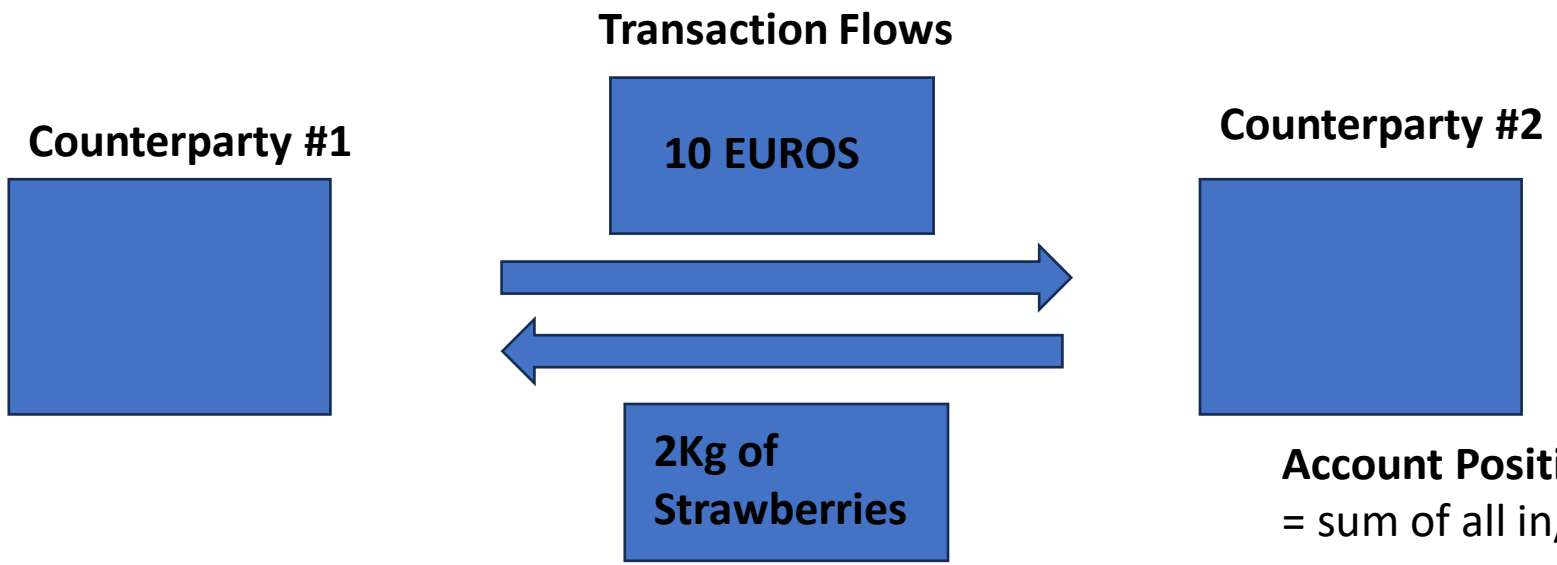
BlockChain & CryptoCurrencies (BitCoin, Ethereum)

arnaud.nauwynck@gmail.com

Traditional e-Trading Database "Transaction (Commit) Log"

Transaction
= Flow exchanges between Counterparties



Account Position
= sum of all in/out

Position (EUR) **-= 10 EUROs**

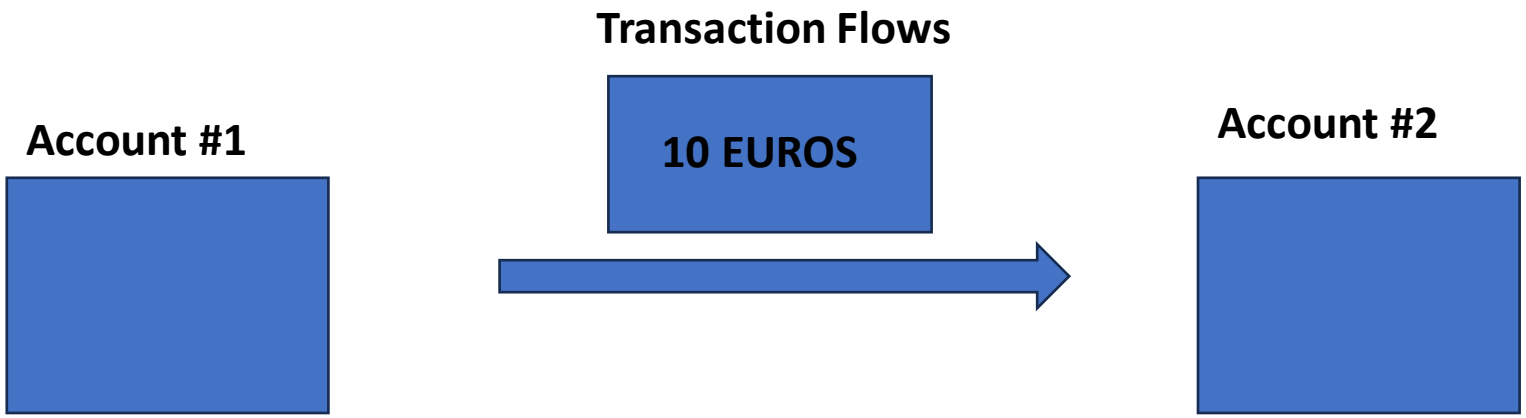
Position(Stawberry) += 2Kg

Account Position
= sum of all in/out

Position (EUR) **+= 10 EUROs**

Position(Stawberry) -= 2Kg

Payment = Flow of Currency only between Accounts of Counterparties



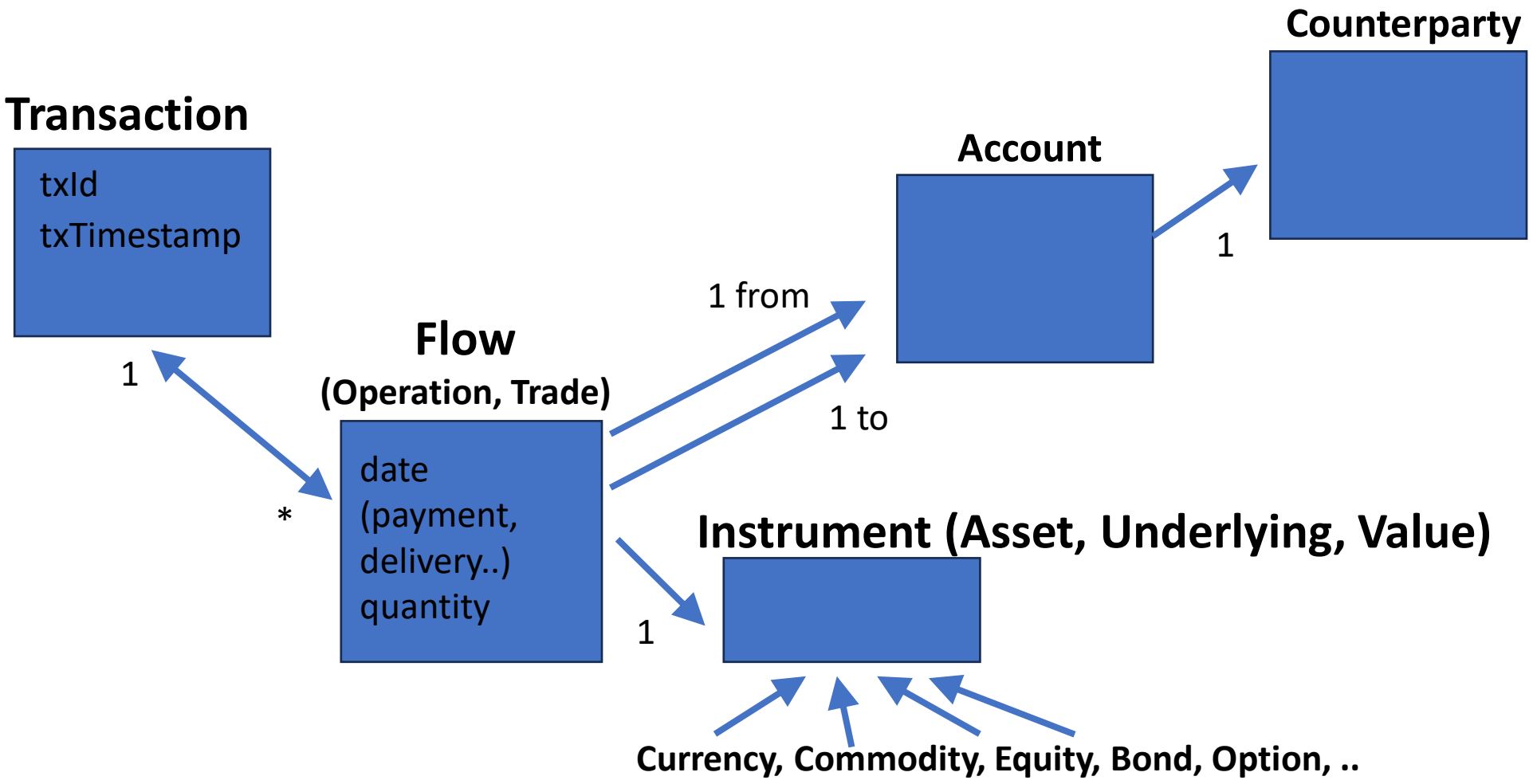
Account Position
= sum of all in/out

Position (EUR) **-= 10 EUROs**

Account Position
= sum of all in/out

Position (EUR) **+= 10 EUROs**

Example Database Tables(Class) Modelisation



Simplified Transaction Log

Transaction Record

at yyyy/mm/dd Account#1 pay 10 EUROs to Account#2

at yyyy/mm/dd Account#25 pay 20 EUROs to Account#7

at yyyy/mm/dd Account#14 pay 600 EUROs to Account#9

at yyyy/mm/dd Account#14 pay 600 EUROs to Account#9

...

Current Committed Transaction Log

??

lock **uncommitted** yet

append-only, then immutable when written
(Time increase)

SQL .. INSERT INTO Transaction VALUES (?,?,?)

begin XA Commit

INSERT INTO Transaction VALUES (?,?,?);

INSERT INTO Flow VALUES (?,?,?,?);
INSERT INTO Flow VALUES (?,?,?,?);
INSERT INTO Flow VALUES (?,?,?,?);

prepare XA Commit
do XA Commit

SQL : re-compute Position from Account

substract quantity :
flow out of account

```
SELECT - sum(f.quantity), f.underlying
FROM   Flow f
WHERE f.from = ?
GROUPBY f.underlying
```

UNION

add quantity :
flow into account

```
SELECT + sum(f.quantity), f.underlying
FROM   Flow f
WHERE f.to = ?
GROUPBY f.underlying
```

Problems of Trust

Confident for ??

1/ keeping ALL history since day 0 (no Data Loss ... can be Terabytes)

2/ keeping ALL past data IMMUTABLE (no alteration of past)

3/ ACID transactions (Atomic, Consistent, Isolated, durable)

4/ Permission to who Append records

5/ Recomputing all position balance accounts correctly (exact millions of addition/substractions)

Full History -> Incremental
Netting, Compensation, Balance per Month ..

You receive your bank account every month
with
"previous month balance"
+ operations of the month
=> "new result month balance"

Behind the scene, Database can be optimized, partitioned, backup, ...

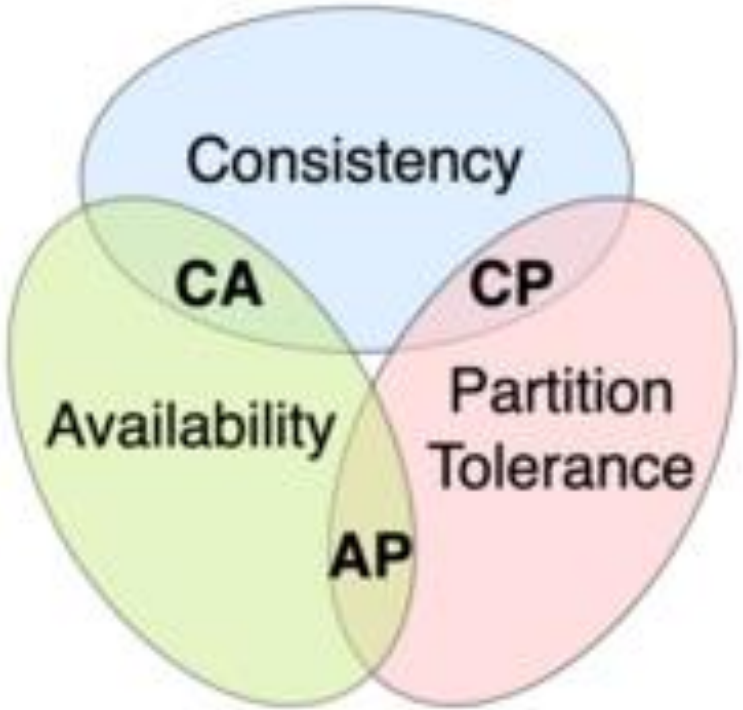
Trust Single Database Owner ?
want Distributed Database ?

The 2-Phase commit protocol is difficult (impossible)
to really be ACID in a **Distributed architecture**

Easy to have a Active-Standby replication
(1 write "master", N readonly synchronized "slaves")

There must be a "master"

CAP Theorem
Consistent- Available - PartitionTolerance
.. choose 2 out of 3



CAP theorem

From Wikipedia, the free encyclopedia

In theoretical computer science, the **CAP theorem**, also named **Brewer's theorem** after computer scientist Eric Brewer, states that any distributed data store can provide only two of the following three guarantees:[1][2][3]

Consistency

Every read receives the most recent write or an error.

Availability

Every request receives a (non-error) response, without the guarantee that it contains the most recent write.

Partition tolerance

The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

CA or CP ?
Not PartitionTolerant, or Not Available ?

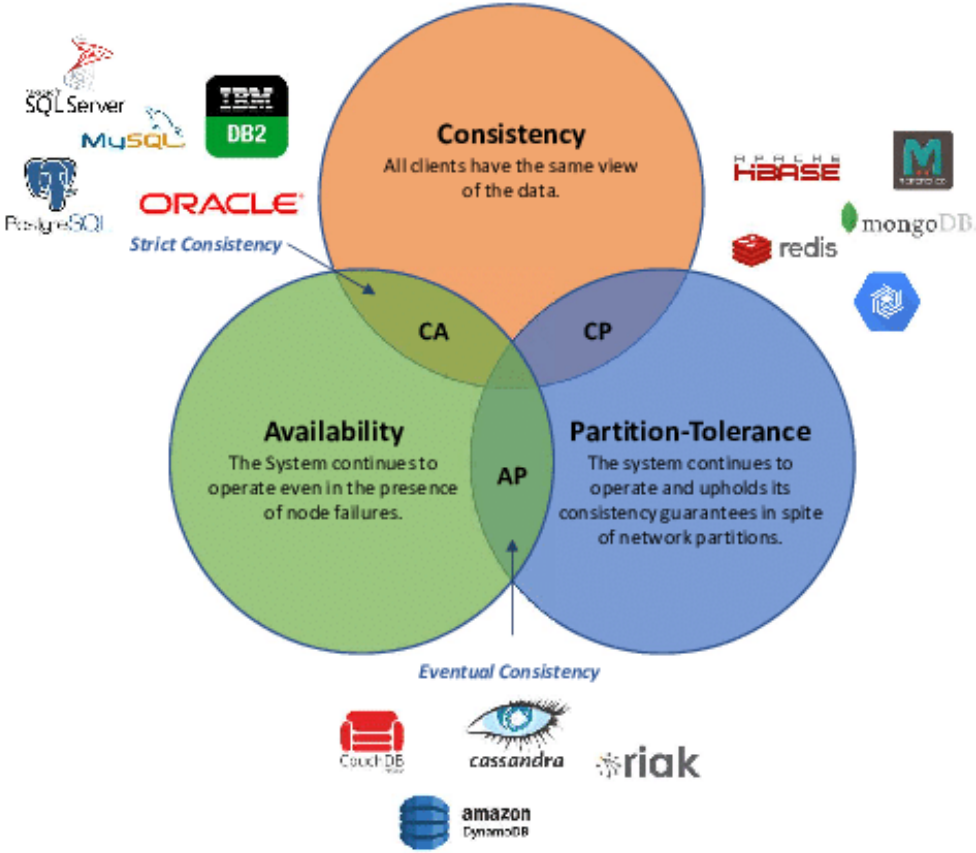
remember the CIA Triad ?
C = Confidential
I = Integrity
A = Available

Eventually Consistent ... means will be consistent when the network will be available

Concensius, Replication, etc...
HA Architecture

All modern Distributed DB are using
- master election protocol (Raft, Gossip, ..) as a Concensius protocol
- replication factors
- choose between CA, CP, AP
  for read-write transaction logs

 preferably
  "Strict Consistent"
or
  "Eventually Consistent"



Link with BlockChain & Crypto currencies ?

at yyyy/mm/dd Account#1 pay 10 EUROs to Account#2

at yyyy/mm/dd Account#25 pay 20 EUROs to Account#7

at yyyy/mm/dd Account#14 pay 600 EUROs to Account#9

at yyyy/mm/dd Account#14 pay 600 EUROs to Account#9

...

Current
Committed
Transaction
Log ... must be PUBLIC
& IMMUTABLE (verifiable)

Append records must be valid
but NO Single Writer
Source of Trust

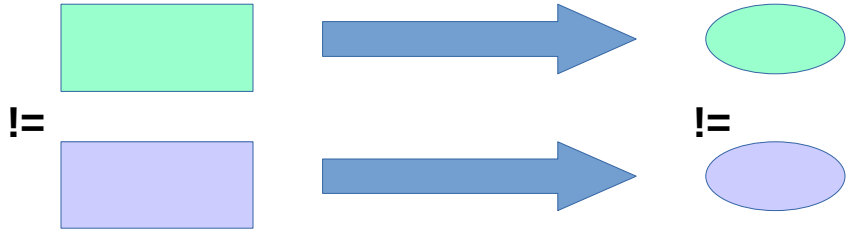
Eventually consistent ??

Reminder Hashing

Hashing …

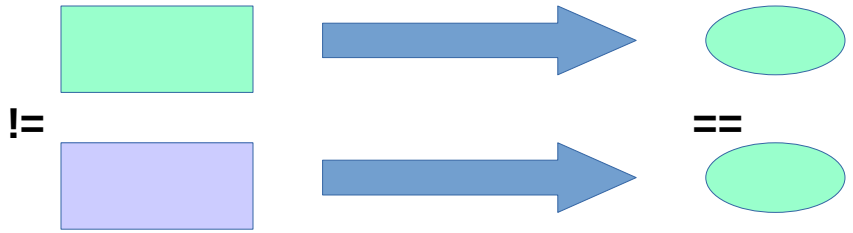
Hashing is One-Way transformation

Hashing different product **usually** give different result



Hashing … must be equi-distributed minimise “Collisions”

A Hashing Collision



Of course if can happen … example in java:
“public int hashCode()”
by default Object.hashCode = System.identityHashCode(x)
= … 64bits pointer (when first hashed) → hashed to 32 bits
= (hashPtr64 ^ (hashPtr64 >>>32))

Hashing : ~~MD5, SHA-1~~, SHA-256, SHA-512 ..

Article Talk Read Edit View h

MD5

From Wikipedia, the fr
(Redirected from Md5

The **MD5 algorithm** hash value. Although cryptographic hash fu vulnerabilities. It can but only against unir

Like most hash funct cracked by brute-forc detailed in the securi

MD5 was designed b function MD4.[3] The RSA license. The abbreviation "MD" stands for "Message Digest."

The security of the MD5 has been severely compromised, with its weaknesses having been exploited in the field, most infamously by the Flame malware in 2012. The CMU Software Engineering Institute considers

Article Talk Read Edit View history

SHA-1

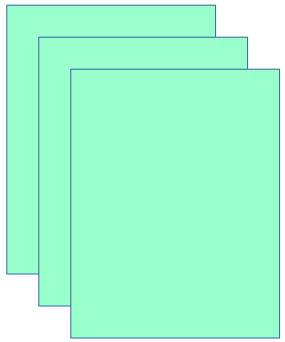
From Wikipedia, the free encyclopedia

In cryptography, **SHA-1** (**Secure Hash Algorithm 1**) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST.[3] SHA-1 produces a 160-bit (20-byte) hash value known as a message digest. A SHA-1 hash value is typically rendered as a hexadecimal number, 40 digits long.

SHA-1 is no longer considered secure against well-funded opponents. In 2005, cryptanalysts found attacks on SHA-1 suggesting that the algorithm might not be secure enough for ongoing use,[4] and since 2010 many organizations have

Hashing for Proof of Ownership/Authenticity

Proof author (ownership) of a file for Copyright



compute HASH

give only HASH
to a depository authority

later..
check HASH from authority
+ data

depositary
/dɪˈpɒzɪt(ə)ri/

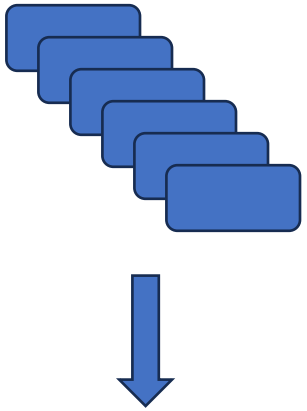
noun

a person to whom something is lodged in trust.

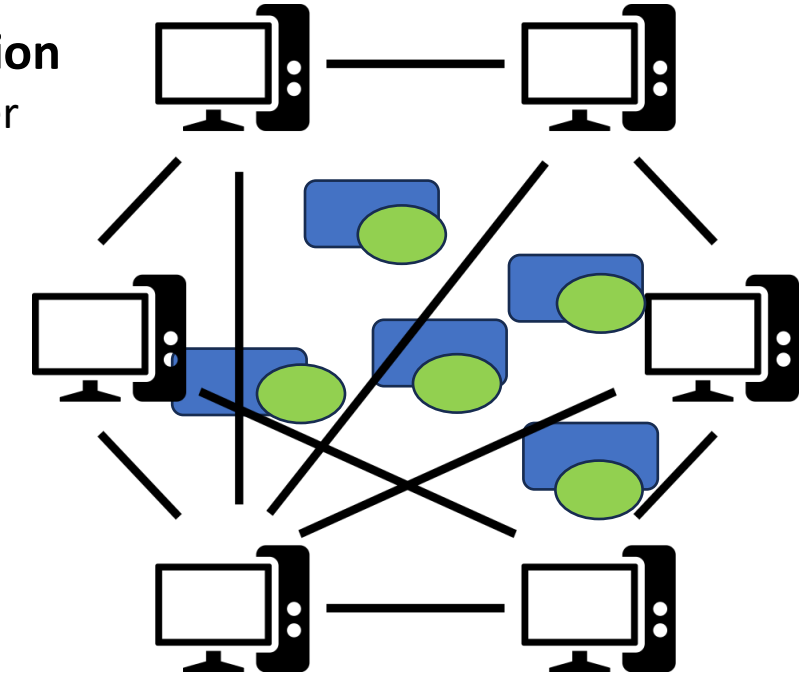
Ensure Immutability Verifiable of Past data
… simple with Block Hashing (+ P2P Replication)

Database, can be HUGE

split Database
in "**Blocks**"



distributed **replication**
ex using Peer-To-Peer



compute all blocks HASH
h1 = hash(block1), h2 = hash(block2), ...

then HASH of HASHes: GlobalHash=hash(h1, h2, ..., hN)

Merkle Tree
Hashing Efficiently Huge Blocks, Blocks of Blocks

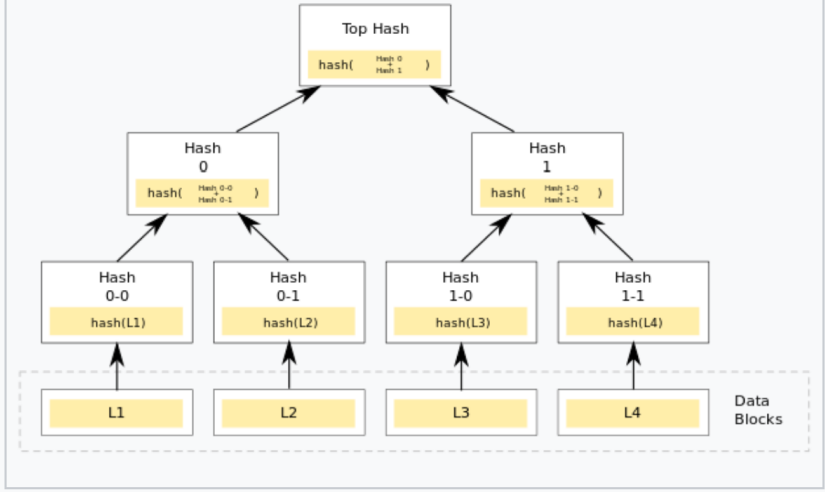
Merkle tree

25 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

In cryptography and computer science, a **hash tree** or **Merkle tree** is a tree in which every "leaf" (node) is labelled with the cryptographic hash of a data block, and every node that is not a leaf (called a *branch*, *inner node*, or *inode*) is labelled with the cryptographic hash of the labels of its child nodes. A hash tree allows efficient and secure verification of the contents of a large data structure. A hash tree is a generalization of a hash list and a hash chain.

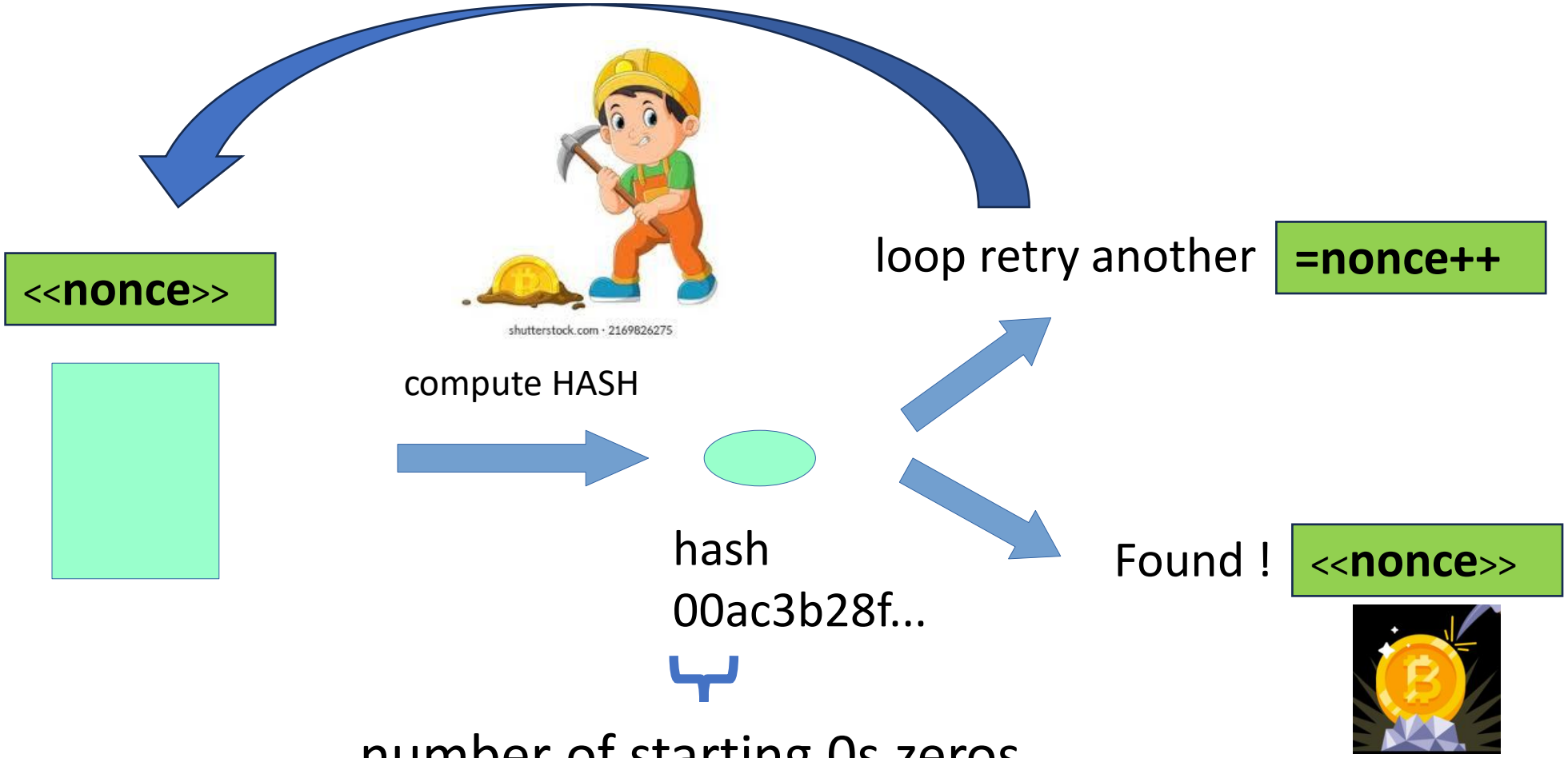


An example of a binary hash tree. Hashes 0-0 and 0-1 are the hash values of data blocks L1 and L2, respectively, and hash 0 is the hash of the concatenation of hashes 0-0 and 0-1.

Demonstrating that a leaf node is a part of a given binary hash tree requires computing a number of hashes proportional to the logarithm of the number of leaf nodes in the tree.[1] Conversely, in a hash list, the number is

Appending Blocks (commit transaction log) = Mining new Blocks Hashes

Mining Nonce - Difficulty Level

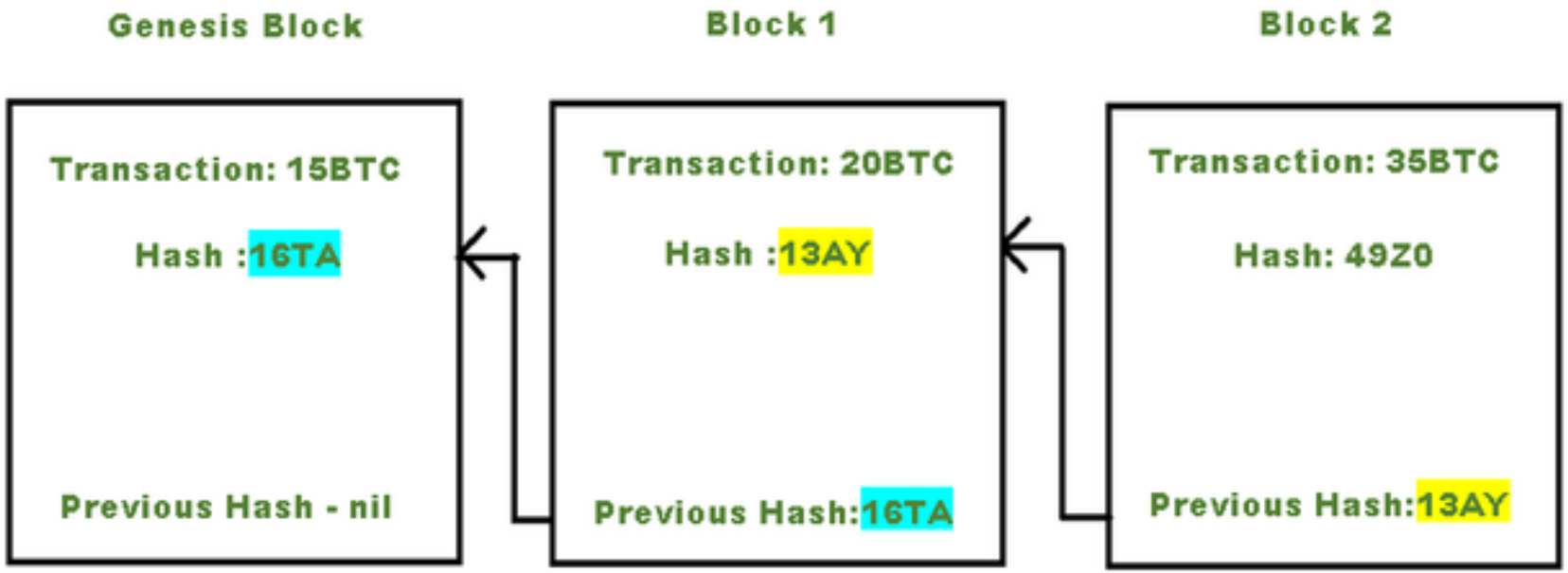


Proof Of Work

Find me a nounce giving 6 zeros in the hash for "xyz"

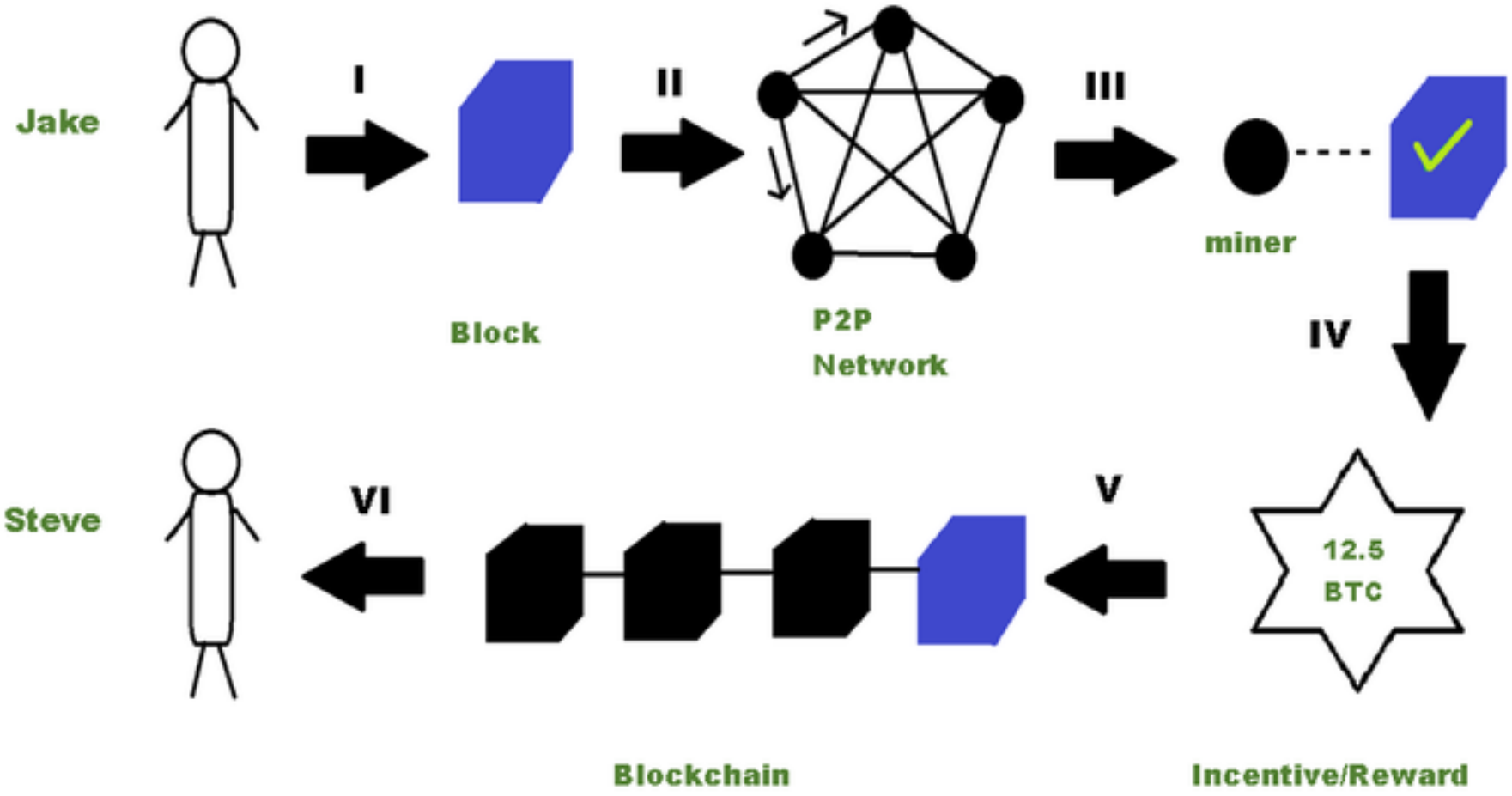
it will prove you did computation work
(proof you spend time and energy, money)

BlockChain =
Chaining Blocks with previous Hash



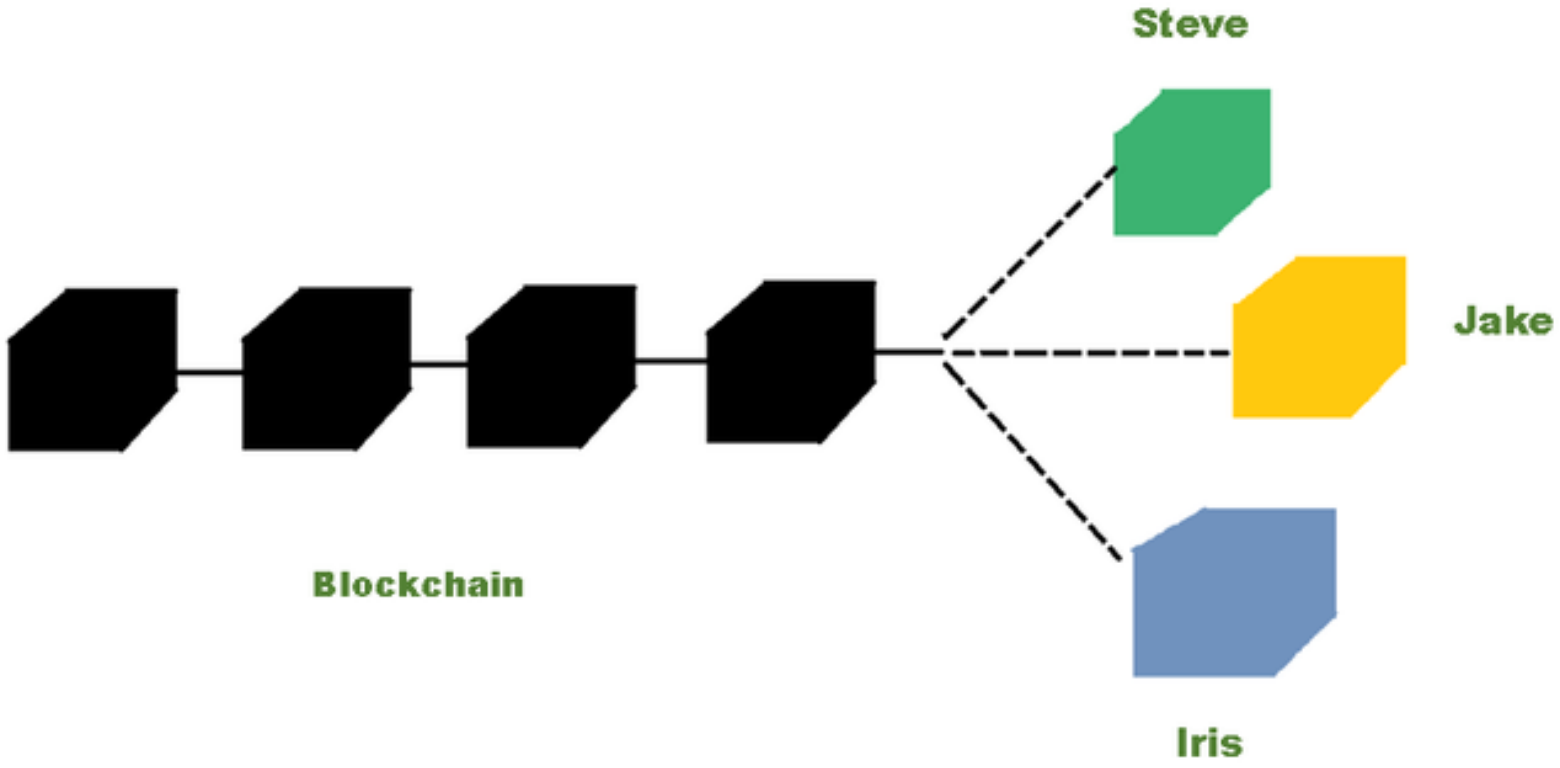
credits: https://www.geeksforgeeks.org/blockchain-resolving-conflicts/

Adding a block at the end of a chain



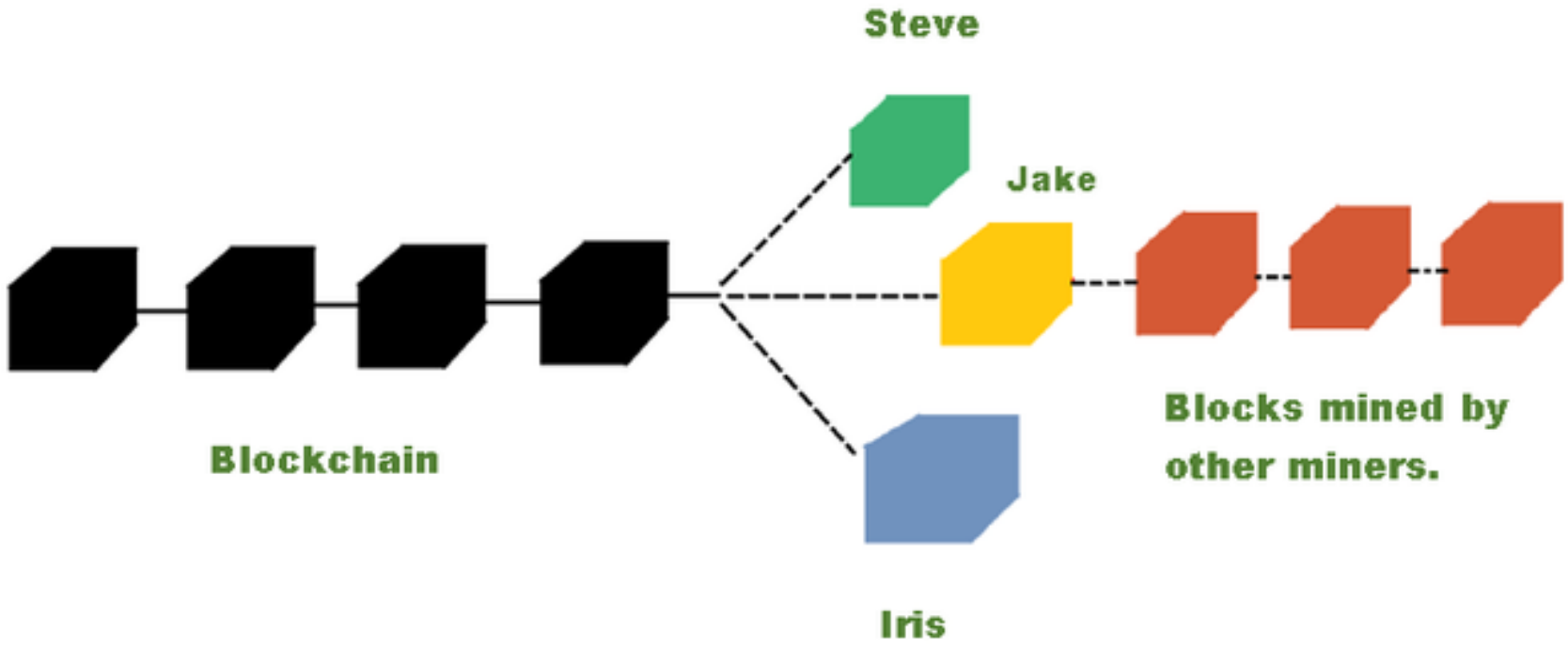
credits: https://www.geeksforgeeks.org/blockchain-resolving-conflicts/

Conflict Appending Blocks (Tree .. not List Chain)



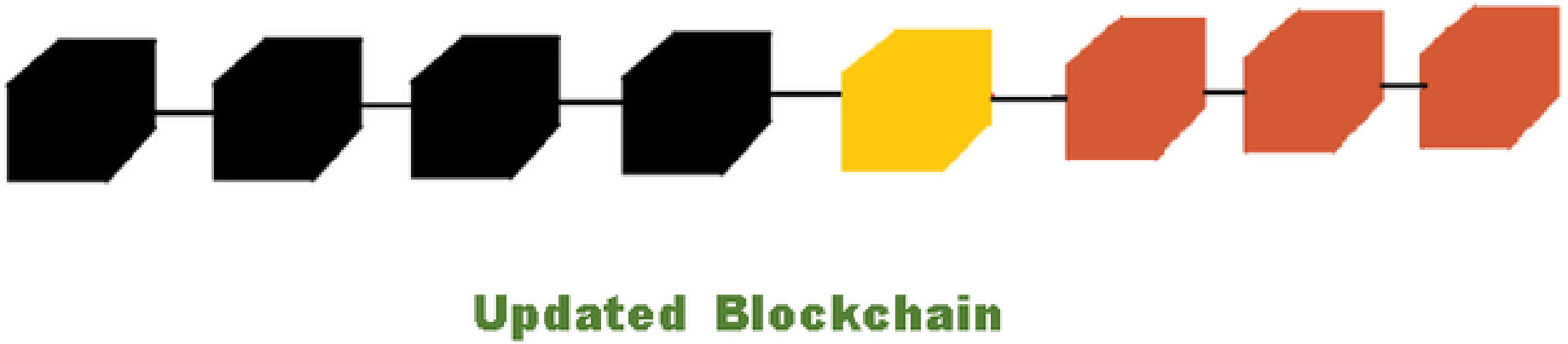
credits: https://www.geeksforgeeks.org/blockchain-resolving-conflicts/

Longest Chain Concensius



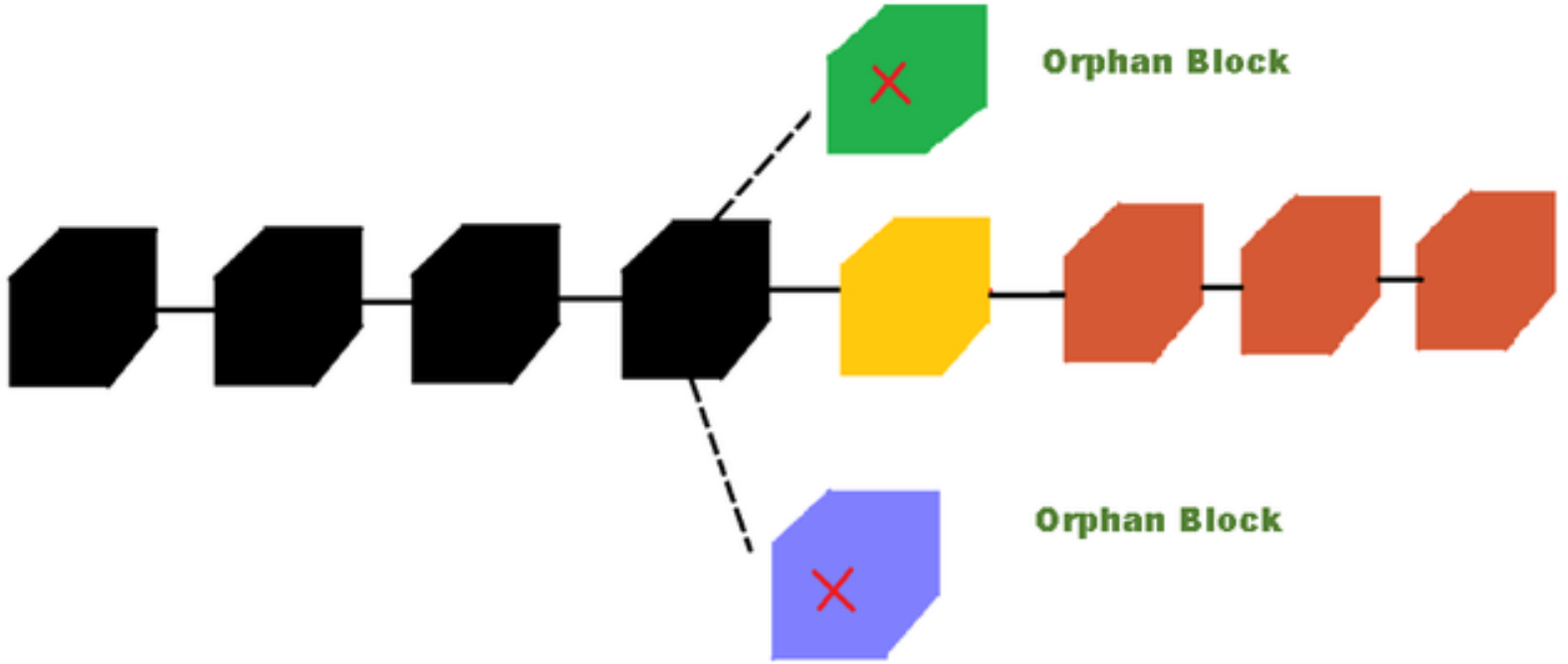
credits: https://www.geeksforgeeks.org/blockchain-resolving-conflicts/

Conscencius : retain only the longest chain(s) eliminate smaller old chains



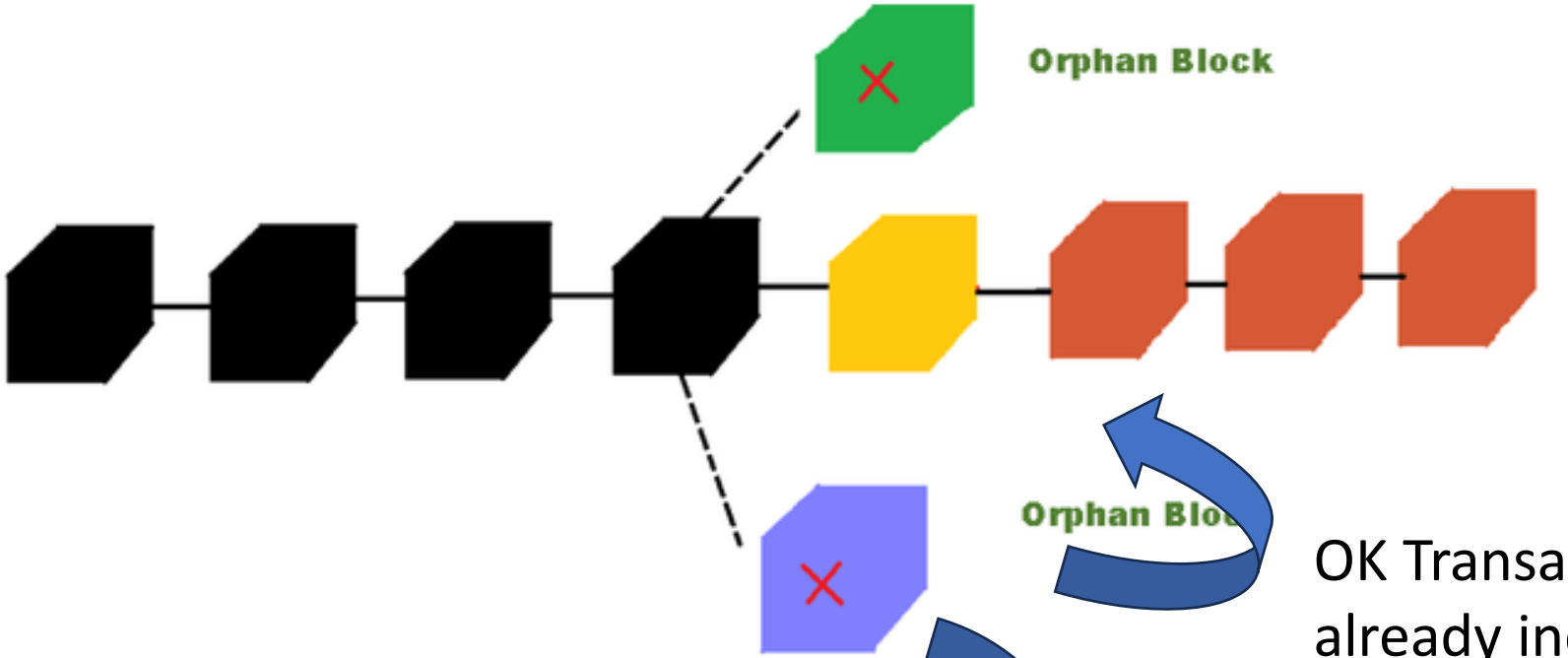
credits: https://www.geeksforgeeks.org/blockchain-resolving-conflicts/

Orphan Blocks ... Waste Mining for nothing



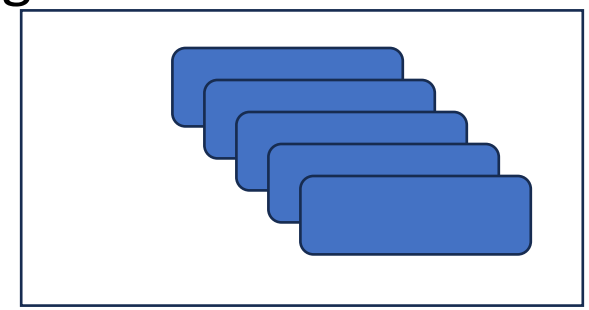
credits: https://www.geeksforgeeks.org/blockchain-resolving-conflicts/

Put back valid Transactions in Pool



OK Transaction
already included in block

Waiting Transactions Pool



Remaining Transactions
put back in Pool

Your Transaction is added in Block? OK ?

to pass a transaction payment, you must:

1/ propose your RAW transaction, sign it with your Private Key
(connect to network, add in Pool of Transactions)

2/ your transaction should include a Fee reward to Miners

3/ wait for miners

4/ & wait MORE .... you must wait >= 2-3 blocks chain
(because 1 block is NOT enough for "commit" garantee !)

CONS of Proof Of Work

SLOW

Eventually Consistent after long time

Energy inneficient

High Cost of Transactions

Risk of 51% Attack

BlockChain weaks [1/4] : **Slowness**

Proof Of Work (ex: BitCoin)

can NOT handle
High Volumes of Transactions
High Frequency Order (like in modern Banks / Trading Places)

So NOT suitable for Modern e-Trading Economy

SLOW ...

difficulty level adjusted (~2weeks) so that 1 block ~10 minutes

must wait several blocks (longest chain)

1 block (=1MB) contains few transactions

BitCoin transaction rate as of 2024

between 3 to 7 transactions per second
(world-wide)

example comparison modern e-trading Bank
> 1000 per seconds, per place

BlockChain weaks [2/4] : **Energy Inneficient**

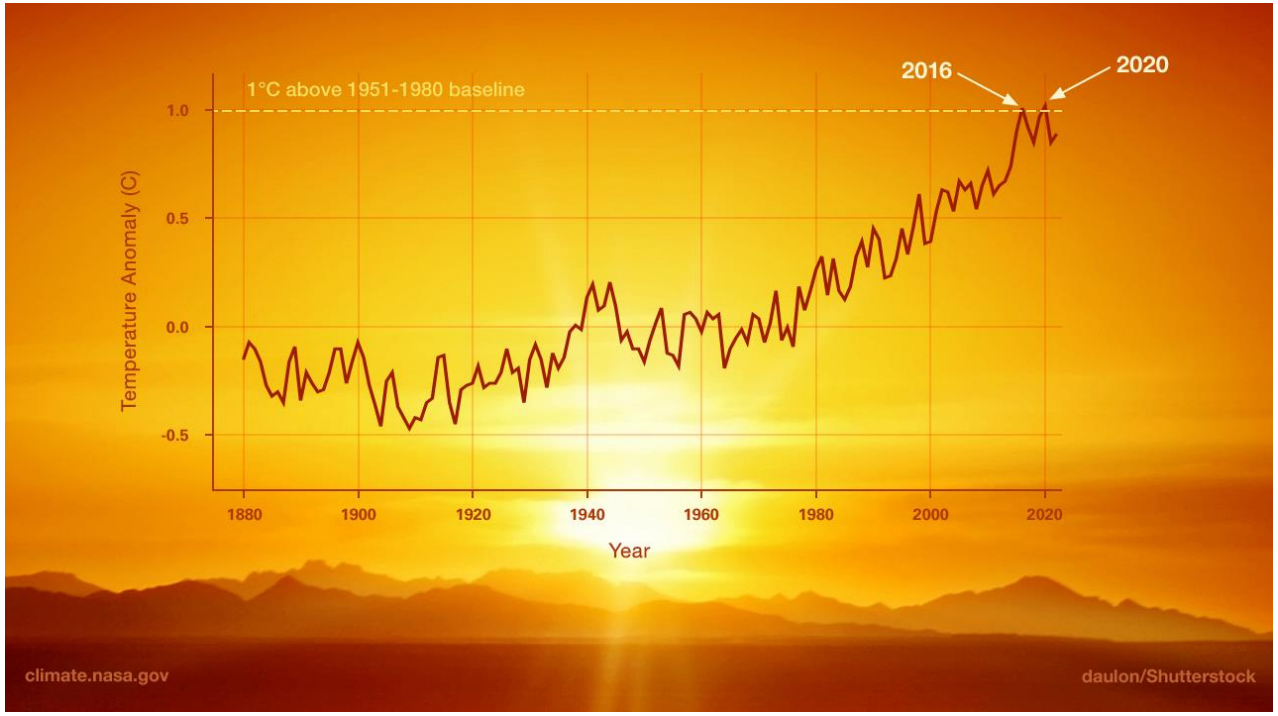
as of 2024, BitCoin consumes ~0.4% of world-wide electricity (~120 Tera Watt Hour)

similar to a country like Greece

https://www.eia.gov/todayinenergy/detail.php?id=61364

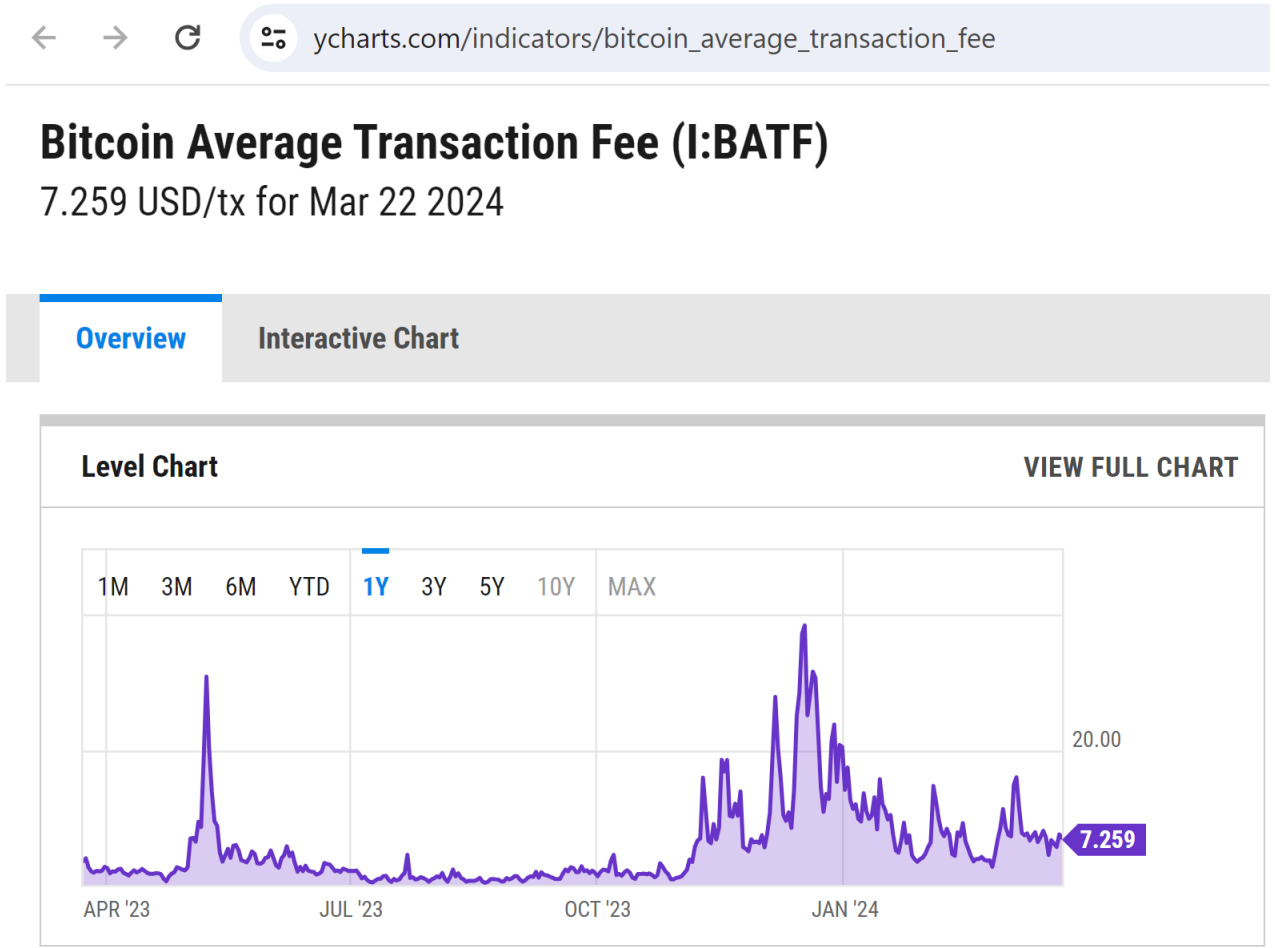
Mining is CPU (Energy Consuming)

Absurd in a Climate Changing World



https://science.nasa.gov/climate-change/

BlockChain weaks [3/4] : high transaction costs (cause: slow + small block + high energy + fees)



BlockChain weaks [4/4] : 51% Attack

What is a 51% Attack?

A 51% attack is a probable infringement on a blockchain network, where a single entity or organization possesses the majority of computational power and uses this power for network disruption. In such a scenario, they can

– Intentionally exclude or modify the ordering of transactions.
– Reverse transactions to cause a double-spending problem.
– Prevent some or all transactions from being confirmed
– Prevent some or all other miners from mining for mining monopoly
– Reverse transactions from other users
– Prevent transactions from being created and broadcasted to the network.
– Change the block's reward,
– Counterfeit crypto coins
– Steal coins

Dedicated Hardware

Can (Did) It Occur ?

https://en.wikipedia.org/wiki/Bitcoin

Scalability and decentralization challenges

Nakamoto limited the block size to one megabyte.[87] The limited block size and frequency can lead to delayed processing of transactions, increased fees and a Bitcoin scalability problem.[88] The Lightning Network, second-layer routing network, is a potential scaling solution.[8]:ch. 8

Research shows a trend towards centralization in bitcoin as miners join pools for stable income.[24]:215,219–222[89]:3 If a single miner or pool controls more than 50% of the hashing power, it would allow them to censor transactions and double-spend coins.[58] In 2014, mining pool Ghash.io reached 51% mining power, causing safety concerns, but later voluntarily capped its power at 39.99% for the benefit of the whole network.[90] A few entities also dominate other parts of the ecosystem such as the client software, online wallets, and simplified payment verification (SPV) clients.[58]

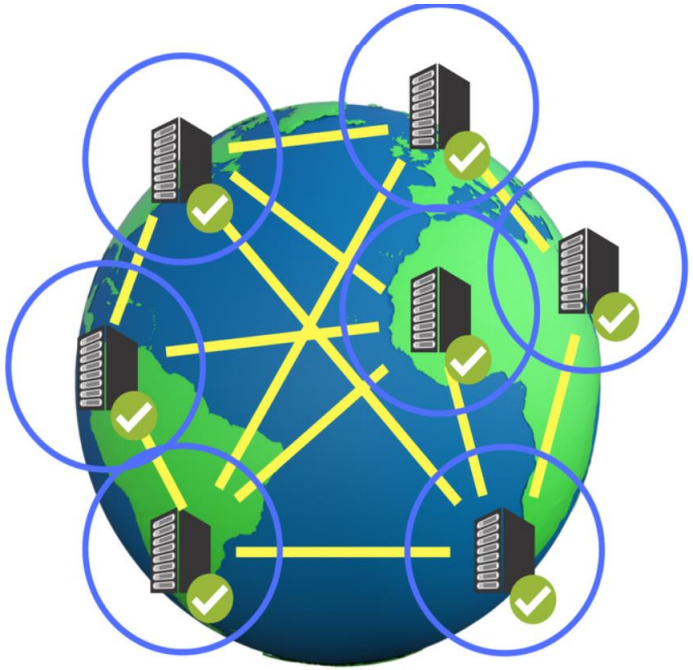
Mining Pool ?


In the context of cryptocurrency mining, a **mining pool** is the pooling of resources by miners, who share their processing power over a network, to split the reward equally, according to the amount of work they contributed to the probability of finding a block. A "share" is awarded to members
d towards centralization in bitcoin as miners join pools

Public -> Private (Permissioned) Block Chains

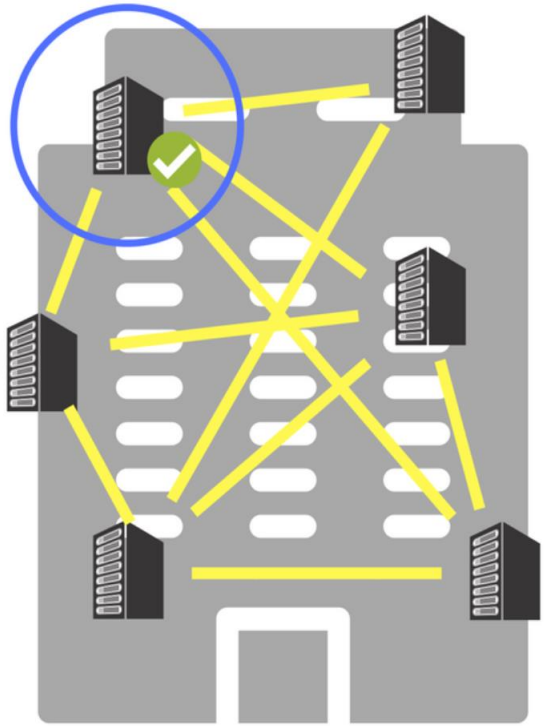
Public Blockchain

All Are Validators



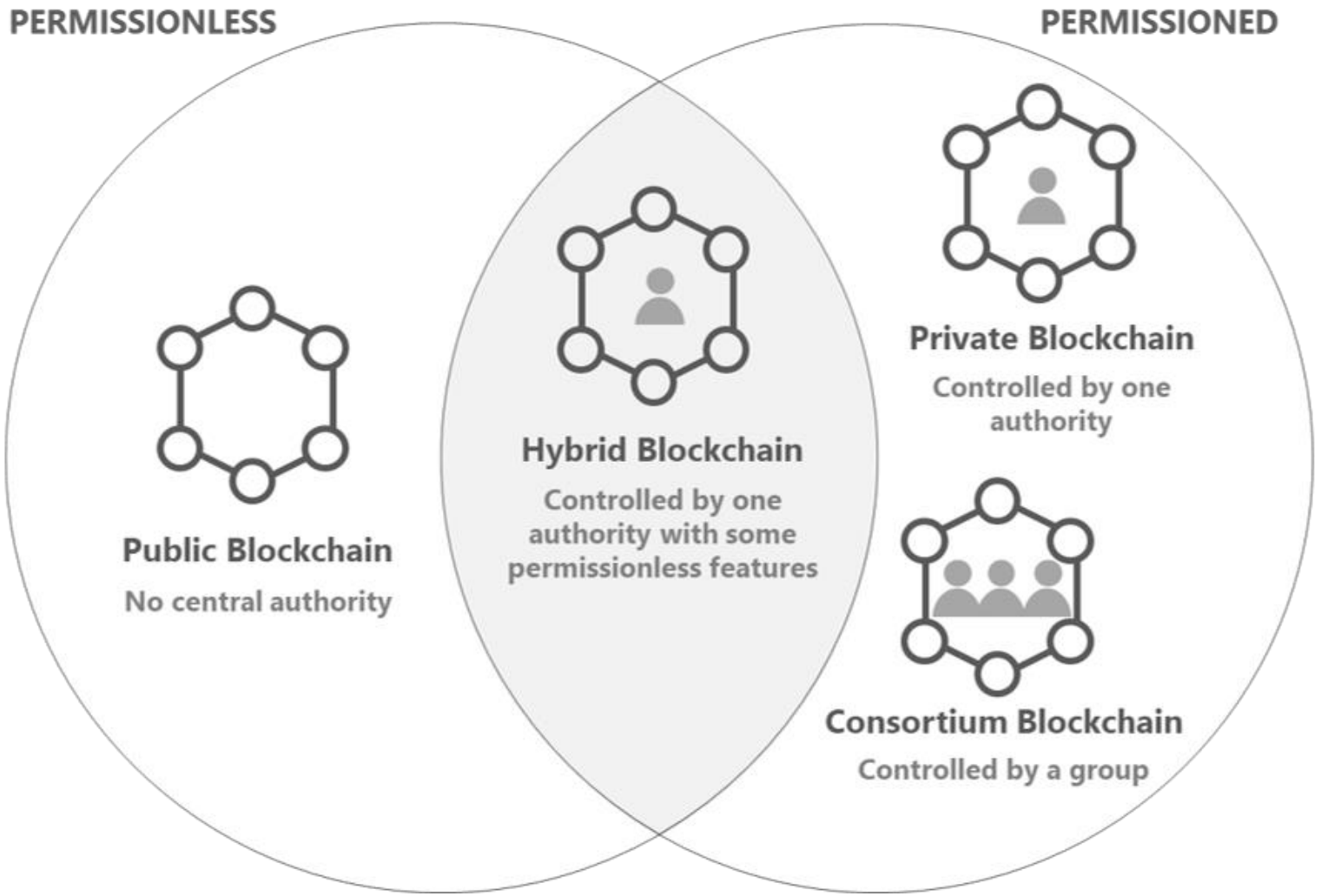
Private Blockchain

One Validator

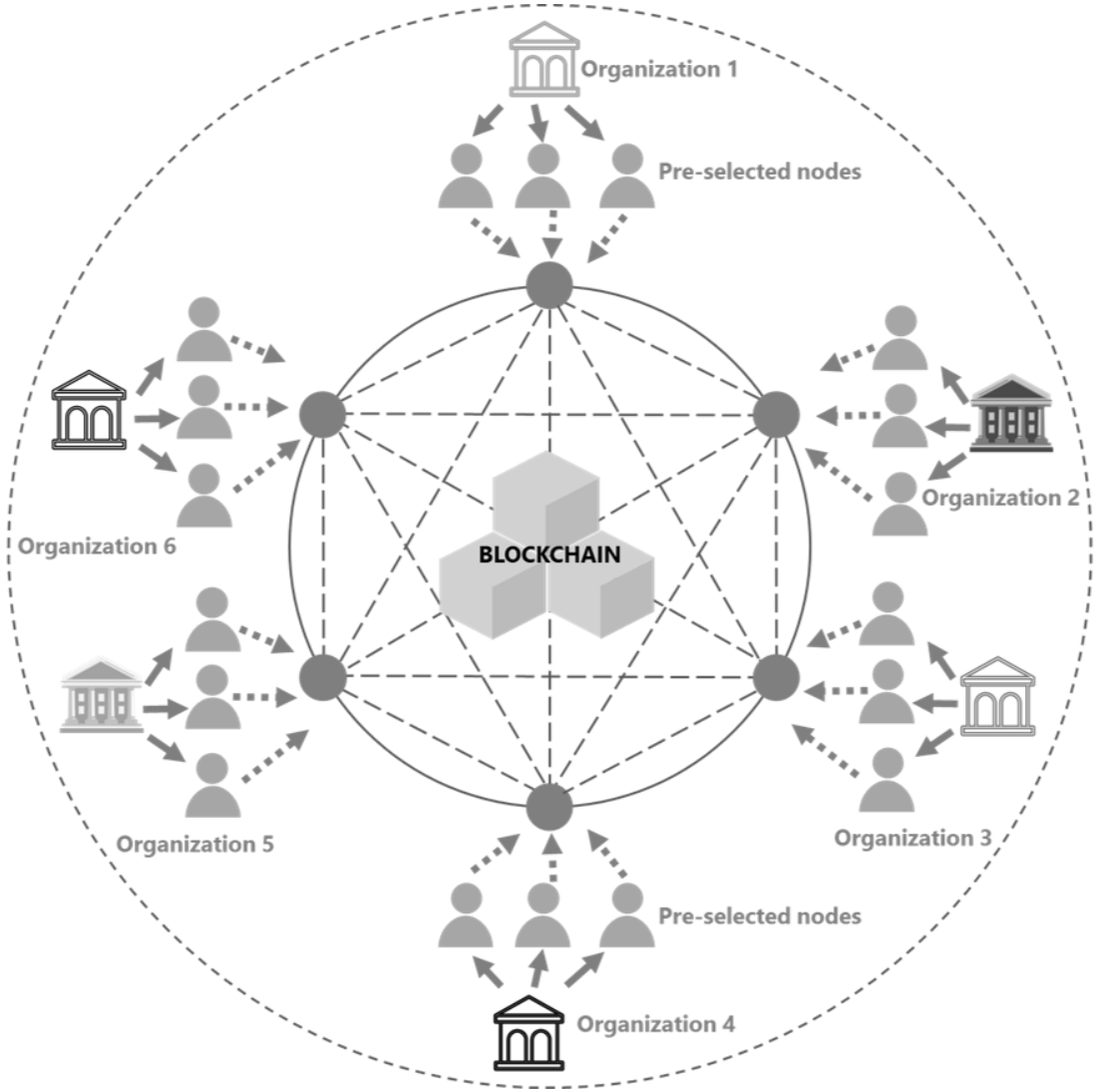


credits: https://blockchainlion.com/public-vs-private-blockchain-and-consortium/

Hybrid BlockChain



Consortium (Federation) BlockChain



https://medium.com/techskill-brew/types-of-blockchain-part-4-blockchain-basics-c0b2e40c1780

Proof Of Stake
Ethereum switch (2015)

PoW -> PoS (Proof Of Stake)

Proof of stake

23 languages

Article Talk Read View source View history Tools

From Wikipedia, the free encyclopedia

Proof-of-stake (**PoS**) protocols are a class of consensus mechanisms for blockchains that work by selecting validators in proportion to their quantity of holdings in the associated cryptocurrency. This is done to avoid the computational cost of proof-of-work (POW) schemes. The first functioning use of PoS for cryptocurrency was Peercoin in 2012, although the scheme, on the surface, still resembled a POW.[1]

PoS Variants : Chain-based, Nominated, ...

Variants

Chain-based PoS

This is essentially a modification of the PoW scheme, where the competition is based not on applying brute force to solving the identical puzzle in the smallest amount of time, but instead on varying the difficulty of the puzzle depending on the stake of the participant; the puzzle is solved if on a tick of the clock (|| is concatenation):

$$Hash(ProposedNewBlock||ClockTime) < target * StakeValue$$

The smaller amount of calculations required for solving the puzzle for high-value stakeholders helps to avoid excessive hardware.[13]

Nominated PoS (NPoS)

Also known as "committee-based", this scheme involves an election of a committee of validators using a verifiable random function with probabilities of being elected higher with higher stake. Validators then randomly take turns producing blocks. NPoS is utilized by Ouroboros Praos and BABE.[14]

BFT-based PoS

Main article: Byzantine fault tolerance

The outline of the BFT PoS "epoch" (adding a block to the chain) is as follows:[15]

1. A "proposer" with a "proposed block" is randomly selected by adding it to the temporary pool used to select just one consensual block;
2. The other participants, validators, obtain the pool, validate, and vote for one;
3. The BFT consensus is used to finalize the most-voted block.

The scheme works as long as no more than a third of validators are dishonest. BFT schemes are used in Tendermint and Casper FFG.[15]

Delegated proof of stake (DPoS)

Proof of stake delegated systems use a two-stage process: first,[16] the stakeholders elect a validation committee,[17] a.k.a. *witnesses*, by voting proportionally to their stakes, then the witnesses take turns in a round-robin fashion to propose new blocks that are then voted upon by the witnesses, usually in the BFT-like fashion. Since there are fewer validators in the DPoS than in many other PoS schemes, the consensus can be established faster. The scheme is used in many chains, including EOS, Lisk, Tron.[16]

Liquid proof of stake (LPoS)

In the liquid PoS anyone with a stake can declare themselves a validator, but for the small holders is makes sense to delegate their voting rights instead to larger players in exchange for some benefits (like periodic payouts). A market is established where the validators compete on the fees, reputation, and other factors. Token holders are free to switch their support to another validator at any time. LPoS is used in Tezos.[18]

Proof of Work vs Stake

Proof of Stake vs Proof of Work: Pros & Cons

	Proof of Stake	Proof of Work
Participants	They are called Validators as they primarily validate the block to add it to the blockchain by staking their cryptos. They are randomly chosen for the right to propose a block.	They are called Miners because they have to use huge resources to solve a puzzle and achieve the right to create and broadcast a block.
Resources Required	Validators should possess Coins or tokens to put as a stake.	The miner should be able to invest in high resources to participate.
Cost	Low Cost in terms of infrastructure. But miners have to invest in the base coin of the blockchain.	High cost of infrastructure required and recurring cost of energy resources.
Pros	– Low Cost of equipment – Energy efficient. – Higher Speed – High scalability – Discourages Centralization and 51% attack	– High cost of equipment to discourage attacks, thus, higher security. – Rewards efficient resources utilization such as energy, infrastructure and technology
Cons	– Comparatively new, not as proven in terms of security as proof of work. – Network Control can be purchased	– Enormous amount of energy required – High cost of participation – Prone to Centralization

BitCoin

BitCoin = first crypto money

Google

bitcoin

All News Images Videos Products More Tools

About 669,000,000 results (0.29 seconds)

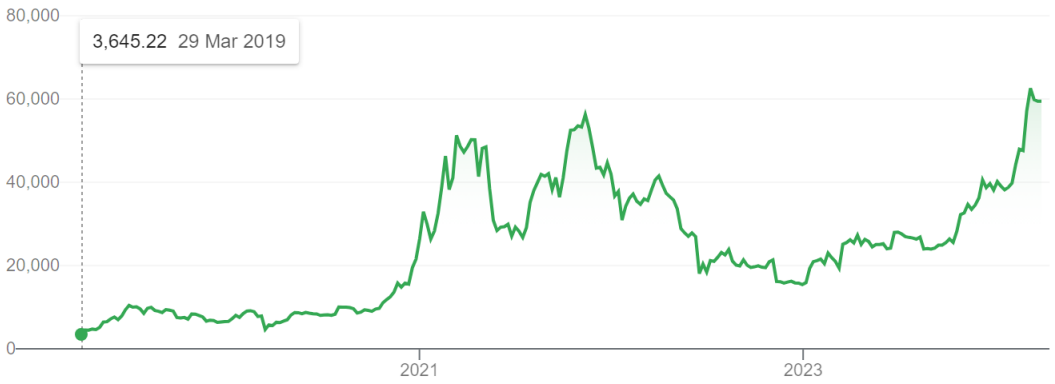
Market Summary > Bitcoin

59,451.60 EUR

+55,806.38 (1,530.95%) past 5 years

Mar 23, 13:09 UTC · Disclaimer

1D 5D 1M 6M YTD 1Y 5Y Max

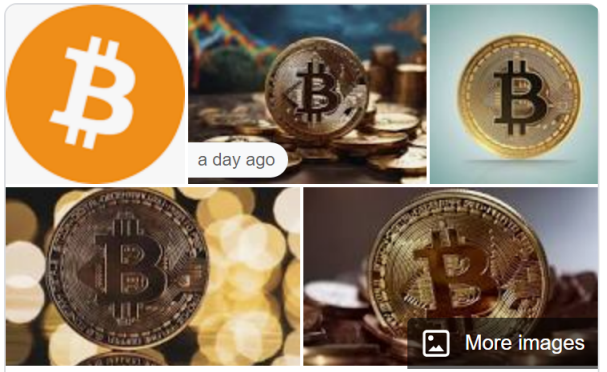


1 BTC 59451.60 EUR

Feedback

More about Bitcoin →

Follow



Bitcoin

Cryptocurrency

Bitcoin is the first decentralized cryptocurrency. Nodes in the peer-to-peer bitcoin network verify transactions through cryptography and record them in a public distributed ledger, called a blockchain, without central oversight. Wikipedia

Symbols: BTC, ฿, ₿

Block reward: ₿6.25 (as of 2023)

Code: BTC

Exchange rate: Floating

BitCoin Origin / (fictious) Author ?

Satoshi Nakamoto

49 languages

Article Talk | Read View source View history Tools

From Wikipedia, the free encyclopedia

Satoshi Nakamoto is the name used by the presumed pseudonymous[1][2][3][4] person or persons who developed Bitcoin, authored the Bitcoin white paper, and created and deployed Bitcoin's original reference implementation.[5] As part of the implementation, Nakamoto also devised the first blockchain database.[6] Nakamoto was active in the development of bitcoin until December 2010.[7]

There has been widespread speculation about Nakamoto's true identity, with various people posited as the person or persons behind the name. Though Nakamoto's name is Japanese, and he said in 2012 that he was a man living in Japan,[8] most of the speculation has involved software and cryptography experts in the United States or Europe.

Development of bitcoin

Nakamoto said that the work of writing Bitcoin's code began in the second quarter of 2007.[9] On 18 August 2008, he or a colleague registered the domain name bitcoin.org,[10] and created a web site at that address. On 31 October, Nakamoto published a white paper on the cryptography mailing list at metzdowd.com describing a digital cryptocurrency, titled "Bitcoin: A Peer-to-Peer Electronic Cash System".[11][12][13]

On 9 January 2009, Nakamoto released version 0.1 of the Bitcoin software on SourceForge and launched the network by defining the *genesis block of bitcoin* (block number 0), which had a reward of 50 bitcoins.[14][15][7][16] Embedded in the coinbase transaction of this block is the text: "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks",[17] citing a headline in the UK newspaper *The Times* published on that date.[18] This note has been interpreted as both a timestamp and a derisive comment on the alleged instability caused by fractional-reserve banking.[19]:18

Satoshi Nakamoto

A statue in Budapest dedicated to Satoshi Nakamoto

Known for	Inventing bitcoin, implementing the first blockchain
Scientific career	
Fields	Digital currencies, computer science, cryptography

Legal ... but controversial

legal in many countries

still controversial

used by too many illegal parties (for anonymous safety)

authorized as ETF since 2024

used as investment more than payment

economic speculative bubble?

Dev Reference

Block Chain

The following subsections briefly document core block details.

Block Headers

Block headers are serialized in the 80-byte format described below and then hashed as part of Bitcoin's proof-of-work algorithm, making the serialized header format part of the consensus rules.

Bytes	Name	Data Type	Description
4	version	int32_t	The block version number indicates which set of block validation rules to follow. See the list of block versions below.
32	previous block	char[32]	A SHA256(SHA256()) hash in internal byte order of the previous block's

Block Header

Block Headers

Block headers are serialized in the 80-byte format described below and then hashed as part of Bitcoin's proof-of-work algorithm, making the serialized header format part of the consensus rules.

Bytes	Name	Data Type	Description
4	version	int32_t	The block version number indicates which set of block validation rules to follow. See the list of block versions below.
32	previous block header hash	char[32]	A SHA256(SHA256()) hash in internal byte order of the previous block's header. This ensures no previous block can be changed without also changing this block's header.
32	merkle root hash	char[32]	A SHA256(SHA256()) hash in internal byte order. The merkle root is derived from the hashes of all transactions included in this block, ensuring that none of those transactions can be modified without modifying the header. See the merkle trees section below.
4	time	uint32_t	The block time is a Unix epoch time when the miner started hashing the header (according to the miner). Must be strictly greater than the median time of the previous 11 blocks. Full nodes will not accept blocks with headers more than two hours in the future according to their clock.
4	nBits	uint32_t	An encoded version of the target threshold this block's header hash must be less than or equal to. See the nBits format described below.
4	nonce	uint32_t	An arbitrary number miners change to modify the header hash in order to produce a hash less than or equal to the target threshold. If all 32-bit values are tested, the time can be updated or the coinbase transaction can be changed and the merkle root updated.

https://developer.bitcoin.org/reference/block_chain.html

Example Block Header

```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... [Unix time][unix epoch time]: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```

Block = Header + Transactions (+ Reward)

Serialized Blocks ¶

Under current consensus rules, a block is not valid unless its serialized size is less than or equal to 1 MB. All fields described below are counted towards the serialized size.

Bytes	Name	Data Type	Description
80	block header	block_header	The block header in the format described in the block header section.
Varies	txn_count	compactSize uint	The total number of transactions in this block, including the coinbase transaction.
Varies	txns	raw transaction	Every transaction in this block, one after another, in raw transaction format. Transactions must appear in the data stream in the same order their TXIDs appeared in the first row of the merkle tree. See the merkle tree section for details.

The first transaction in a block must be a coinbase transaction which should collect and spend any transaction fees paid by transactions included in this block.

All blocks with a block height less than 6,930,000 are entitled to receive a block subsidy of newly created bitcoin value, which also should be spent in the coinbase transaction. (The block subsidy started at 50 bitcoins and is being halved every 210,000 blocks—approximately once every four years. As of November 2017, it's 12.5 bitcoins.)

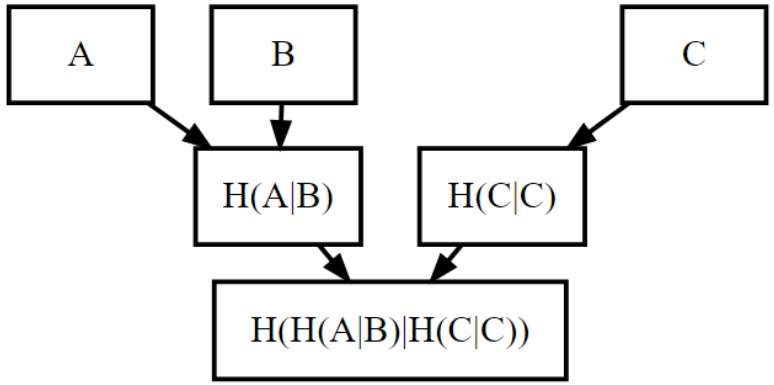
Together, the transaction fees and block subsidy are called the block reward. A coinbase transaction is invalid if it tries to spend more value than is available from the block reward.

Transactions in Merkle Tree using double SHA256(SHA256(...))

Row 1: Transaction hashes (TXIDs)
(A is coinbase; C can spend output from B)

Row 2: Hashes of paired TXIDs

Merkle root



Example Merkle Tree Construction [Hash function H() = SHA256(SHA256())]

Example Merkle Tree Construction

Example Merkle Tree

```
01000000 ............................ Block version: 1
82bb869cf3a793432a66e826e05a6fc3
7469f8efb7421dc88067010000000000 ... Hash of previous block's header
7f16c5962e8bd963659c793ce370d95f
093bc7e367117b3c30c1f8fdd0d97287 ... Merkle root
76381b4d ............................ Time: 1293629558
4c86041b ............................ nBits: 0x04864c * 256**(0x1b-3)
554b8529 ............................ Nonce

07000000 ............................ Transaction count: 7
04 .................................. Hash count: 4

3612262624047ee87660be1a707519a4
43b1c1ce3d248cbfc6c15870f6c5daa2 ... Hash #1
019f5b01d4195ecbc9398fbf3c3b1fa9
bb3183301d7a1fb3bd174fcfa40a2b65 ... Hash #2
41ed70551dd7e841883ab8f0b16bf041
76b7d1480e4f0af9f3d4c3595768d068 ... Hash #3
20d2a7bc994987302e5b1ac80fc425fe
25f8b63169ea78e68fbaaefa59379bbf ... Hash #4

01 .................................. Flag bytes: 1
1d .................................. Flags: 1 0 1 1 1 0 0 0
```

https://developer.bitcoin.org/reference/p2p_networking.html

Transactions: Inputs, Outputs

A raw transaction has the following top-level format:

Bytes	Name	Data Type	Description
4	version	int32_t	Transaction version number (note, this is signed); currently version 1 or 2. Programs creating transactions using newer consensus rules may use higher version numbers. Version 2 means that BIP 68 applies.
Varies	tx_in count	compactSize uint	Number of inputs in this transaction.
Varies	tx_in	txIn	Transaction inputs. See description of txIn below.
Varies	tx_out count	compactSize uint	Number of outputs in this transaction.
Varies	tx_out	txOut	Transaction outputs. See description of txOut below.
4	lock_time	uint32_t	A time (Unix epoch time) or block number. See the locktime parsing rules.

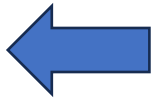
Tx Input

TxIn: A Transaction Input (Non-Coinbase)

Each non-coinbase input spends an outpoint from a previous transaction. (Coinbase inputs are described separately after the example section below.)

Bytes	Name	Data Type	Description
36	previous_output	outpoint	The previous outpoint being spent. See description of outpoint below.
Varies	script bytes	compactSize uint	The number of bytes in the signature script. Maximum is 10,000 bytes.
Varies	signature script	char[]	A script-language script which satisfies the conditions placed in the outpoint's pubkey script. Should only contain data pushes; see the signature script modification warning.
4	sequence	uint32_t	Sequence number. Default for Bitcoin Core and almost all other programs is 0xffffffff.

How manyBitCoins
you had before ?



Your BitCoin "Address"
i.e. your **publick Key hashed**

TxOutput

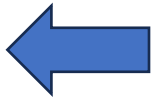
Bytes	Name	Type	Description
32	hash	char[32]	The TXID of the transaction holding the output to spend. The TXID is a hash provided here in internal byte order.
4	index	uint32_t	The output index number of the specific output to spend from the transaction. The first output is 0x00000000.

TxOut: A Transaction Output

Each output spends a certain number of satoshis, placing them under control of anyone who can satisfy the provided pubkey script.

Bytes	Name	Data Type	Description
8	value	int64_t	Number of satoshis to spend. May be zero; the sum of all outputs may not exceed the sum of satoshis previously spent to the outpoints provided in the input section. (Exception: coinbase transactions spend the block subsidy and collected transaction fees.)
1+	pk_script bytes	compactSize uint	Number of bytes in the pubkey script. Maximum is 10,000 bytes.
Varies	pk_script	char[]	Defines the conditions which must be satisfied to spend this output.

How manyBitCoins
to spend



Destination BitCoin "Address"
i.e. **publick Key hashed**
... signed by your Private Key

2 Types of Transactions: Pay to {PubKey | Script}

Types of Transaction

Bitcoin currently creates two different scriptSig/scriptPubKey pairs. These are described below.

It is possible to design more complex types of transactions, and link them together into cryptographically enforced agreements. These are known as Contracts.

Pay-to-PubkeyHash

```
scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>
```

A Bitcoin address is only a hash, so the sender can't provide a full public key in scriptPubKey. When redeeming coins that have been sent to a Bitcoin address, the recipient provides both the signature and the public key. The script verifies that the provided public key does hash to the hash in scriptPubKey, and then it also checks the signature against the public key.

Pay-to-Script-Hash

```
scriptPubKey: OP_HASH160 <scriptHash> OP_EQUAL
scriptSig: ..signatures... <serialized script>
```

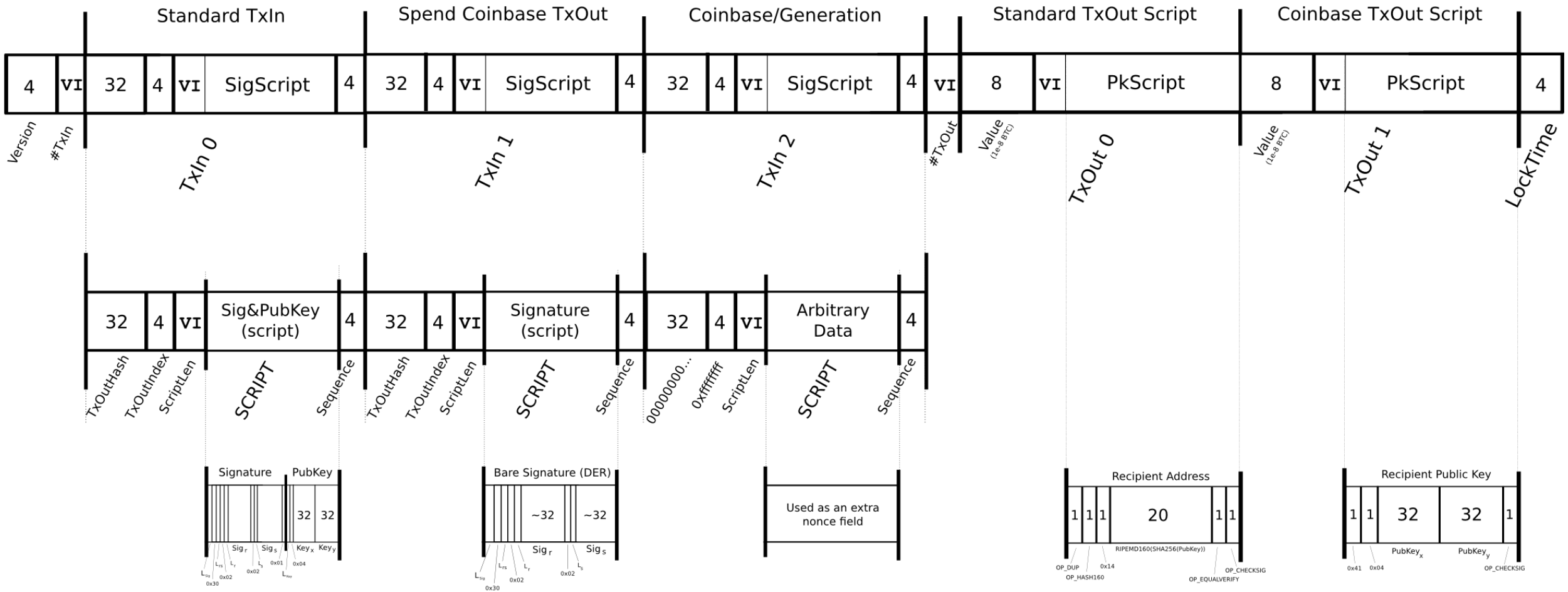
```
m-of-n multi-signature transaction:
scriptSig: 0 <sig1> ... <script>
script: OP_m <pubKey1> ... OP_n OP_CHECKMULTISIG
```

P2SH addresses were created with the motivation of moving "the responsibility for supplying the conditions to redeem a transaction from the sender of the funds to the redeemer. They allow the sender to fund an arbitrary transaction, no matter how complicated, using a 20-byte hash"1. Pay-to-Pubkey-hash addresses are similarly a 20-byte hash of the public key.

Pay-to-script-hash provides a means for complicated transactions, unlike the Pay-to-pubkey-hash, which has a specific definition for scriptPubKey, and scriptSig. The specification places no limitations on the script, and hence absolutely any contract can be funded using these addresses.

https://en.bitcoin.it/wiki/Transaction

Transaction



Scripts and DER encoding both use big-endian values, all other serializations use little-endian

BitCoin "Address"
= Public Key Hash

Generate a Private/Pub
Key Pair

HASH your PubKey
it is your bitcoin "Address"

save your Private Key
in your "WALLET"
(and do several encrypted backups)

NO recovery mecanism if you loose your private Key

Lost Private Key ?

Losing a private key means losing access to the bitcoins, with no other proof of ownership accepted by the protocol.[24] For instance, in 2013, a user lost ₿7,500, valued at US$7.5 million, by accidentally discarding a hard drive with the private key.[63] It is estimated that around 20% of all bitcoins are lost.[64] The private key must also be kept secret as its exposure, such as through a data breach, can lead to theft of the associated bitcoins.[8]:ch. 10[65] As of December 2017, approximately ₿980,000 had been stolen from cryptocurrency exchanges.[66]

https://en.wikipedia.org/wiki/Bitcoin

pseudonymous ?

Public Key are "anonymous"
may need to be declared in some countries

someone may re-create several new public keys

Difficult to trace identities from history of transactions

so "pseudonymous" (but not "anonymous")

Ethereum SmartContract

Ethereum

Google ethereum

All News Images Videos Books More Tools

About 212,000,000 results (0.39 seconds)

Market Summary > Ether

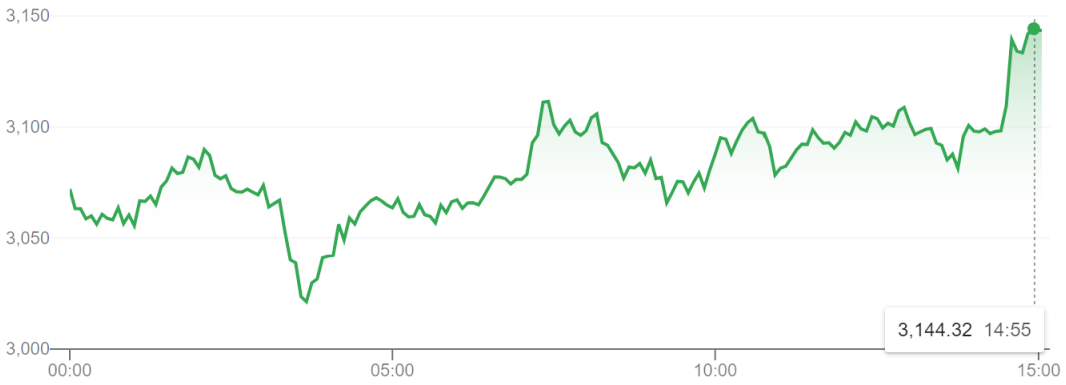
3,143.47 EUR

+71.38 (2.32%) ↑ today

Mar 23, 15:03 UTC · Disclaimer

Follow

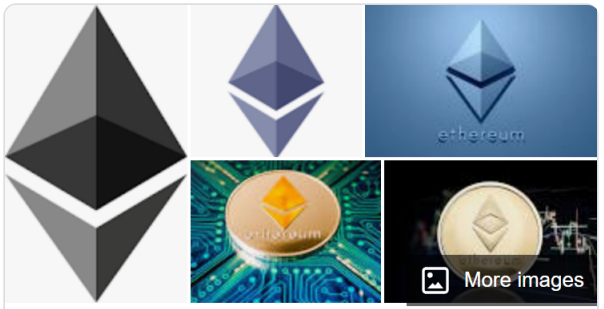
1D 5D 1M 6M YTD 1Y 5Y Max



1 ETH = 3143.47 EUR

Feedback

More about Ether →



Ethereum

Software

Ethereum is a decentralized blockchain with smart contract functionality. Ether is the native cryptocurrency of the platform. Among cryptocurrencies, ether is second only to bitcoin in market capitalization. It is open-source software. Ethereum was conceived in 2013 by programmer Vitalik Buterin. Wikipedia

Developer: Vitalik Buterin, Gavin Wood

Platforms: x86-64, ARM architecture family

Compatible Operating Systems: Linux, Microsoft Windows, macOS, POSIX

Initial release date: July 30, 2015

Ethereum ... compared to BitCoin

Ethereum

68 languages

Article Talk | Read View source View history Tools

From Wikipedia, the free encyclopedia

Ethereum is a decentralized blockchain with smart contract functionality. **Ether** (Abbreviation: **ETH**;[a] sign: Ξ) is the native cryptocurrency of the platform. Among cryptocurrencies, ether is second only to bitcoin in market capitalization.[2][3] It is open-source software.

Ethereum was conceived in 2013 by programmer Vitalik Buterin.[4] Additional founders of Ethereum included Gavin Wood, Charles Hoskinson, Anthony Di Iorio, and Joseph Lubin.[5] In 2014, development work began and was crowdfunded, and the network went live on 30 July 2015.[6] Ethereum allows anyone to deploy permanent and immutable decentralized applications onto it, with which users can interact.[7] Decentralized finance (DeFi) applications provide financial instruments that do not directly rely on financial intermediaries like brokerages, exchanges, or banks. This facilitates borrowing against cryptocurrency holdings or lending them out for interest.[8][9] Ethereum also allows users to create and exchange non-fungible tokens (NFTs), which are tokens that can be tied to unique digital assets, such as images. Additionally, many other cryptocurrencies utilize the ERC-20 token standard on top of the Ethereum blockchain and have utilized the platform for initial coin offerings.

On 15 September 2022, Ethereum transitioned its consensus mechanism from proof-of-work (PoW) to proof-of-stake (PoS) in an upgrade process known as "the Merge". This has cut Ethereum's energy usage by 99%.[10]

Ethereum

Logo

Original author(s)	Vitalik Buterin Gavin Wood
Initial release	30 July 2015; 8 years ago
Stable release	1.12.2 / 13 August 2023; 7 months ago

Ether (ETH) <--> BitCoin
Ethereum <--> BitCoin network

Ethereum Transactions rate per day
~1 Million per Day (>= Bitcoin 7 per sec)

Smart Contract

Smart contract

33 languages

Article Talk | Read View source View history Tools

From Wikipedia, the free encyclopedia

This article is about contractual transactions on a decentralized platform. For smart legal contracts, see Smart legal contract.

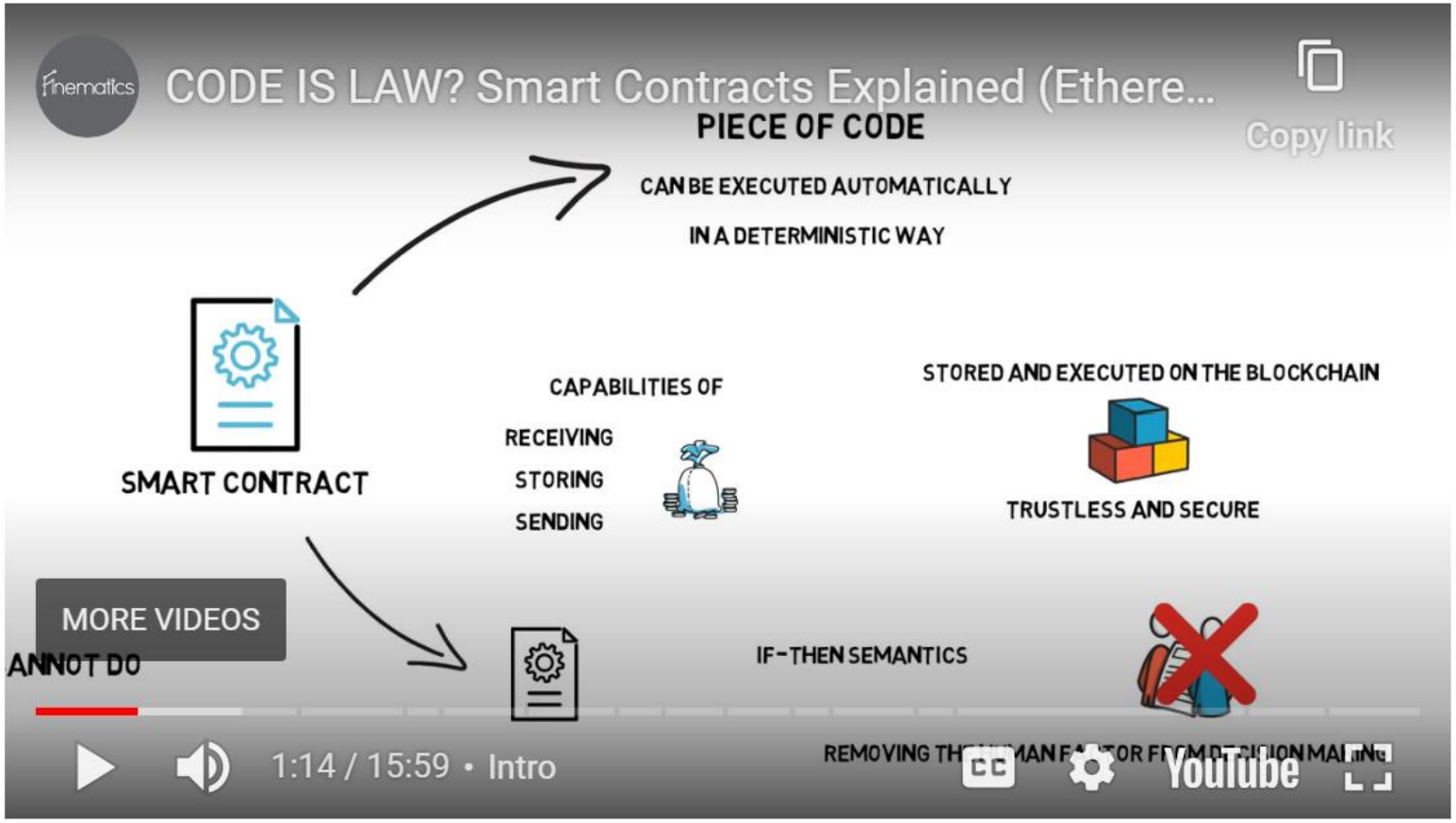
A **smart contract** is a computer program or a transaction protocol that is intended to automatically execute, control or document events and actions according to the terms of a contract or an agreement.[1][2][3][4] The objectives of smart contracts are the reduction of need for trusted intermediators, arbitration costs, and fraud losses, as well as the reduction of malicious and accidental exceptions.[5][2] Smart contracts are commonly associated with cryptocurrencies, and the smart contracts introduced by Ethereum are generally considered a fundamental building block for decentralized finance (DeFi) and NFT applications.[6][7]

Vending machines are mentioned as the oldest piece of technology equivalent to smart contract implementation.[3] The original Ethereum white paper by Vitalik Buterin in 2014[8] describes the Bitcoin protocol as a weak version of the smart contract concept as originally defined by Nick Szabo, and proposed a stronger version based on the Solidity language, which is Turing complete. Since Bitcoin,[*clarification needed*] various cryptocurrencies have supported programming languages which allow for more advanced smart contracts between untrusted parties.[9]

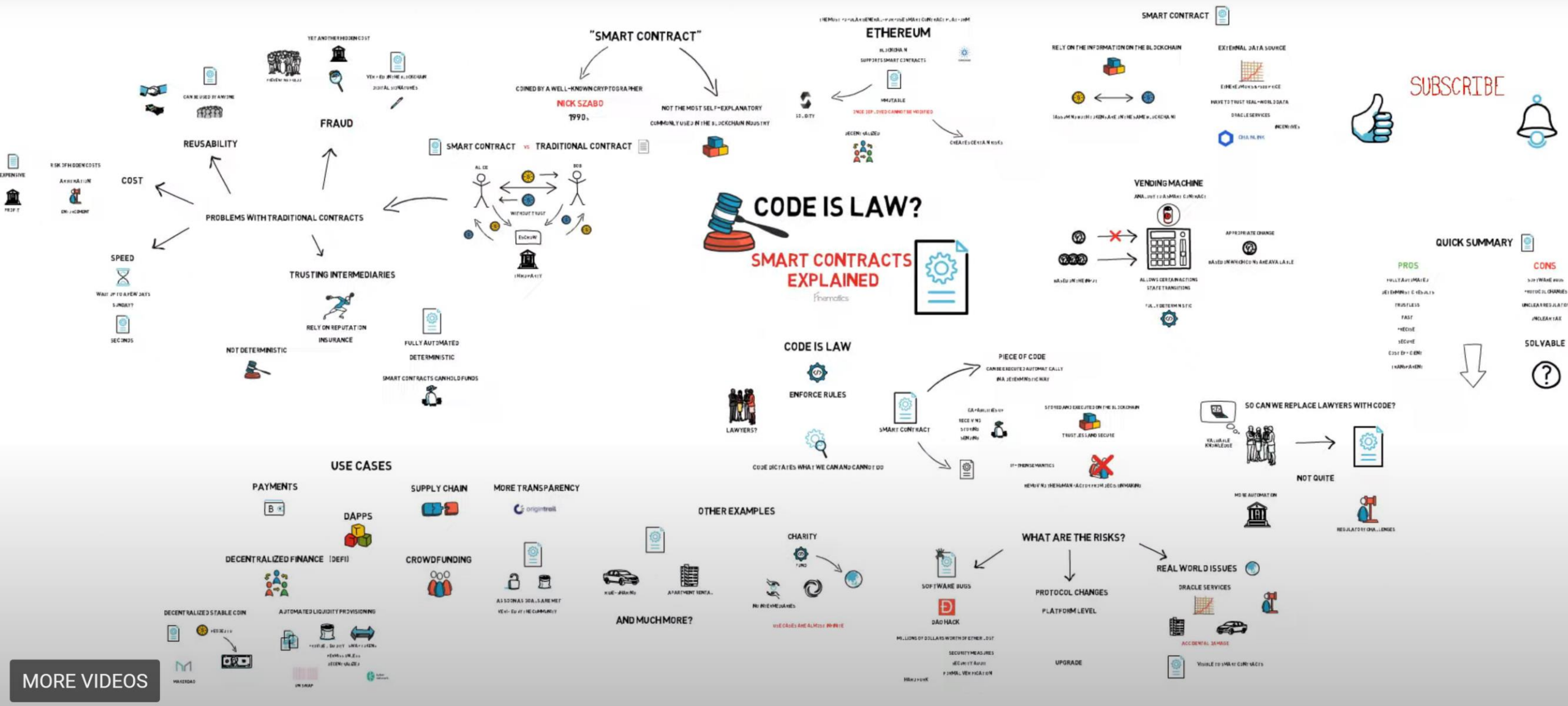
https://en.wikipedia.org/wiki/Smart_contract

Smart Contract
... Not Only "Payment" Transactions / No Human

https://ethereum.org/en/smart-contracts/



https://youtu.be/pWGLtjG-F5c


"SMART CONTRACT"
COINED BY A WELL-KNOWN CRYPTOGRAPHER
NICK SZABO
1990s
NOT THE MOST SELF-EXPLANATORY
COMMONLY USED IN THE BLOCKCHAIN INDUSTRY
ETHEREUM
SMART CONTRACT
RELY ON THE INFORMATION ON THE BLOCKCHAIN
EXTERNAL DATA SOURCE
SUBSCRIBE
FRAUD
REUSABILITY
COST
SPEED
TRUSTING INTERMEDIARIES
RELY ON REPUTATION
INSURANCE
NOT DETERMINISTIC
FULLY AUTOMATED
DETERMINISTIC
SMART CONTRACTS CAN HOLD FUNDS
PROBLEMS WITH TRADITIONAL CONTRACTS
SMART CONTRACT vs TRADITIONAL CONTRACT
ALICE
BOB
WITHOUT A TRUST
EXCHANGE
THIRD PARTY
CODE IS LAW?
SMART CONTRACTS EXPLAINED
finematics
VENDING MACHINE
ALLOWS CERTAIN ACTIONS
STATE TRANSITIONS
QUICK SUMMARY
PROS
CONS
SOLVABLE
CODE IS LAW
ENFORCE RULES
LAWYERS?
CODE DICTATES WHAT WE CAN AND CANNOT DO
SMART CONTRACT
PIECE OF CODE
STORED AND EXECUTED ON THE BLOCKCHAIN
TRUSTLESS AND SECURE
SO CAN WE REPLACE LAWYERS WITH CODE?
NOT QUITE
USE CASES
PAYMENTS
DAPPS
SUPPLY CHAIN
MORE TRANSPARENCY
DECENTRALIZED FINANCE (DEFI)
CROWDFUNDING
DECENTRALIZED STABLE COIN
AUTOMATED LIQUIDITY PROVISIONING
OTHER EXAMPLES
CHARITY
AND MUCH MORE?
WHAT ARE THE RISKS?
SOFTWARE BUGS
DAO HACK
PROTOCOL CHANGES
PLATFORM LEVEL
UPGRADE
REAL WORLD ISSUES
ORACLE SERVICES

Solid & EVM (Ethereum Virtual Machine)

Solid = a Turing Complete programming langage

run on a EVM

consume "GAS"

Ethereum GAS

GAS = Fee you must pay in ETH unit to ask Minters

to add your transaction
to run each operation code of your SmartContract program

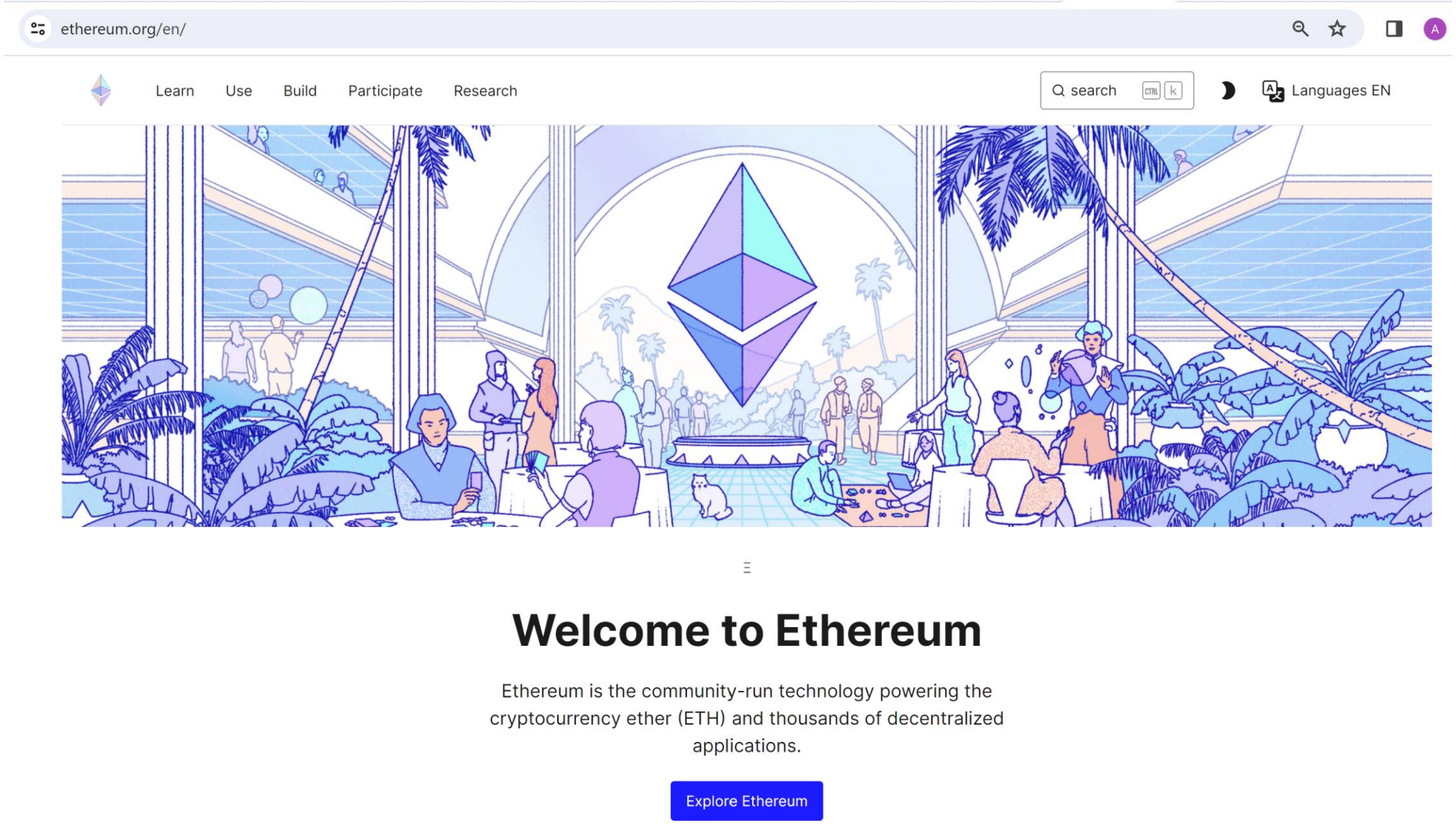
Solidity Code example

Example of a Solidity program:[19][20]

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.4;

contract Coin {
    // The keyword "public" makes variables
    // accessible from other contracts
    address public minter;
    mapping(address => uint) public balances;

    // Events allow clients to react to specific
    // contract changes you declare
    event Sent(address from, address to, uint amount);

    // Constructor code is only run when the contract
    // is created
    constructor() {
        minter = msg.sender;
    }

    // Sends an amount of newly created coins to an address
    // Can only be called by the contract creator
    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        balances[receiver] += amount;
    }

    // Errors allow you to provide information about
    // why an operation failed. They are returned
    // to the caller of the function.
    error InsufficientBalance(uint requested, uint available);

    // Sends an amount of existing coins
    // from any caller to an address
    function send(address receiver, uint amount) public {
        if (amount > balances[msg.sender])
            revert InsufficientBalance({
                requested: amount,
                available: balances[msg.sender]
            });

        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

https://en.wikipedia.org/wiki/Solidity

https://ethereum.org/



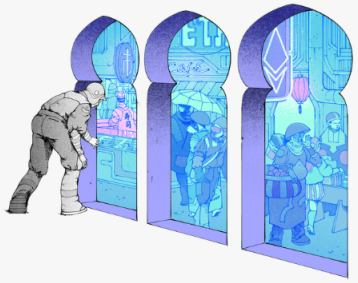
https://ethereum.org/en/learn/

What is Ethereum?

Cryptocurrencies, such as bitcoin, enable anyone to transfer money globally. Ethereum does too, but it can also run code that enables people to create apps and organizations. It's both resilient and flexible: any computer program can run on Ethereum. Learn more and find out how to get started:

What is Ethereum?

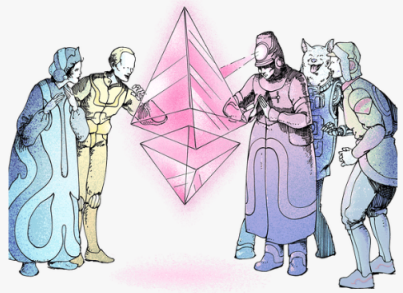
If you are new, start here to learn why Ethereum matters.



What is Ethereum?

What is ETH?

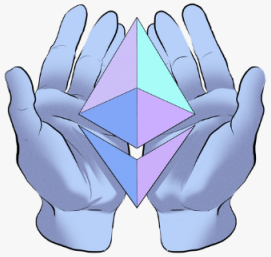
Ether (ETH) is the currency powering the Ethereum network and apps.



What is ETH?

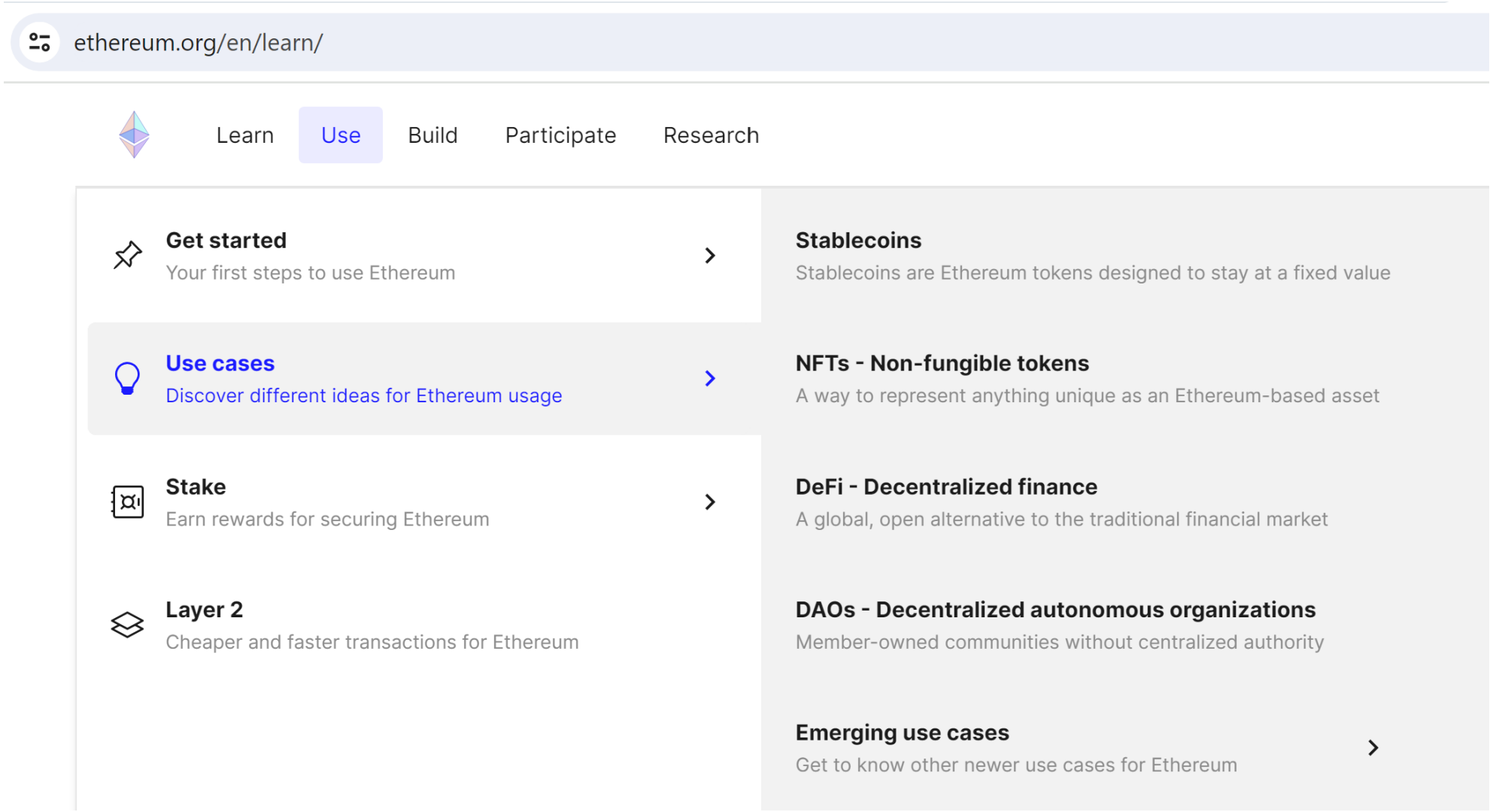
What is Web3?

Web3 is a model for the internet valuing ownership of your assets and identity.



What is Web3?

Ethereum Use Cases



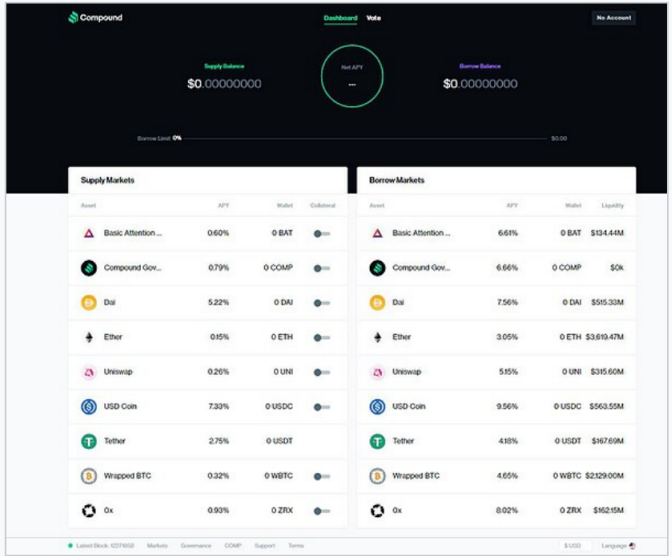
Example of DeFi (Decentralized Finance)

Decentralized finance

Main article: Decentralized finance

Decentralized finance (DeFi) offers traditional financial instruments in a decentralized architecture, outside of companies' and governments' control, such as money market funds which let users earn interest.[70] DeFi applications are typically accessed through a Web3-enabled browser extension or application, such as MetaMask, which allows users to directly interact with the Ethereum blockchain through a website.[71] Many of these DApps can connect and work together to create complex financial services.[72]

Examples of DeFi platforms include MakerDAO and Compound.[73] Uniswap, a decentralized exchange for tokens on Ethereum grew from US$20 million in liquidity to US$2.9 billion in 2020.[74] As of October 2020, over US$11 billion was invested in various DeFi protocols.[75] Additionally, through a process called "wrapping", certain DeFi protocols allow synthetic versions of various assets (such as bitcoin, gold, and oil) to be tradeable on Ethereum and also compatible with all of Ethereum's major wallets and applications.[75]



The web interface to Compound Finance's decentralized application where users can lend and borrow cryptocurrencies for interest

DAI … stable coin ~1 USD

Dai (cryptocurrency)

7 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

(Redirected from MakerDAO)

Dai (or **DAI**, formerly **Sai** or **SAI**) is a stablecoin on the Ethereum blockchain whose value is kept as close to one United States dollar as possible through a system of decentralized participants incentivized by smart contracts to perform maintenance and governance functions.[1] Dai is maintained and regulated by MakerDAO, a decentralized autonomous organization (DAO) composed of the owners of its governance token, MKR, who may propose and vote on changes to certain parameters in its smart contracts in order to ensure the stability of Dai.[2][3]

Together, Dai and MakerDAO are considered the first example of decentralized finance to receive significant adoption.[4]

Overview [edit]

Dai is created and destroyed through an overcollateralized loan and repayment process facilitated by MakerDAO's smart contracts in the form of a decentralized application. Users who deposit one of the accepted collateral types (such as Ether) into a contract are able to mint new DAI, as a loan, against the value of their collateral. The USD value of the collateral at any given time divided by the amount of DAI borrowed is the loan's "collateralization ratio"; this is calculated using the USD price of a unit of the collateral asset as reported regularly to a contract by a set of decentralized oracles. Each loan type has a

Dai

Denominations	
Plural	Dai
Code	DAI
Previous names	Sai
Precision	10^{-18}
Development	
White paper	makerdao.com/en/whitepaper/
Initial release	December 18, 2017 (6 years ago)

Electronic Vote <-?-> BlockChain

Voting ...

Limited to basic question "YES / NO"

Occurs once every N (=5) years

Small participation rates

Stop the world campaign

Survey / Fake News inferences

Huge Cost
(example: 40% for paper in french election)

e-Voting

A worthy e-voting system must perform most of these tasks while complying with a set of standards established by regulatory bodies, and must also be capable to deal successfully with strong requirements associated with security, accuracy, integrity, swiftness, privacy, auditability, accessibility, cost-effectiveness, scalability and ecological sustainability.

e-Voting Mathematical Criteria

!= using an Electronic Voting Machine
!= Voting on Internet

individual votes must be secret ("cast" your vote = encrypt it)

individual can check their vote is included (unaltered) in the ballot

can not replace/stole a vote

validators (anyone) can do the voting ballots (count even when encrypted)

…

Homomorphic Encryption

Homomorphic encryption

17 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

Homomorphic encryption is a form of encryption that allows computations to be performed on encrypted data without first having to decrypt it. The resulting computations are left in an encrypted form which, when decrypted, result in an output that is identical to that produced had the operations been performed on the unencrypted data. Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted.

Homomorphic encryption eliminates the need for processing data in the clear, thereby preventing attacks that would enable a hacker to access that data while it is being processed, using privilege escalation.[1]

Homomorphic encryption

General	
Derived from	Various assumptions, including learning with errors, Ring learning with errors or even RSA (multiplicative) and others
Related to	Functional encryption

Example Homomorphic Encryption/Operations

Partially homomorphic cryptosystems [edit]

In the following examples, the notation $\mathcal{E}(x)$ is used to denote the encryption of the message x.

Unpadded RSA

If the RSA public key has modulus n and encryption exponent e, then the encryption of a message m is given by $\mathcal{E}(m) = m^e \mod n$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= m_1^e m_2^e \mod n \\ &= (m_1 m_2)^e \mod n \\ &= \mathcal{E}(m_1 \cdot m_2)\end{aligned}$$

ElGamal

In the ElGamal cryptosystem, in a cyclic group G of order q with generator g, if the public key is (G, q, g, h), where $h = g^x$, and x is the secret key, then the encryption of a message m is $\mathcal{E}(m) = (g^r, m \cdot h^r)$, for some random $r \in \{0, \ldots, q-1\}$. The homomorphic property is then

$$\begin{aligned}\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{r_1}, m_1 \cdot h^{r_1})(g^{r_2}, m_2 \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, (m_1 \cdot m_2)h^{r_1+r_2}) \\ &= \mathcal{E}(m_1 \cdot m_2).\end{aligned}$$

e-Voting ?

At a glance

Scientific Foresight: What if …?

European Parliament

What if blockchain technology revolutionised voting?

Is blockchain the revolution in security and transparency that is needed to enable e-voting and, if so, what are the implications for the future of democracy?

©Adstcok

Despite the digitalisation of several important aspects of modern life, elections are still largely conducted offline, on paper. Since the turn of the century, e-voting has been considered a promising and (eventually) inevitable development, which could speed up, simplify and reduce the cost of elections, and might even lead to higher voter turnouts and the development of stronger democracies. E-voting could take many forms: using the internet or a dedicated, isolated network; requiring voters to attend a polling station or allowing unsupervised voting; using existing devices, such as mobile phones and laptops, or requiring specialist equipment. Now we have a further choice; to continue trusting central authorities to manage elections or to use blockchain technology to distribute an open voting record among citizens. Many experts agree that e-voting would require revolutionary developments in security systems. The debate is whether blockchain will represent a transformative or merely incremental development, and what its implications could be for the future of democracy.

How blockchain technology could be used for e-voting

https://www.europarl.europa.eu/RegData/etudes/ATAG/2016/581918/EPRS_ATA(2016)581918_EN.pdf

Privacy-Preserving Voting ?

You

give Privacy-Preserving Voting Protocols with blockchain

ChatGPT

Here are two privacy-preserving voting protocols that can be implemented using blockchain:

1. **Zero-Knowledge Proof (ZKP) Voting**:
   - **Setup Phase**:
     - A voting authority generates parameters for a zero-knowledge proof system.
     - Parameters are published to the blockchain.
   - **Voting Phase**:
     - Each voter creates a zero-knowledge proof attesting to the validity of their vote without revealing the vote itself.
     - Voters submit their proofs as transactions to the blockchain.
   - **Tallying Phase**:
     - After the voting phase ends, the voting authority collects the proofs from the blockchain.
     - The authority verifies the proofs and counts the valid votes without learning the actual votes.
     - The tally is published to the blockchain in an encrypted format.
   - **Verification Phase**:
     - Voters can verify that their vote was included in the tally by confirming the presence of their proof on the blockchain.
     - The authority can provide a method for independently verifying the correctness of the tally without revealing individual votes.
2. **Mixnet-based Voting**:
   - **Setup Phase**:
     - A set of mix servers is established, each with its public/private key pair.
     - Public keys of mix servers are published to the blockchain.
   - **Voting Phase**:
     - Each voter encrypts their vote using the public keys of the mix servers in a random order.
     - Encrypted votes are submitted as transactions to the blockchain.
   - **Mixing Phase**:
     - Mix servers receive encrypted votes and shuffle them to break the link between voters and their votes.
     - The shuffled votes are then published to the blockchain.
   - **Tallying Phase**:
     - After the mixing phase ends, the voting authority collects the shuffled votes from the blockchain.
     - The authority decrypts the votes, counts them, and computes the tally without knowing the identity of individual voters.
     - The tally is published to the blockchain.
   - **Verification Phase**:
     - Voters can verify that their encrypted vote was included in the tally by confirming the presence of their vote on the blockchain.
     - The authority can provide a method for independently verifying the correctness of the tally without revealing individual votes.

These protocols leverage the transparency and immutability of the blockchain while preserving the privacy of individual votes. Implementations would require careful consideration of cryptographic techniques, smart contract design, and network architecture to ensure security and

ZKP : Zero-Knowledge Proof ?

You

what is zero-knowledge proof for voting ?

ChatGPT

Zero-knowledge proof (ZKP) is a cryptographic technique that allows one party, called the prover, to prove to another party, called the verifier, that a statement is true without revealing any additional information beyond the validity of the statement itself. In the context of voting, zero-knowledge proofs can be used to prove that a vote is valid without revealing the actual vote itself, thereby preserving the privacy of the voter.

In a zero-knowledge proof for voting, the prover (the voter) aims to convince the verifier (the voting authority or anyone else verifying the vote) that they have cast a valid vote without revealing the content of the vote. Here's a simplified explanation of how this can work:

NIZKP : Non-Interactive Zero Kowledge Proof

You

what is nizkp ?

ChatGPT

"NIZKP" stands for "Non-Interactive Zero-Knowledge Proof." It refers to a cryptographic protocol where a prover can convince a verifier that a certain statement is true without engaging in an interactive process. In other words, the prover generates a proof that can be verified by the verifier without any back-and-forth communication between them.

NIZK proofs are particularly useful in scenarios where interactive communication between the prover and verifier is not feasible or desirable, such as in decentralized systems, distributed networks, or scenarios involving privacy-preserving protocols.

NIZK proofs have applications in various fields, including:

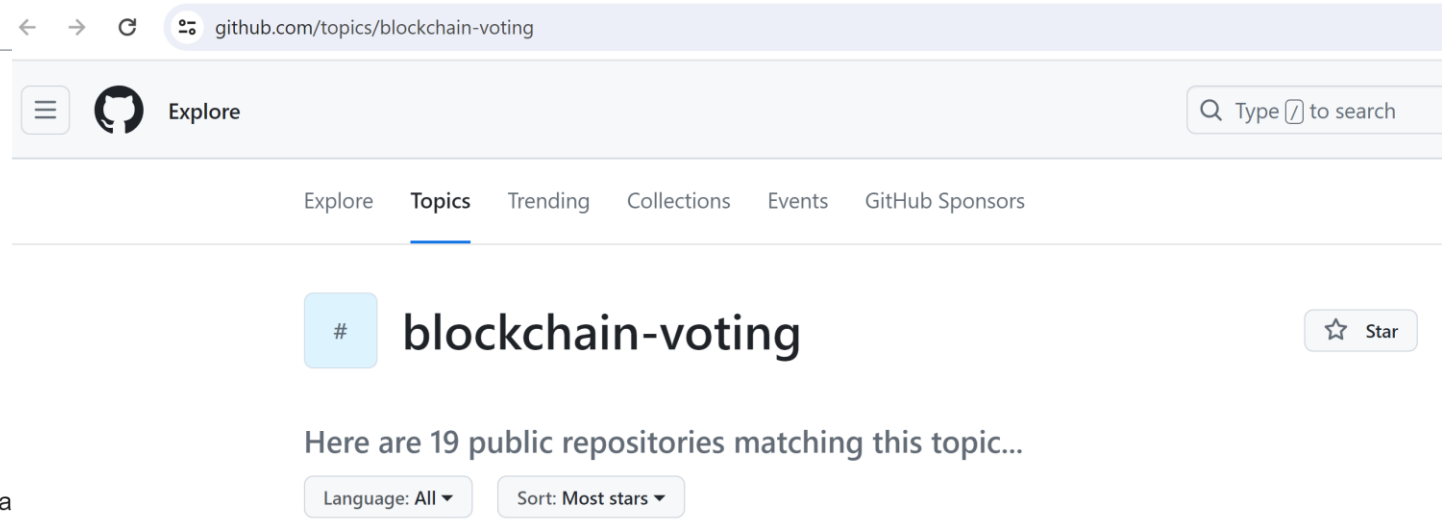
1. **Privacy-Preserving Cryptography**: NIZK proofs are used to demonstrate the validity of statements or transactions in a way that preserves privacy. For example, in anonymous credential systems or cryptographic voting protocols, NIZK proofs can be used to verify the authenticity of identities or votes without revealing sensitive information.
2. **Blockchain Technology**: In blockchain networks, NIZK proofs are used to provide cryptographic evidence of the correctness of transactions, smart contract executions, or other computations performed on the blockchain without revealing any additional information about the transactions

Auditable / BlockChain e-Voting Examples

Examples [edit]

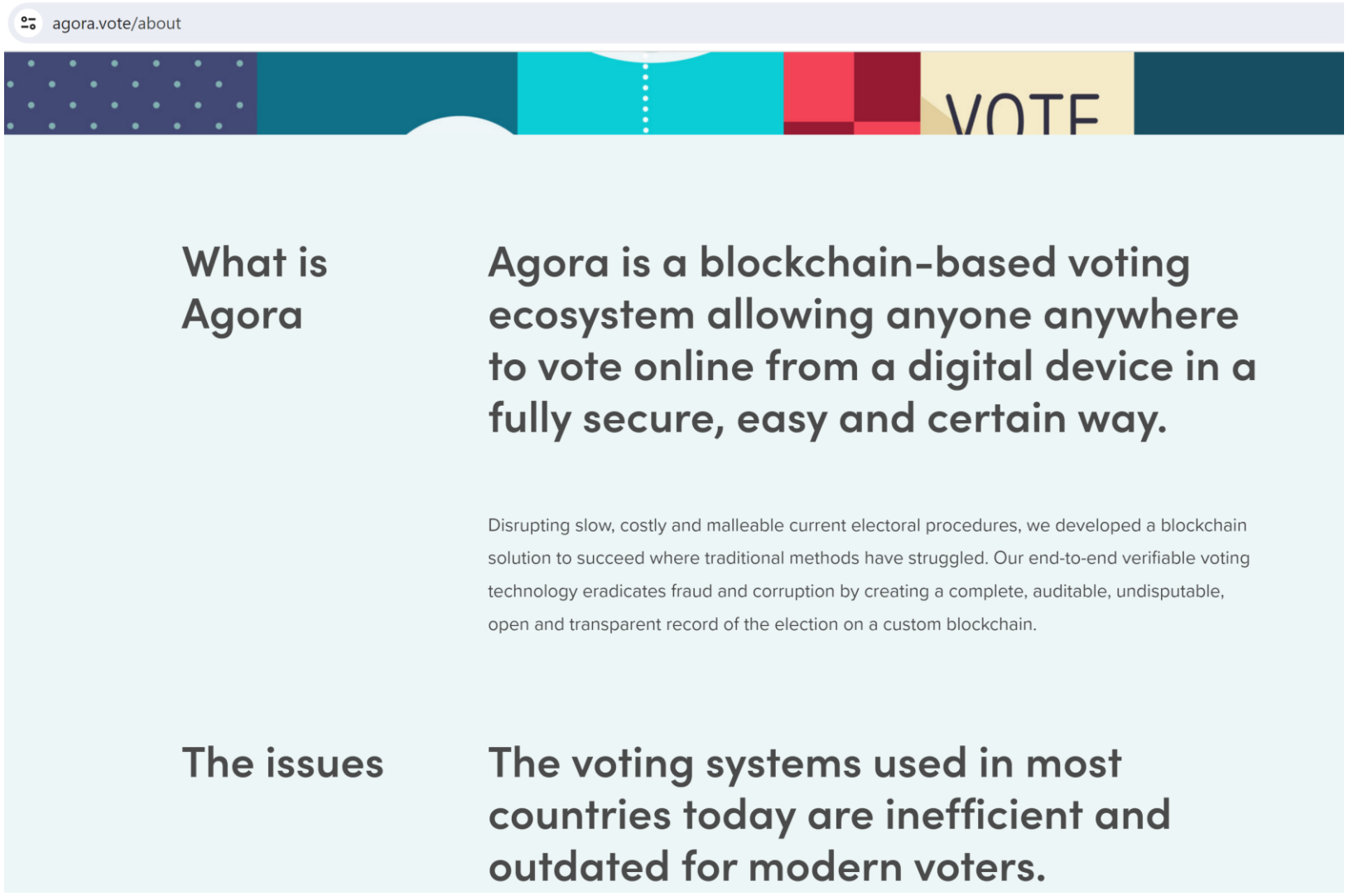
- ADDER[37]
- Helios[38]
- Prêt à Voter
- Punchscan
- Scantegrity
- Wombat Voting [39]
- ThreeBallot
- Bingo Voting
- homomorphic secret sharing
- DRE-i[40] (E2E verifiable e-voting without tallying authorities based on pre-computation)
- DRE-ip[41] (E2E verifiable e-voting without tallying authorities based on real-time computa

https://en.wikipedia.org/wiki/
End-to-end_auditable_voting_systems



https://github.com/topics/blockchain-voting

Example : https://agora.vote



Agora BlockChain
… mixed Hashes with BitCoin BlockChain

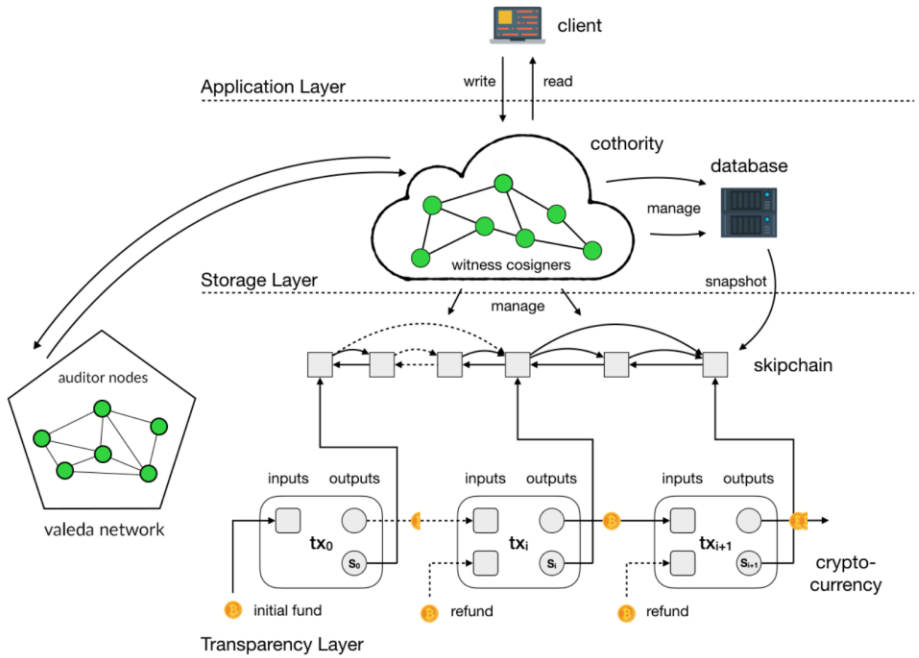
https://www.agora.vote/technology

Learn more in our Whitepaper.

static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5f37eed8cedac41642edb534/1597501378925/Agora_Whitepaper.pdf

Agora_Whitepaper.pdf 16 / 41 100%

Agora is composed of five technology layers: the *Bulletin Board* blockchain, *Cotena*, the Bitcoin blockchain, the *Valeda* network and *Votapp*. These layers communicate with each other at various instances throughout the election process to provide a cryptographically secure voting environment with auditable proofs. A visualization of our technology layers is provided below.



Conclusion

"Trust" is at the core of commerce / bank / insurance / risks / votes / ...

"De-Centralized" Trust model may be a 2.0 disruptive change

Traditional institutions did not enter the game yet of Crypto Money & Smart Contracts

Only small projects, only the beginning

BitCoin may be a investment speculative bubble, but there is more

Questions ?

arnaud.nauwynck@gmail.com