permissioned blockchain networks only

Proof of Elapsed Time (PoET) consensus model

- each publishing node requests a wait time from a secure **hardware time source** within their computer system

- the secure hardware time source will generate a **random wait time**

- publishing nodes take the random time they are given and become **idle** for that duration

- once a publishing node wakes up from the idle state, it creates and publishes a block to the blockchain network

- any publishing node that is still idle will stop waiting, and the entire process starts over

- requires a permissioned and trusted execution environment