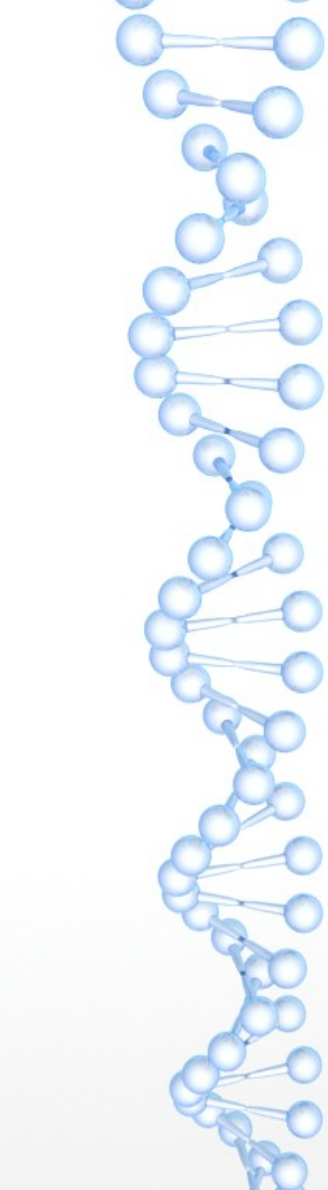
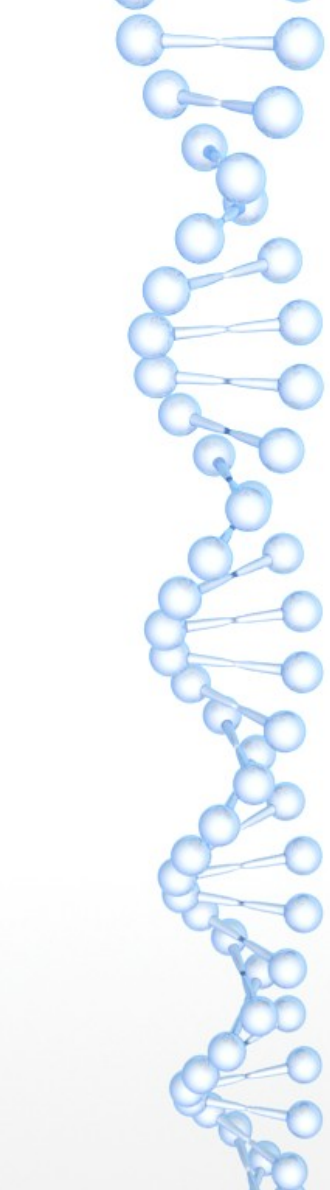


Introduction to fundamental concepts of a public blockchain (Bitcoin).

Gheorghe Curelet-Balan

KWLUG, Oct. 7, 2019

t: @curelet



Who am I?

- Interested in researching the latest advancements in software technologies / dev process (blockchain, ML, AI, IoT, web, cloud, microservices (DDD), security, mobile, enterprise, open source & agile (lean) development)
- Computer science university Math 4 year, 1983 graduate (Magna cum Laude) in Iasi, Romania
- 10 years of software engineering / testing / DB / technical project management at Romania's biggest computer science research institute ITCI
- 6 years of industrial networks communication programming (PLC drivers network gateway emulation & networks configuration Windows tools) at Molex & previous acquired businesses
- 2 years of mobile operating systems development at RIM / Blackberry
- 6 years programming of embedded telephony, VoIP, communication networks (integration, configuration, monitoring, security, DB, GUI) at GE Security / United Technologies
- 1 year of industrial automation / internet traffic filtering at PEER Group / Netsweeper

BITCOIN, the first implementation of a BLOCKCHAIN

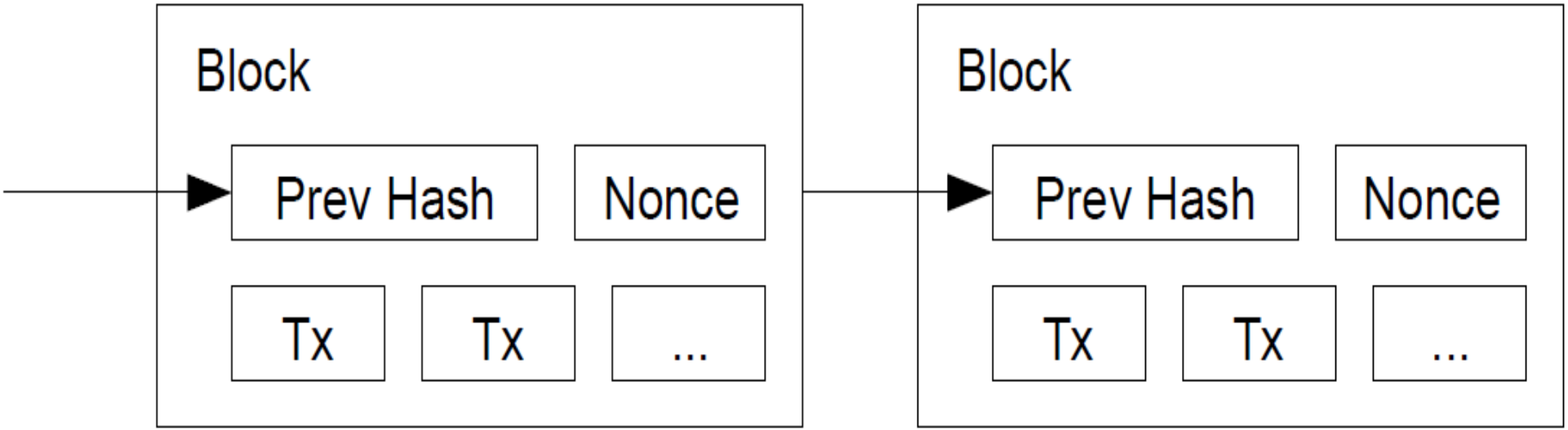
When speaking of blockchain most people think of Bitcoin digital currency (coin).

- It allows trusted anonymous peer-to-peer (P2P) public transfer of money (as bitcoins) all over the world
  - without any intermediaries (banks, governments, etc.)
- Bitcoin is implemented as a BLOCKCHAIN
  - a “trusted” shared public ledger (log) of TRANSACTIONS
  - maintained by an open “decentralized”, fault-tolerant NETWORK of computers

Bitcoin & Blockchain contextual meaning

- Shared trusted LEDGER
- NETWORK of computers maintaining the ledger (each node builds its copy of the ledger)
- COIN as network incentive and form of value storage & transfer

Blockchain ledger/log structure in a nutshell



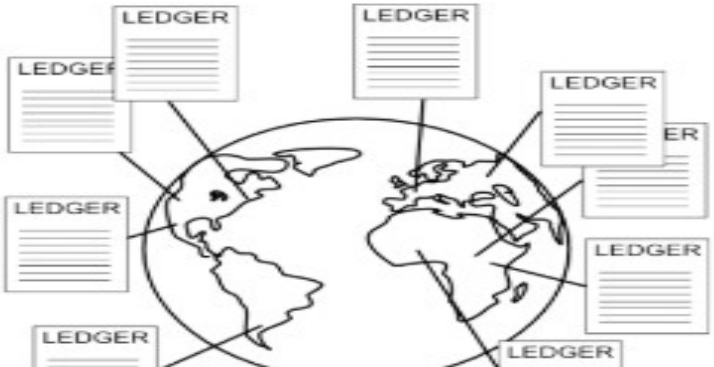
Bitcoin as immutable ledger of assets (Who owns what)

- Ledger DB = permanent distributed shared recorded history of asset TRANSACTIONS (Bitcoin also called “Internet of Money”)
- Transaction (Tx) = exchange of ASSETS
  - Derived bitcoin requirements:
    - Immutable (hard to change since asserts = value)
    - Decentralized (assets --> no central authority) to enhance TRUST
    - Ledger DB = storage of immutable history of Tx

Bitcoin DB = List of account balances & history of all transactions since the beginning of chain

Not secure www.imponderablethings.com/2013/07/how-bitcoin-works-under-hood.html#more

At its core, Bitcoin is just a digital file that lists accounts and money like a ledger. A copy of this file is maintained on every computer in the Bitcoin network. (update: you don't have to maintain a ledger just to use Bitcoin to send and receive money, this is for people who want to help maintain the system).



Ledger

Alice	5.3
Bob	100
Frank	700
Carlos	3
Jane	1.3
Charlie	4.645
Scott	.00000001
Kristin	1

...

Different record keeping models

- Bitcoin UTXO (Unspent Transaction Output) is different from abstract model in previous slide. User's bitcoins partial amount is implemented as a chain of ownership transfers (wallet app determines user's total amount).
- Ethereum blockchain implements the Account / Balance model, as in previous slide
  - https://medium.com/@sunflora98/utxo-vs-account-balance-model-5e6470f4e0cf
- Cardano, academic driven blockchain, implements both models: UTXO & Account due to its 2 implementation layers

Reasons for Bitcoin birth

- Failure of DB based CENTRALIZED payment transactions processing software architectures because:
  - Are vulnerable, being single point of failure even if DB is distributed, e.g. financial DBs
  - Of lack of built-in TRUST (intermediation makes it hard to solve disputes)
  - Are subject to DOUBLE-SPEND attack, e.g. DigiCash centralization
  - Cypherpunks libertarian crypto-anarchy voluntarism movement.
    - Bitcoin launch triggered by lack of transparency & centralized bureaucracy failure in 2008 financial crisis
    - Cypherpunks agenda: “automation of voluntary economic interactions by decreasing the cognitive load to perform them e.g. coordinated automation of roads use & maintenance.”(credit: cypherpunk youtube interview)

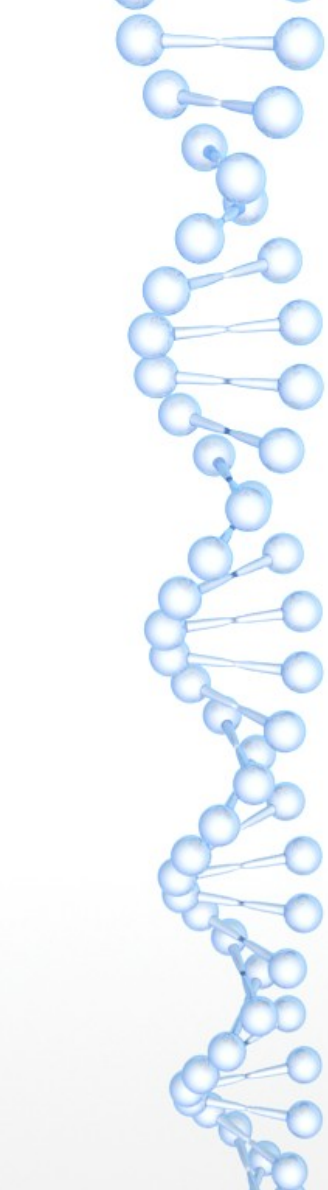
Bitcoin: A Peer-to-Peer Electronic Cash System

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

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

1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot

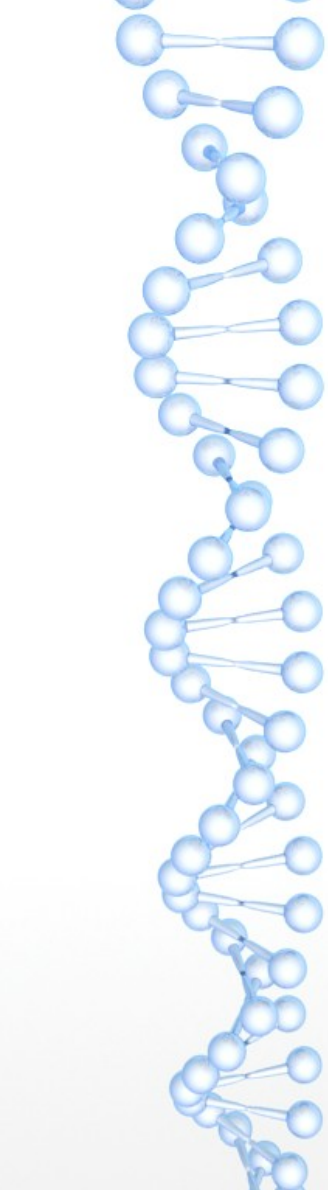


Bitcoin operation?

It’s COMPLICATED!

Bitcoin Tx initiation details

- User generates locally a private/public key pairs using a wallet app (https://en.bitcoin.it/wiki/Wallet) or other tool (if it is an Bitcoin exchange player)
  - Public Key is used to generate the bitcoin ADDRESS that identify the user (owner of funds) in the transactions on the network
- Private key is used to digital sign the transactions against ownership transfer tampering
- User creates a Tx to transfer funds with another bitcoin user using his address & receiver's Bitcoin address & public key
- User's client app sends Tx over the Bitcoin network (Tx is unconfirmed)
- User waits for Tx confirmation by the network



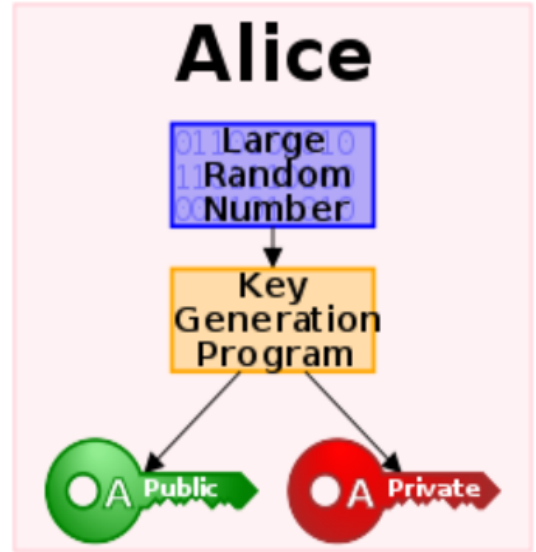
Bitcoin OWNERSHIP assured by ASYMMETRIC cryptography

- Private key
- Public key
- User's many Bitcoin addresses (managed by wallet)
- Digital signatures
- Network's Proof of Work (Pow) to assure ledger consistency (network consensus) & avoid double spending
  - Makes it hard & slow to write to ledger

Public-key cryptography, or **asymmetric cryptography**, is a cryptographic system that uses pairs of keys: *public keys* which may be disseminated widely, and *private keys* which are known only to the owner. The generation of such keys depends on cryptographic algorithms based on mathematical problems to produce one-way functions. Effective security only requires keeping the private key private; the public key can be openly distributed without compromising security.[1]

In such a system, any person can encrypt a message using the receiver's *public key*, but that encrypted message can only be decrypted with the receiver's *private key*.

Robust authentication is also possible. A sender can combine a message with a private key to create a short *digital signature* on the message. Anyone with the corresponding public key can combine a message, a putative digital signature on it, and the known public key to verify whether the signature was valid, i.e. made by the owner of the corresponding private key.[2][3]



An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

Bob

Bitcoin account

Like a public drawer whose random label is derived from the public key

- e.g. generate public / private key pair at Bitaddress.org
- Bitcoin address algorithm:

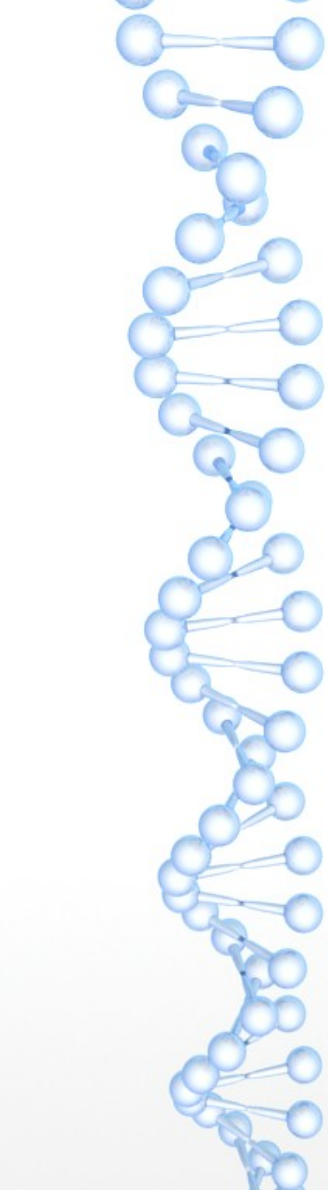
en.bitcoin.it/wiki/Protocol_documentation#Addresses

Addresses

A bitcoin address is in fact the hash of a ECDSA public key, computed this way:

```
Version = 1 byte of 0 (zero); on the test network, this is 1 byte of 111
Key hash = Version concatenated with RIPEMD-160(SHA-256(public key))
Checksum = 1st 4 bytes of SHA-256(SHA-256(Key hash))
Bitcoin Address = Base58Encode(Key hash concatenated with Checksum)
```

The Base58 encoding used is home made, and has some differences. Especially, leading zeroes are kept as single zeroes when conversion happens.



Building Transactions (Tx)

Building Tx

- Specify amount
- Specify source & destination addresses
- Wallet app
  - Uses user’s private key to DIGITAL SIGN Tx
    - Verifier uses user’s public key to check if he initiated Tx

A bitcoin TRANSACTION as shown in https://blockexplorer.com

Transaction

Transaction a601f7bf89c1f9d8a6dfc75e6ba02eddaf1cb4c1e7f36cd07b90cc0c90f98ce1

Summary

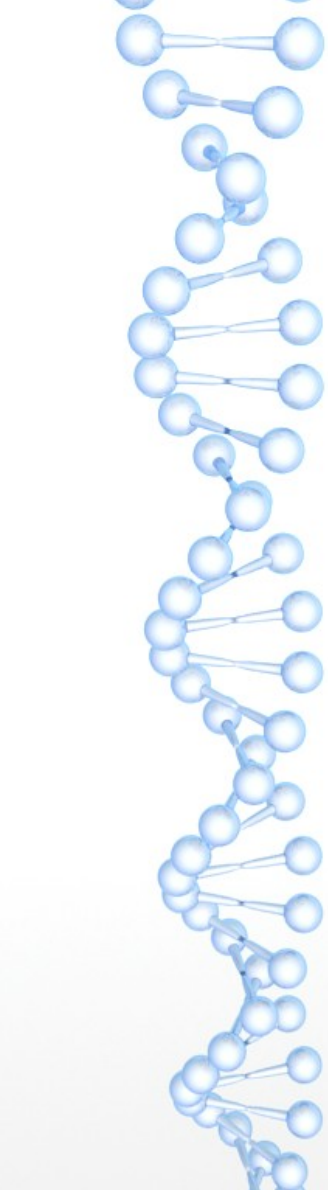
Size	356 (bytes)
Fee Rate	0.0028089887640449437 BTC per kB
Received Time	Nov 14, 2018 8:55:06 PM
Mined Time	Nov 14, 2018 8:55:06 PM
Included in Block	000000000000000000205747634b27e03998da41c52865e2fccca3b417c75772

Details

a601f7bf89c1f9d8a6dfc75e6ba02eddaf1cb4c1e7f36cd07b90cc0c90f98ce1 mined Nov 14, 2018 8:55:06 PM

17A16QmavnUfCW11DAApiJxp7ARnxN5pGX	1001.94025673 BTC	3BMEXCCkC9vJgxFXxanmaJVMdwnG3qi1BN	0.0044 BTC (U)
		1DdFPGn2Rr8XeL71B3srz5DiadSqUcCzT9	0.1995 BTC (U)
		1KNG1PiHcVTmjeMDoX7WYiCXWyaf7oqf13	3.7995 BTC (U)

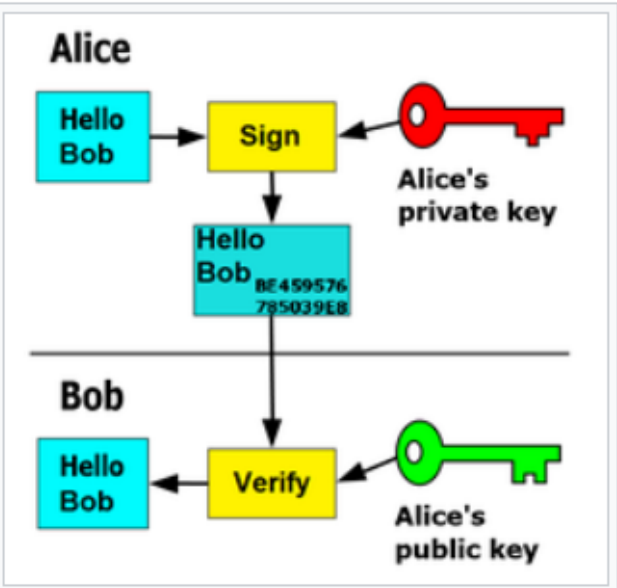
1DdFPGn2Rr8XeL71B3srz5DiadSqUcCzT9	0.1995 BTC (U)
1KNG1PiHcVTmjeMDoX7WYiCXWyaf7oqf13	3.7995 BTC (U)



Digital Signature

Two of the best-known uses of public key cryptography are:

- *Public key encryption*, in which a message is encrypted with a recipient's public key. The message cannot be decrypted by anyone who does not possess the matching private key, who is thus presumed to be the owner of that key and the person associated with the public key. This is used in an attempt to ensure confidentiality.
- *Digital signatures*, in which a message is signed with the sender's private key and can be verified by anyone who has access to the sender's public key. This verification proves that the sender had access to the private key, and therefore is likely to be the person associated with the public key. This also ensures that the message has not been tampered with, as a signature is mathematically bound to the message it originally was made with, and verification will fail for practically any other message, no matter how similar to the original message.



In this example the message is only digitally signed and not encrypted. 1) Alice signs a message with her private key. 2) Bob can verify that Alice sent the message and that the message has not been modified.

One important issue is confidence/proof that a particular public key is authentic, i.e. that it is correct and belongs to the person or entity claimed, and has not been tampered with or replaced by a malicious third party. There are several possible approaches, including:

A public key infrastructure (PKI), in which one or more third parties – known as certificate authorities – certify ownership of key pairs. TLS relies upon this.

A "web of trust" which decentralizes authentication by using individual endorsements of the link between user and public key. PGP uses this approach.

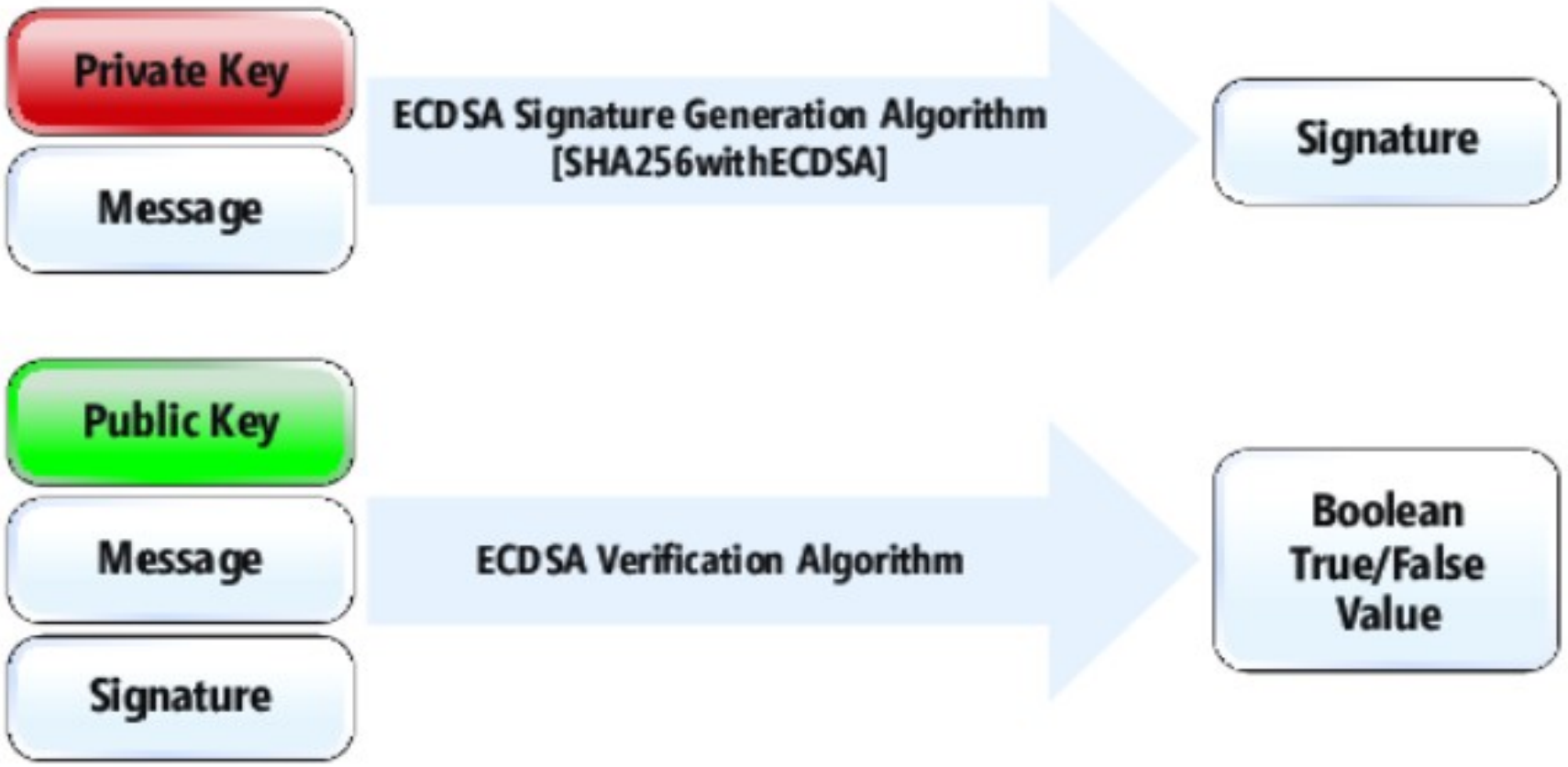


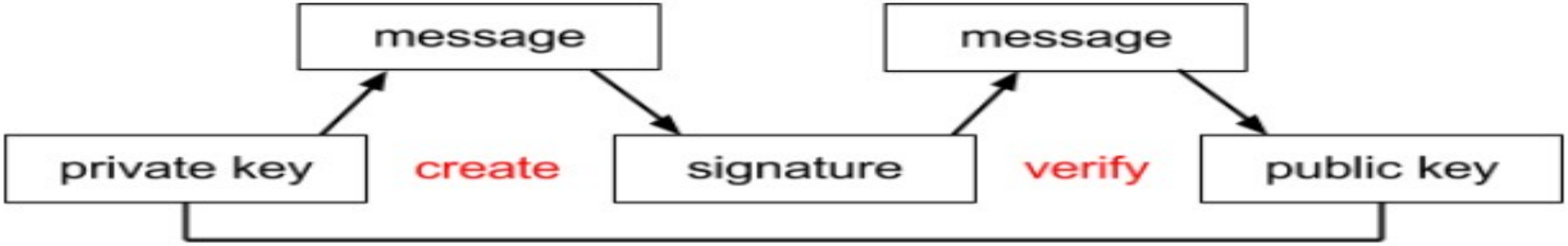
Figure 2 Elliptic Curve Digital Signature Algorithm Signature Generation (Top) and Verification Algorithm (Bottom)

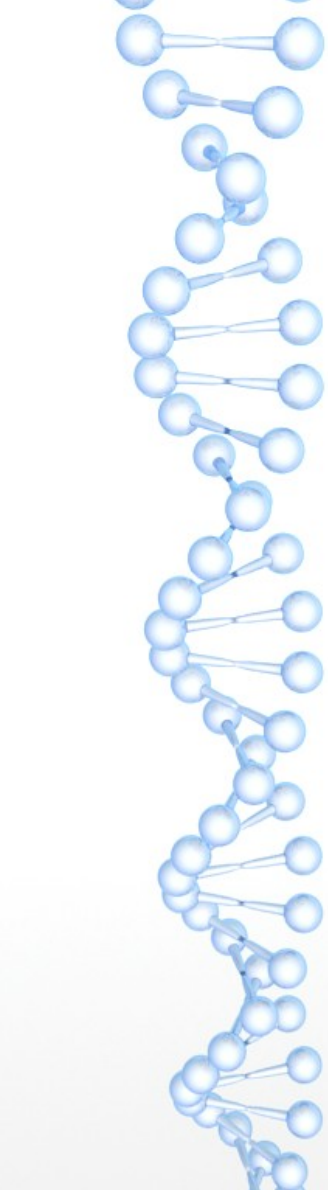
Transaction Messages

		Digital Signature
Alice → Bob	5.0 BTC	04323784...
Alice → Dave	12 BTC	88432738...
Alice → Juan	2000 BTC	00328434...
Alice → Bob	14 BTC	19382637...

^
different every time

A Digital Signature works by utilizing two different (but connected) keys, a "private key" to create a signature, and a "public key" that others can use to check it.



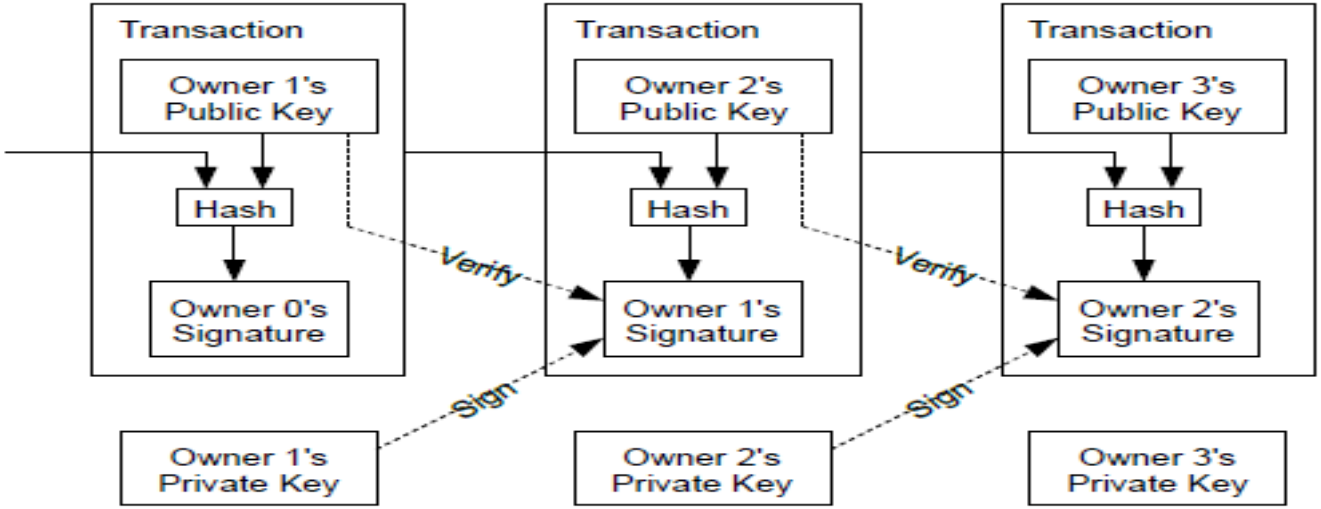


Bitcoin coins chain of ownership (UTXO)

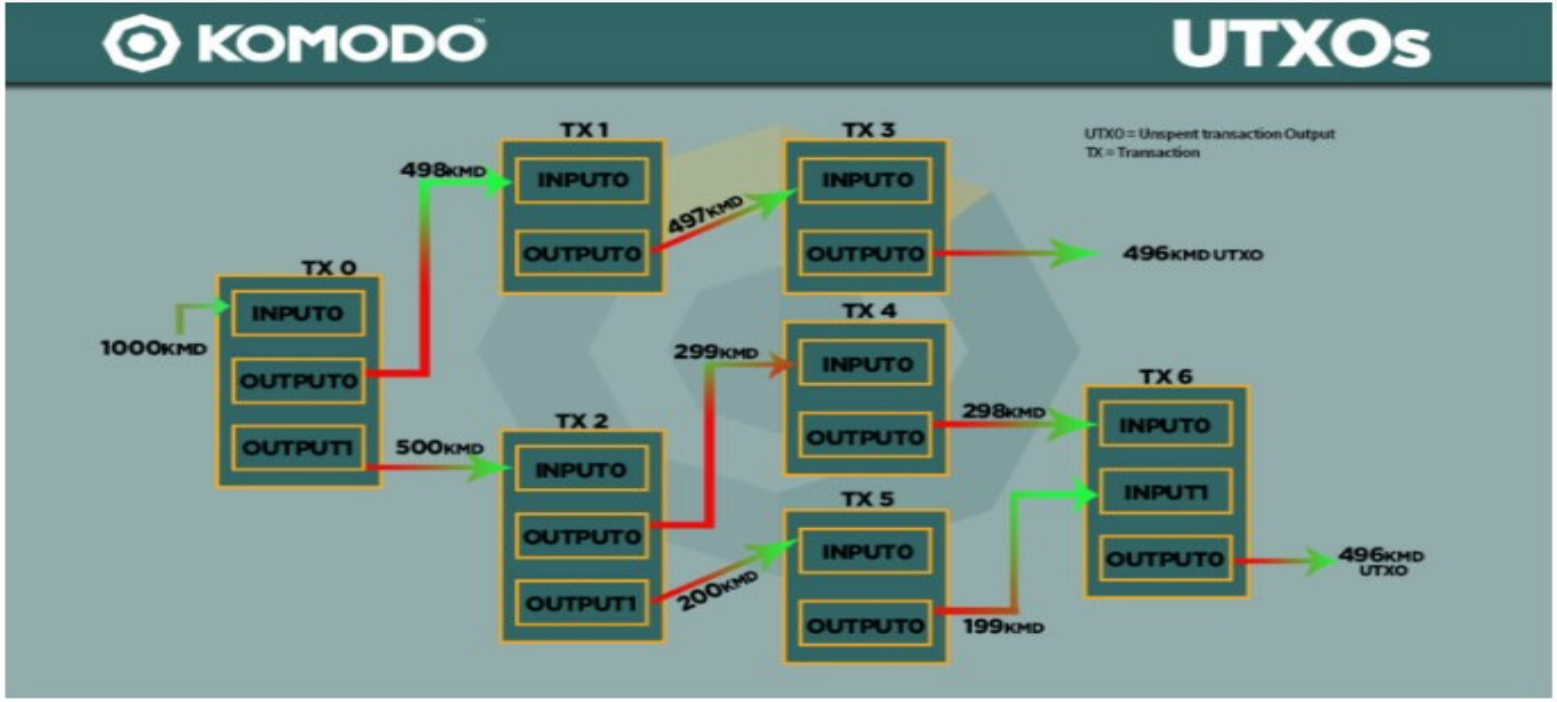
Satoshi's definition of the Bitcoin coin

2. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

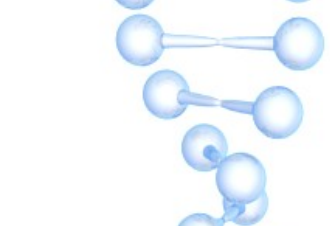


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.



The second way in which our previous UTXO analogy breaks down is that fiat bills are fixed in value. In other words, fiat bills are limited to the value that governments choose to print in.

In the USA, the only bills in existence are: $1, $5, $10, $20, $50, and $100. In nations that have adopted the Euro, the only bill denomination are: €5, €10, €20, €50, €100, €200 and



New Coins creation as network reward incentive (ref. Satoshi paper)

6. Incentive

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

Understanding Bitcoin network of miners (ledger administrators)

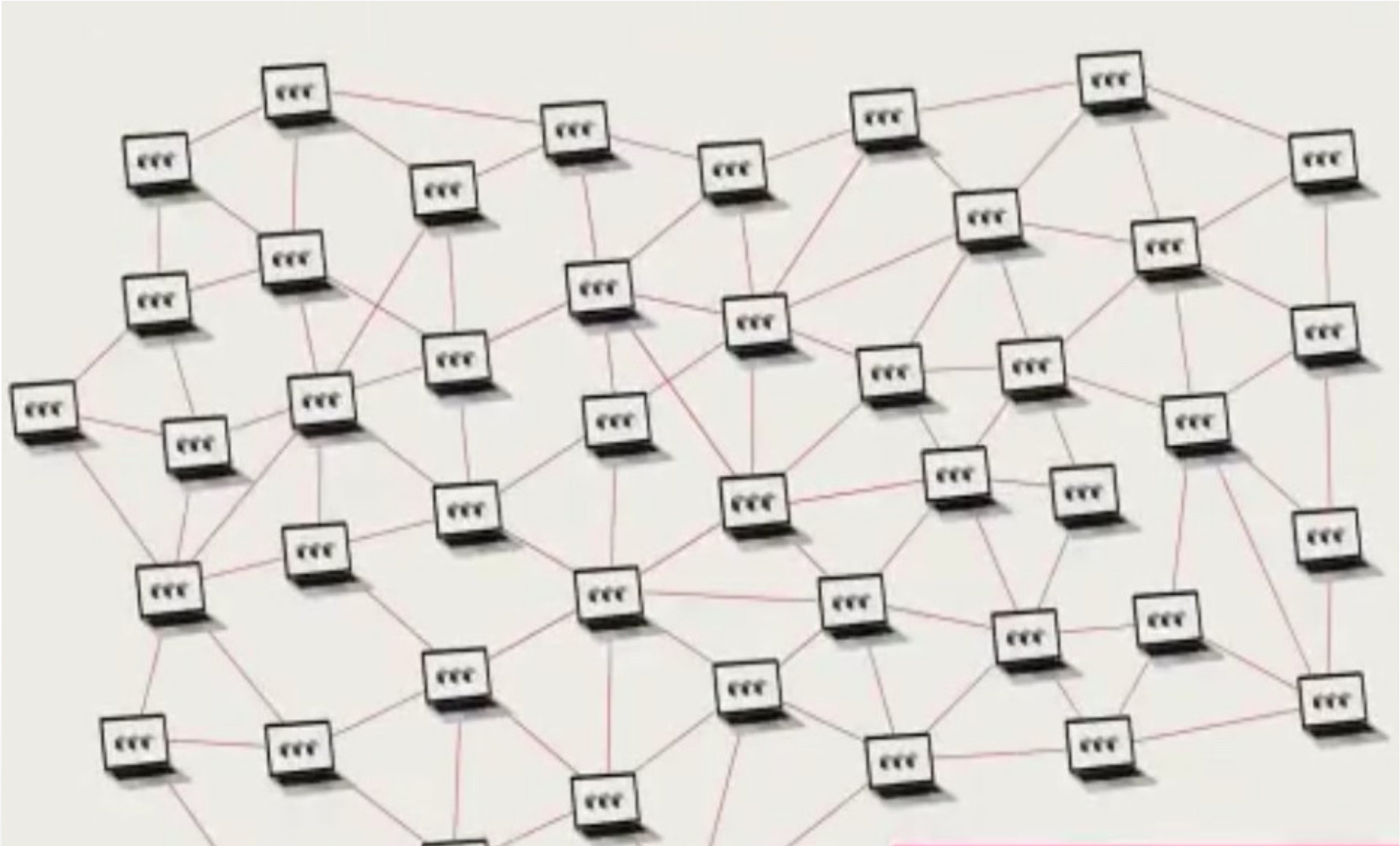
- How it operates perspective

Bitcoin network (miners) competes to build next block

Step 1:

- All unconfirmed Tx sent by clients are propagated throughout the Bitcoin mesh network.

Bitcoin decentralized P2P network (CIGI view)



Bitcoin network

- a.k.a. DLT network (Distributed Ledger Technology).
- Mesh network of 10K nodes (https://bitnodes.earn.com). Each node connects with few neighbours who relay the traffic (Tx & block forwarding)
- Permissionless: nodes can join & leave at will
- Clients communicate with the nearest node
- No single point of failure
- The network offers trust & protection against ledger attacks by periodically (10 min) peer block replication (1 block at a time) & selectively offering the write privilege to a peer via PROOF OF WORK (PoW) protocol.
- Trustless & BFT (Byzantine Fault-Tolerant) operating in the most adversarial environment (can have malicious nodes).
  - Bitcoin PoW protocol guarantees ledger integrity as long as 51% of nodes are honest.

Bitcoin network competes to build next block

Step 2, Block Creation:

- Each node (miner) verifies each received Tx & shelves it (in a mempool) for consideration in next block
- Once current block is finalized (via PoW), the 10 minutes race to build next block begins
- Each node collects most (or all) unconfirmed transactions into a block up to 1MB size limit (block size is a debatable issue in Bitcoin). Tx selection depends also by the Tx fee paid to miner
- Due to network delay in Tx propagation the blocks built by the farthest apart nodes might be different. PoW will decide which block wins.

Transaction Verification

Transactions are cryptographically signed records that reassign ownership of Bitcoins to new addresses. Transactions have *inputs* - records which reference the funds from other previous transactions - and *outputs* - records which determine the new owner of the transferred Bitcoins, and which will be referenced as inputs in future transactions as those funds are respent.

Each *input* must have a cryptographic digital signature that unlocks the funds from the prior transaction. Only the person possessing the appropriate private key is able to create a satisfactory signature; this in effect ensures that funds can only be spent by their owners.

Each *output* determines which Bitcoin address (or other criteria, see Script) is the recipient of the funds.

In a transaction, the sum of all inputs must be equal to or greater than the sum of all outputs. If the inputs exceed the outputs, the difference is considered a transaction fee, and is redeemable by whoever first includes the transaction into the block chain.

A special kind of transaction, called a coinbase transaction, has no inputs. It is created by miners, and there is one coinbase transaction per block. Because each block comes with a reward of newly created Bitcoins (e.g. 50 BTC for the first 210,000 blocks), the first transaction of a block is, with few exceptions, the transaction that grants those coins to their recipient (the miner). In addition to the newly created Bitcoins, the coinbase transaction is also used for assigning the recipient of any transaction fees that were paid within the other transactions being included in the same block. The coinbase transaction can assign the entire reward to a single Bitcoin address, or split it in portions among multiple addresses, just like any other transaction. Coinbase transactions always contain outputs totalling the sum of the block reward plus all transaction fees collected from the other transactions in the same block.

The coinbase transaction in block zero cannot be spent. This is due to a quirk of the reference client implementation that would open the potential for a block chain fork if some nodes accepted the spend and others did not[1].

Most Bitcoin outputs encumber the newly transferred coins with a single ECDSA private key. The actual record saved with inputs and outputs isn't necessarily a key, but a *script*. Bitcoin uses an interpreted scripting system to determine whether an output's criteria have been satisfied, with which more complex operations are possible, such as outputs that require two ECDSA signatures, or two-of-three-signature schemes. An output that references a single Bitcoin address is a *typical* output; an output actually contains this information in the form of a script that requires a single ECDSA signature (see OP_CHECKSIG). The output script specifies what must be provided to unlock the funds later, and when the time comes in the future to spend the transaction in another input, that input must provide all of the thing(s) that satisfy the requirements defined by the original output script.

block

The **block** message is sent in response to a getdata message which requests transaction information from a block hash.

Field Size	Description	Data type	Comments
4	version	int32_t	Block version information (note, this is signed)
32	prev_block	char[32]	The hash value of the previous block this particular block references
32	merkle_root	char[32]	The reference to a Merkle tree collection which is a hash of all transactions related to this block
4	timestamp	uint32_t	A Unix timestamp recording when this block was created (Currently limited to dates before the year 2106!)
4	bits	uint32_t	The calculated difficulty target being used for this block
4	nonce	uint32_t	The nonce used to generate this block… to allow variations of the header and compute different hashes
1+	txn_count	var_int	Number of transaction entries
?	txns	tx[]	Block transactions, in format of "tx" command

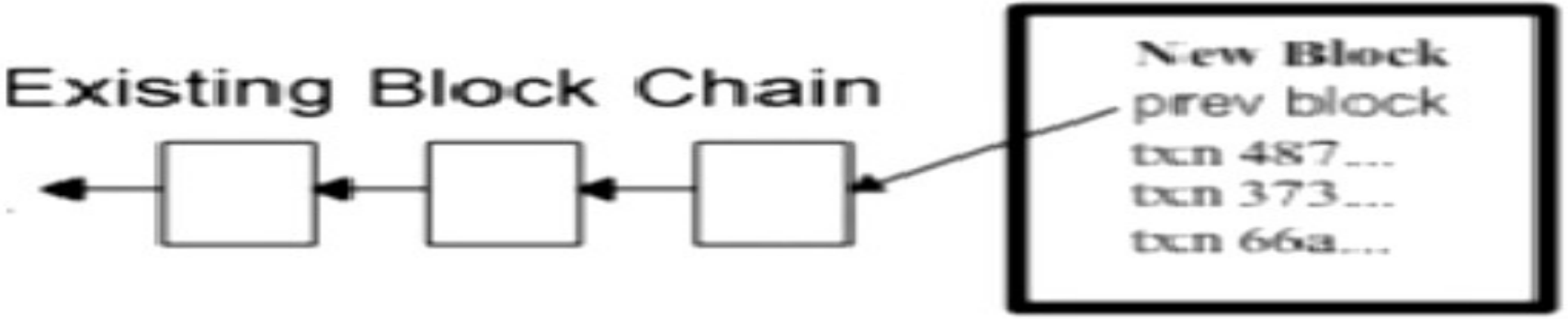
The SHA256 hash that identifies each block (and which must have a run of 0 bits) is calculated from the first 6 fields of this structure (version, prev_block, merkle_root, timestamp, bits, nonce, and standard SHA256 padding, making two 64-byte chunks in all) and *not* from the complete block. To calculate the hash, only two chunks need to be processed by the SHA256 algorithm. Since the *nonce* field is in the second chunk, the first chunk stays constant during mining and therefore only the second chunk needs to be processed. However, a Bitcoin hash is the hash of the hash, so two SHA256 rounds are needed for each mining iteration. See Block hashing algorithm for details

Block Headers

Block headers are sent in a headers packet in response to a getheaders message.

Field Size	Description	Data type	Comments
4	version	int32_t	Block version information (note, this is signed)
32	prev_block	char[32]	The hash value of the previous block this particular block references
32	merkle_root	char[32]	The reference to a Merkle tree collection which is a hash of all transactions related to this block
4	timestamp	uint32_t	A timestamp recording when this block was created (Will overflow in 2106[2])
4	bits	uint32_t	The calculated difficulty target being used for this block
4	nonce	uint32_t	The nonce used to generate this block... to allow variations of the header and compute different hashes
1+	txn_count	var_int	Number of transaction entries, this value is always 0

Block Creation



Algorithms used to build a tamper proof block

- SHA256 hashing algorithm
  - used to digital fingerprint Tx & the block
- Merkle tree algorithm
  - A binary tree of hashes
    - https://en.bitcoin.it/wiki/Protocol_documentation#Merkle_Trees
    - Leaf nodes are TX hashes
    - Each parent node is a concatenation of its children hashes
  - used to digital fingerprint all Tx
    - To prove fast if a Tx is part of the block
    - To speed up block hashing by only using its header where Merkle tree root is

Merkle Trees

Merkle trees are binary trees of hashes. Merkle trees in bitcoin use a **double** SHA-256, the SHA-256 hash of the SHA-256 hash of something.

If, when forming a row in the tree (other than the root of the tree), it would have an odd number of elements, the final double-hash is duplicated to ensure that the row has an even number of hashes.

First form the bottom row of the tree with the ordered double-SHA-256 hashes of the byte streams of the transactions in the block.

Then the row above it consists of half that number of hashes. Each entry is the double-SHA-256 of the 64-byte concatenation of the corresponding two hashes below it in the tree.

This procedure repeats recursively until we reach a row consisting of just a single double-hash. This is the **Merkle root** of the tree.

For example, imagine a block with three transactions *a*, *b* and *c*. The Merkle tree is:

```
d1 = dhash(a)
d2 = dhash(b)
d3 = dhash(c)
d4 = dhash(c)            # a, b, c are 3. that's an odd number, so we take the c twice

d5 = dhash(d1 concat d2)
d6 = dhash(d3 concat d4)

d7 = dhash(d5 concat d6)
```

where

```
dhash(a) = sha256(sha256(a))
```

d7 is the Merkle root of the 3 transactions in this block.

Note: Hashes in Merkle Tree displayed in the Block Explorer are of little-endian notation. For some implementations and calculations, the bytes need to be reversed before they are hashed, and again after the hashing operation.

Signatures

SHA256 Hashing

One way function offers unique numeric identity (256 bits hash) to any digital content. Hash is highly sensitive to change.

SHA256("short sentence")

0x 0acdf28f4e8b00b399d89ca51f07fef34708e729ae15e85429c5b0f403295cc9

SHA256("The quick brown fox jumps over the lazy **dog**")

0x d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592

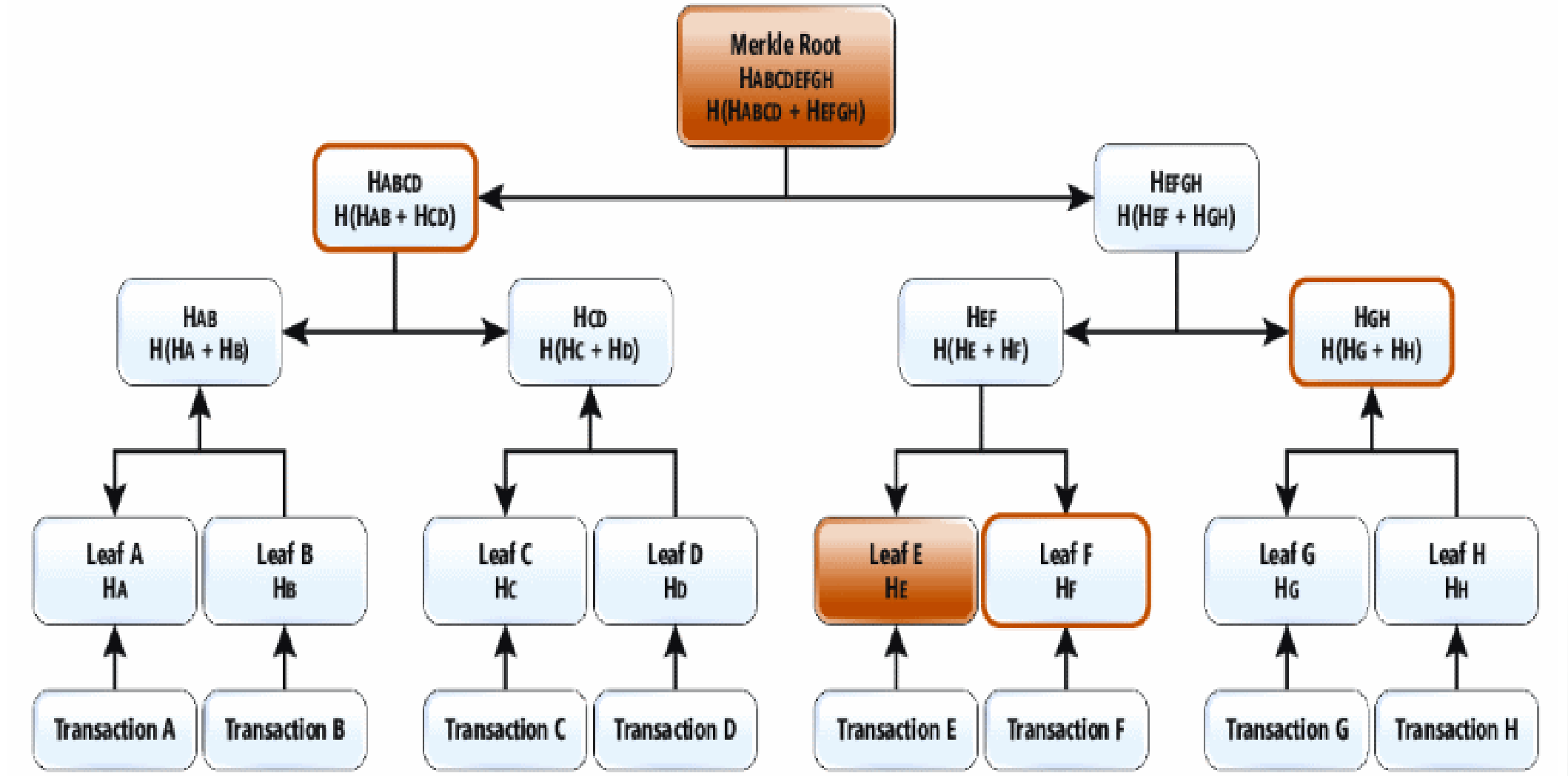
SHA256("The quick brown fox jumps over the lazy **dog.**")

(extra period added)

0x ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c

Merkle path proves Tx belonging to blockchain

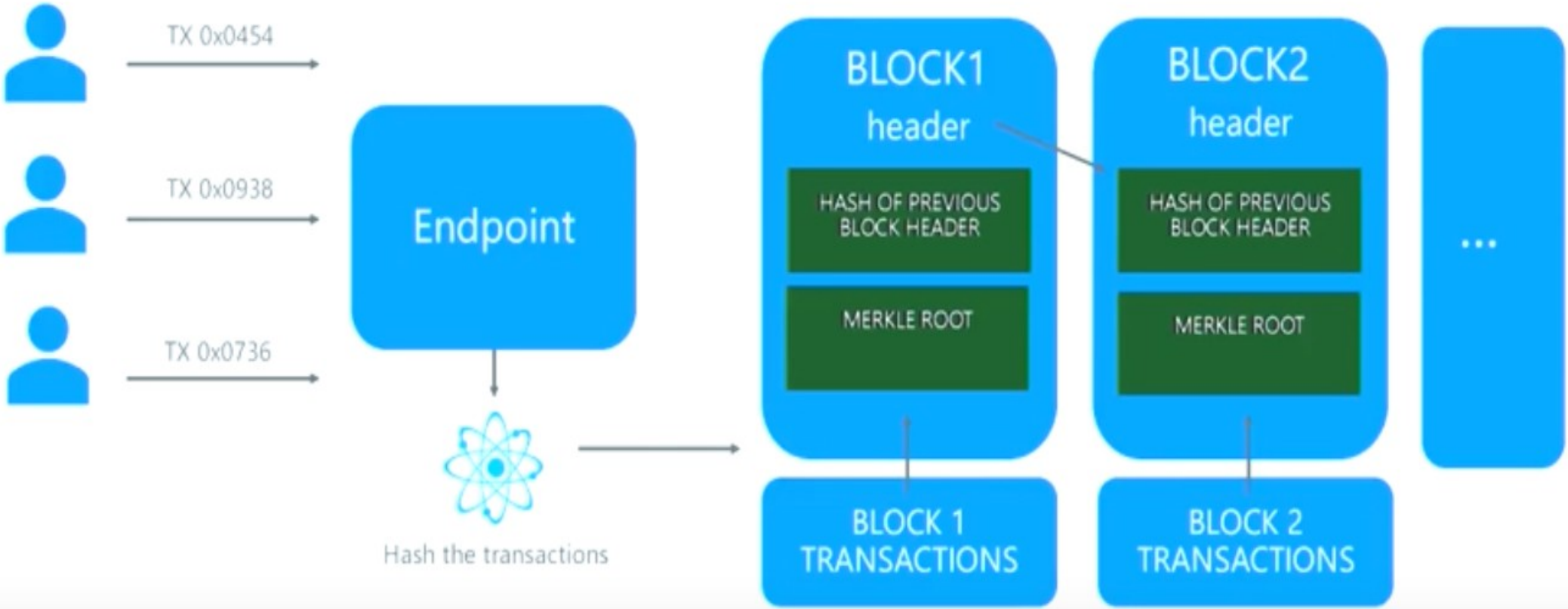
- A light blockchain client (phone wallet) queries blockchain if it contains a Tx
- The phone stores only the chain of blockchain headers
- Phone receives the Merkle path to that Tx, i.e. only the required Merkle Tree hashes (in Tx path) to compute the Merkle root
- Phone computes the block's Merkle root & compares it with that in the block's header



Building a tamper proof block

- Block structure:
  - Block's header hash
  - Collection of all Tx
  - Header
    - Previous block header hash
    - Merkle root hash (a digital fingerprint of all Tx)
    - Nonce
- A Merkle Tree is built with all Tx
- The Merkle tree root hash is set in block's header
- Block's header hash is decentralized competitive determined by the network

Core concept – chain of blocks



Blockchain v.s. DB

- Data tamper protection by using heavy cryptography & chaining of blocks.
  - Hash chain v.s. independent DB records
    - Each block's hash is a fingerprint off all the previous data in the chain since the beginning (the genesis block)
      - Altering any data will invalidate hash of the last current block

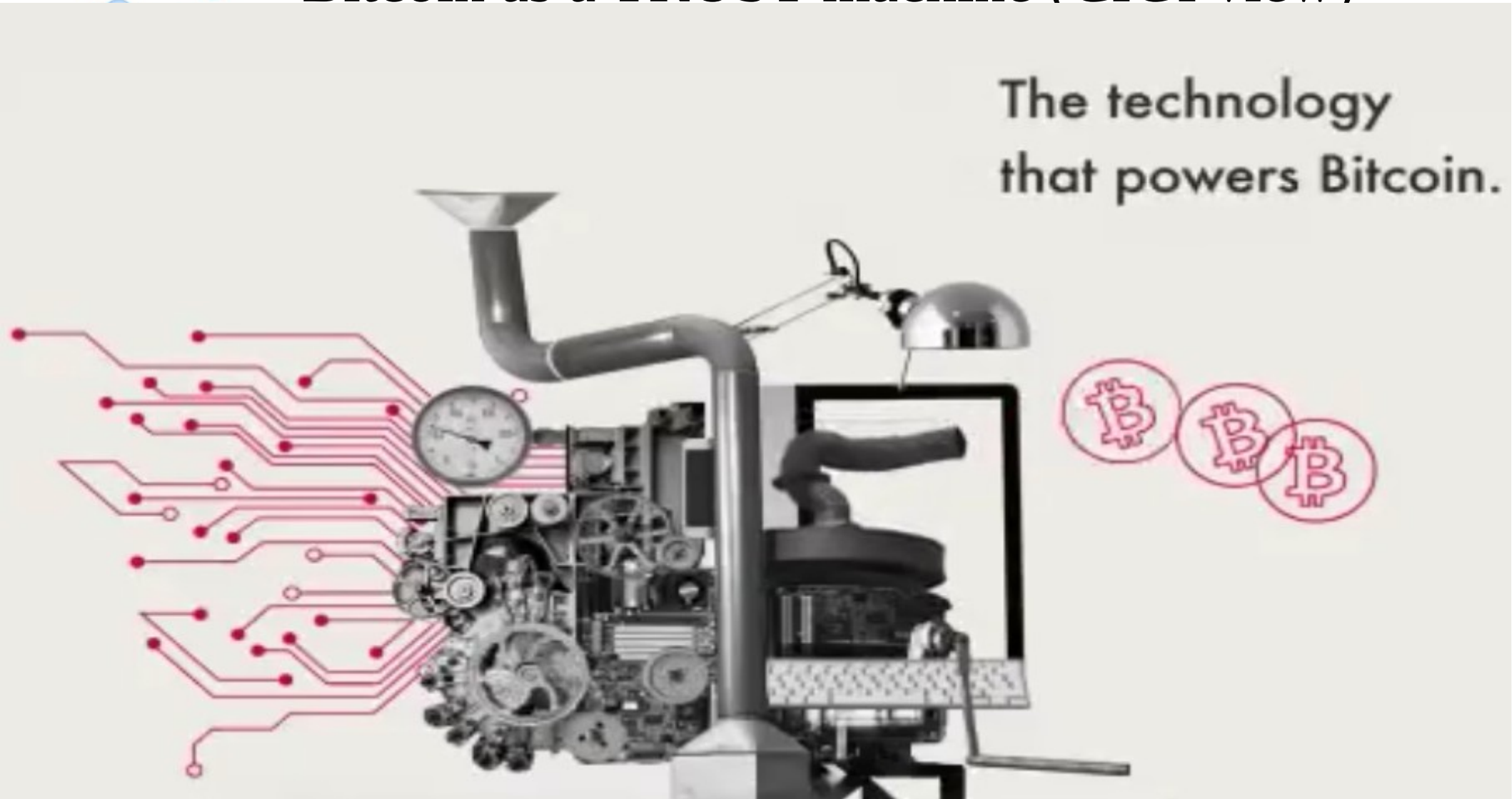
Bitcoin network competes to build next block

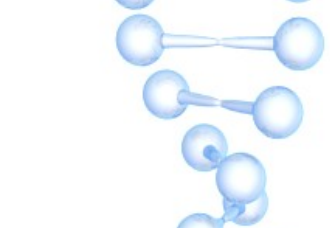
Step 3: Proof of Work (PoW):

RANDOM NODE SELECTION of chain extension privilege.

- PoW algorithm. Inspired by Adam Back's Hashcash PoW used to limit spam in message boards.
  - Each node competes to solve a CPU intensive simple math puzzle with a common target to all nodes
  - A nonce in block's header is incremented till computed block's hash is less than network's target hash ("difficulty" hash should start with a predefined number of zeros 0x0000000000000....). Kind of bingo game.
  - Difficulty is dynamically adjusted (every 2 weeks) to result in mining time being around 10 min depending on majority miners CPU power)
  - CPU power measured in Mega hashes per second

Bitcoin as a TRUST machine (CIGI view)





Miner's reward: Coinbase Tx (has no inputs) per Satoshi paper

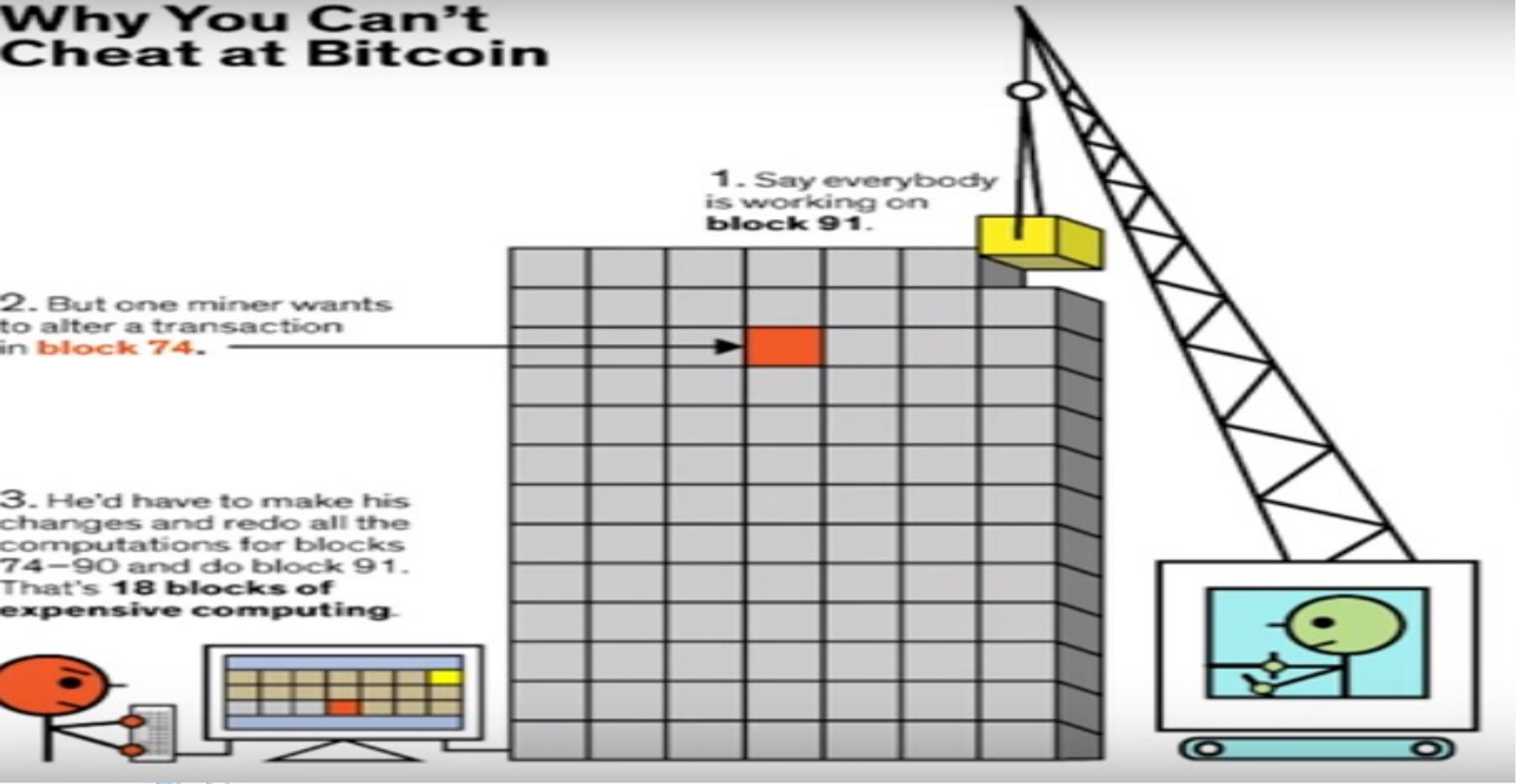
6. Incentive

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

Why You Can't Cheat at Bitcoin



Credit Avery Carter

Bitcoin Mining

13 Pins • 259 Followers

A look inside data centers specializing in bitcoin mining and other cryptocurrency transactions.

by Data Center Frontier

Share this board

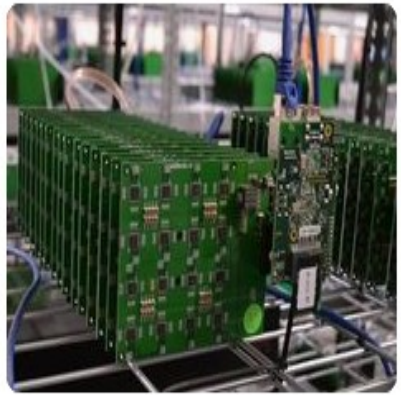
Follow



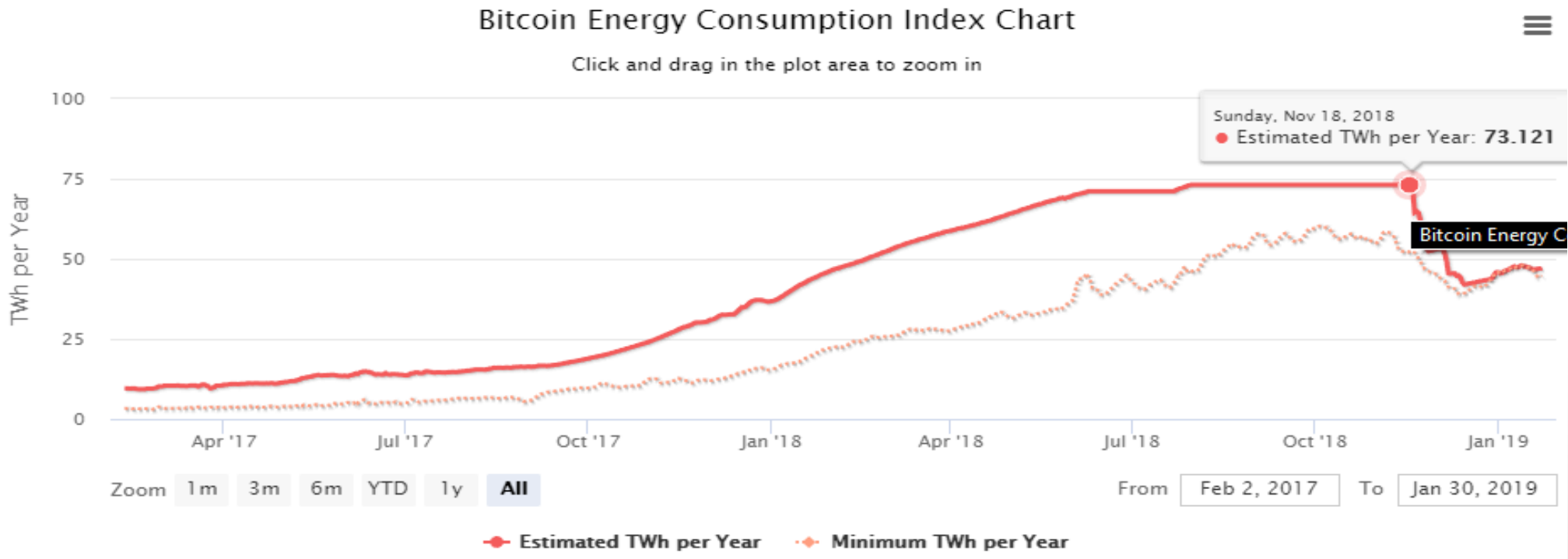









Bitcoin Energy Consumption Index



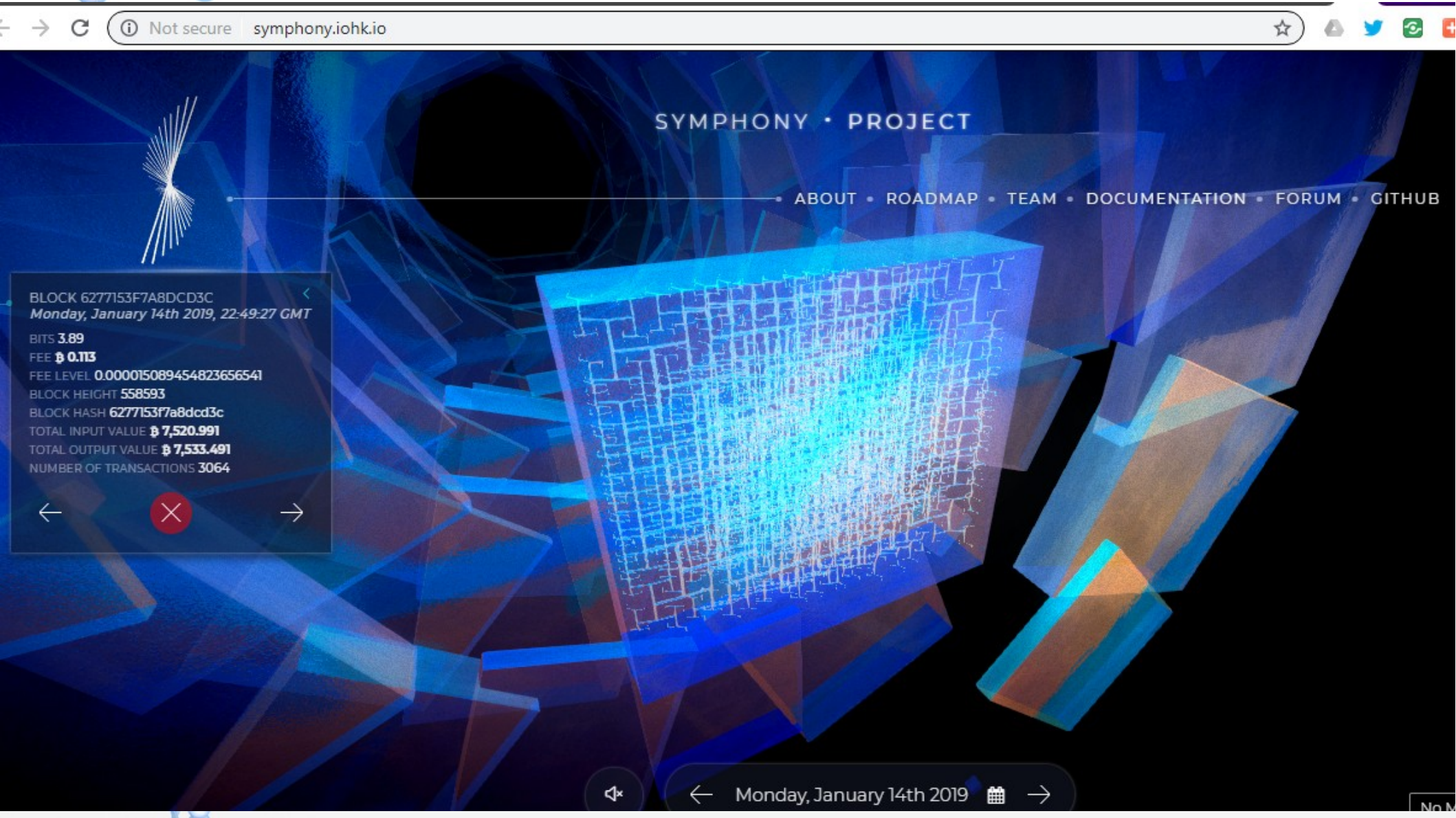
Bitcoin network competes to build next block

Step 4:

- The winning node broadcasts the found block to all nodes (block contains a Tx (coinbase) that rewards the miner with (now 12.5) bitcoins

Bitcoin network competes to build next block

- Step 5: Nodes easily verify winning block (hash < PoW target difficulty) + all Tx validity & if Tx are not already spent (TRUST by VERIFICATION).
- Step 6: Nodes accept the block (reach Nakamoto consensus) if they start building next block using the accepted block's hash as previous block.
  - If more correct blocks are received at approx the same time, nodes will extend the longest chain since it has the most PoW (hashing power) invested in (hard fork).


Not secure
symphony.iohk.io
SYMPHONY • PROJECT
ABOUT • ROADMAP • TEAM • DOCUMENTATION • FORUM • GITHUB
BLOCK 6277153F7A8DCD3C
Monday, January 14th 2019, 22:49:27 GMT
BITS 3.89
FEE ฿ 0.113
FEE LEVEL 0.000015089454823656541
BLOCK HEIGHT 558593
BLOCK HASH 6277153f7a8dcd3c
TOTAL INPUT VALUE ฿ 7,520.991
TOTAL OUTPUT VALUE ฿ 7,533.491
NUMBER OF TRANSACTIONS 3064
Monday, January 14th 2019

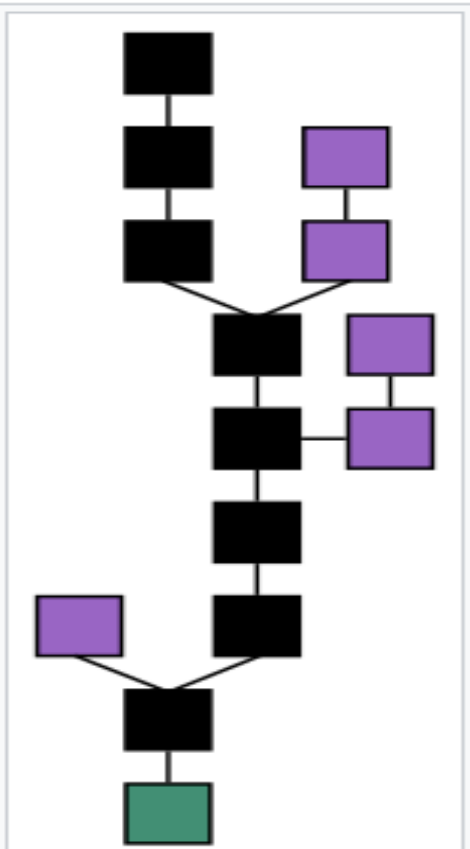
Chain Hard Fork

- Happen when a longer chain containing previous block emerges (see next image)
- Caused by PoW not finalizing the blocks
- Trust in blocks increases as time goes by (rule of 1 hour waiting (6 blocks) after Tx confirmation)

A **blockchain**,[1][2][3] originally **block chain**,[4][5] is a growing list of records, called *blocks*, which are linked using cryptography.[1][6] Each block contains a cryptographic hash of the previous block,[6] a timestamp, and transaction data (generally represented as a merkle tree root hash).

By design, a blockchain is resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way".[7] For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority. Although blockchain records are not unalterable, blockchains may be considered secure by design and exemplify a distributed computing system with high Byzantine fault tolerance. Decentralized consensus has therefore been claimed with a blockchain.[8]

Blockchain was invented by Satoshi Nakamoto in 2008 to serve as the public transaction ledger of the cryptocurrency bitcoin.[1] The invention of the blockchain for bitcoin made it the first digital currency to solve the double-spending problem without the need of a trusted authority or central server. The bitcoin design has inspired other applications,[1][3] and blockchains which are readable by the public are widely used by cryptocurrencies. Blockchain is considered a type of payment rail.[9] Private blockchains have been proposed for business use. Sources such as the *Computerworld* called the marketing of



Blockchain formation. The main chain (black) consists of the longest series of blocks from the genesis block (green) to the current block. Orphan blocks (purple) exist

Bitcoin Governance (changing chain technology)

- Via BIPs (Bitcoin Improvement Proposals)
- Changes need to be adopted by main players:
  - Developers
  - Miners
  - Exchanges
  - Wallets

What's all the fuss about & what's the promise of this technology that is causing another speculative "tulips mania"?

Hype or revolutionary payments technology?

Bitcoin features:

- ANONYMOUS users
- PUBLIC shared ledger/log/DB (TRANSPARENCY = anyone can see all transactions)
- OPEN / PERMISSIONLESS network. Anyone can join & participate in network operation (effective if participant has enough CPU power).
- DISTRIBUTED (transactions are stored (1 block every 10 minutes) on each network node)
- SECURE (hard to attack) since maintained by the distributed network (no single point of failure)
- GOVERNED (network agreed) WRITE (append only write agreed by all nodes following a network consensus protocol). Makes the write to blockchain slow & expensive due to consensus overhead. Makes the ledger IMMUTABLE as time goes by.
- TAMPER RESISTANT. Tampering with a block/Tx will invalidate the chain from there forward; chain needs to be rebuilt from the changed block, forward.
- TRUSTED using cryptography (no disputes since each node can verify for itself the shared data in each block)
- SCRIPTING feature. Limited ability to store & run scripts inside Tx

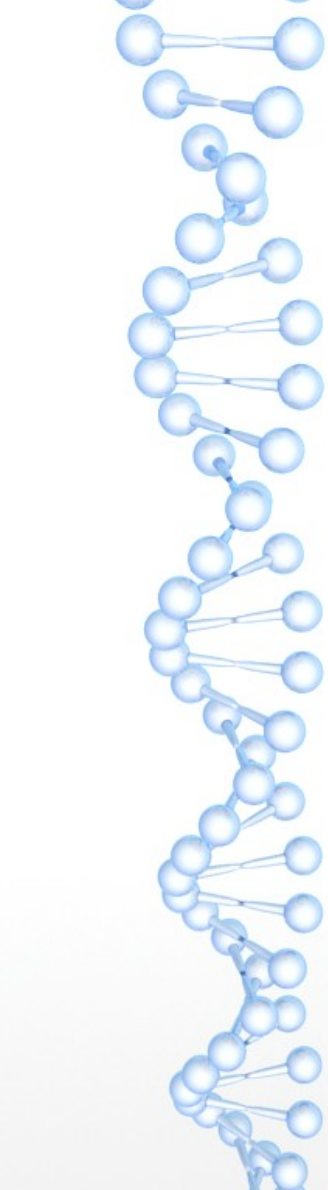
Bitcoin scripting limitations

Different types of Bitcoin applications (like P2P gambling, making own currency, decentralized exchange, financial derivatives, decentralized DNS, P2P insurance, color coins) required different types of Tx or features added to Bitcoin protocol (protocol evolution was impacted by applications).

- see "Vitalik Buterin - Ethereum" video on Texas Bitcoin Youtube channel @3:00

2nd generation of blockchains

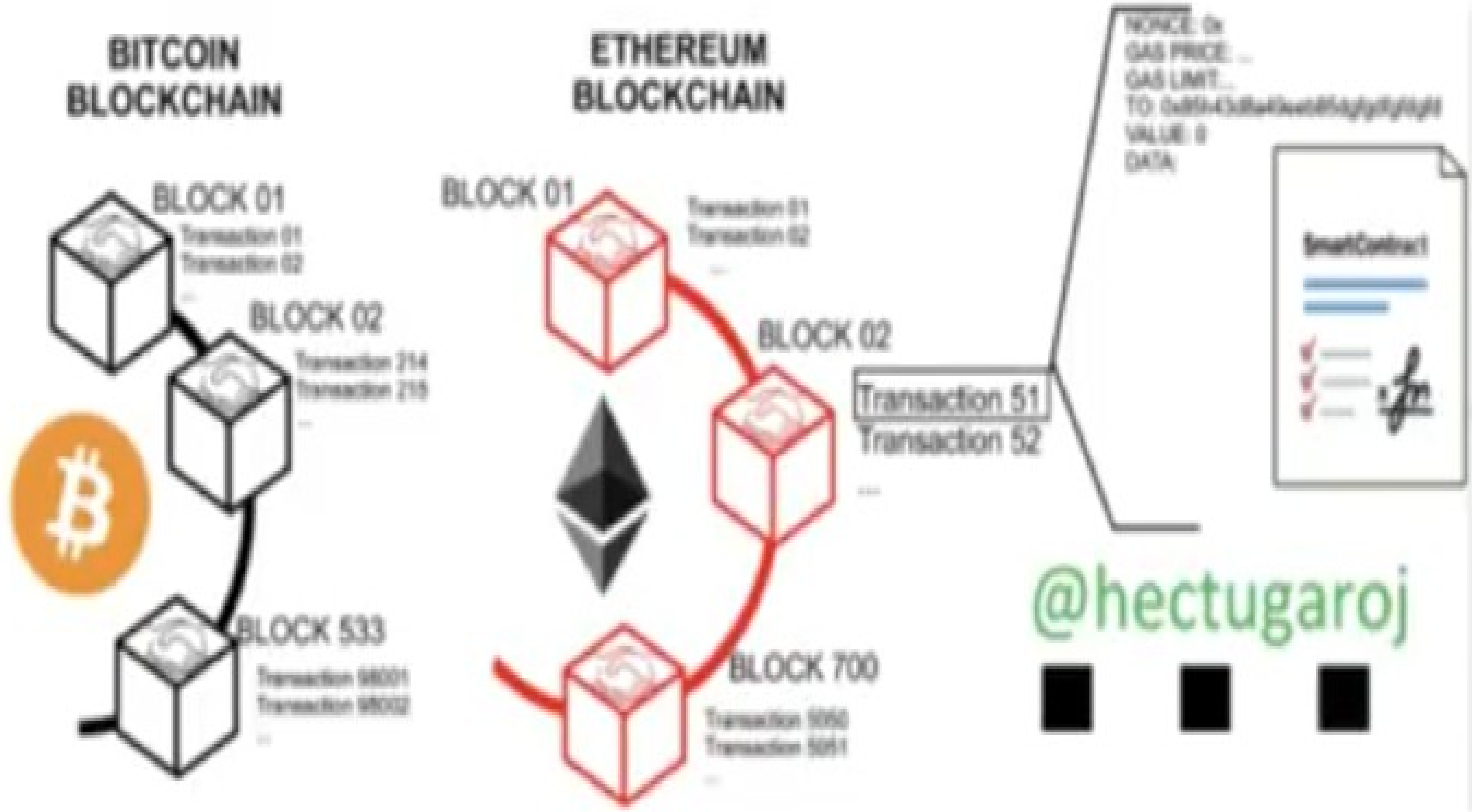
- Vitalik Buterin (former University of Waterloo 2nd year dropout) removes scripting limitations of Bitcoin by adding full programming feature to blockchain & other enhancements
  - See account differences between Bitcoin UTXO & Ethereum, at: https://hackernoon.com/getting-deep-into-ethereum-how-data-is-stored-in-ethereum-e3f669d96033
- SMART CONTRACT redefined = kind of object-oriented class that lives at a blockchain deployed account address & owned by deployer. Contract object functions can be called (via Tx) by invoking the contract address with required parameters.
  - White paper released initially in November 2013.



Smart Contract

Smart Contract concept introduced in 1997 by Nick Szabo

- A protocol that helps execute the terms of a contract
- Challenges of smart contracts:
  - Security
    - Self-enforcement
    - Evidence
      - Observation by parties in privity
      - Verification by adjudicator
  - Mental Transaction Costs
    - Measurement of Value
    - Ex Ante
      - Negotiations – should I agree to this smart contract?
    - Ex Post
      - Determination of damages by adjudicator



BITCOIN BLOCKCHAIN
BLOCK 01
Transaction 01
Transaction 02
BLOCK 02
Transaction 214
Transaction 215
BLOCK 533
ETHEREUM BLOCKCHAIN
BLOCK 01
Transaction 01
Transaction 02
BLOCK 02
Transaction 51
Transaction 52
BLOCK 700
NONCE: 0x
GAS PRICE: ...
GAS LIMIT...
VALUE: 0
DATA
SmartContract
@hectugaroj

The Coin

We are going to create a digital token. Tokens in the Ethereum ecosystem can represent any fungible tradable good: coins, loyalty points, gold certificates, IOUs, in-game items, etc. Since all tokens implement some basic features in a standard way, this also means that your token will be instantly compatible with the Ethereum wallet and any other client or contract that uses the same standards.

MINIMUM VIABLE TOKEN

The standard token contract can be quite complex. But in essence a very basic token boils down to this:

```
pragma solidity >=0.4.22 <0.6.0;

contract MyToken {
    /* This creates an array with all balances */
    mapping (address => uint256) public balanceOf;

    /* Initializes contract with initial supply tokens to the creator of the contract */
    constructor(
        uint256 initialSupply
        ) public {
        balanceOf[msg.sender] = initialSupply;              // Give the creator all initial tokens
    }

    /* Send coins */
    function transfer(address _to, uint256 _value) public returns (bool success) {
        require(balanceOf[msg.sender] >= _value);           // Check if the sender has enough
        require(balanceOf[_to] + _value >= balanceOf[_to]); // Check for overflows
        balanceOf[msg.sender] -= _value;                    // Subtract from the sender
        balanceOf[_to] += _value;                           // Add the same to the recipient
        return true;
    }
}
```

```
contract Voting {

    address owner;

    event AddedCandidate(uint candidateID);

    constructor public {owner = msg.sender;}

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    struct Voter {
        bytes32 uid;            // bytes32 type are basically strings
        uint candidateIDVote;
    }

    struct Candidate {
        bytes32 name;
        bytes32 party;
        bool doesExist; //keep track of candidates
    }
    uint numCandidates;
    uint numVoters;

    mapping (uint => Candidate) candidates;
    mapping (uint => Voter) voters;

    function addCandidate(bytes32 name, bytes32 party) onlyOwner public {
        uint candidateID = numCandidates++;
        candidates[candidateID] = Candidate(name,party,true);
        AddedCandidate(candidateID);                            // EVENT
    }
    function vote(bytes32 uid, uint candidateID) public {
        if (candidates[candidateID].doesExist == true) {
            uint voterID = numVoters++;
            voters[voterID] = Voter(uid,candidateID);
        }
    }
    function totalVotes(uint candidateID) view public returns (uint) {
        uint numOfVotes = 0;
        for (uint i = 0; i < numVoters; i++) {
            if (voters[i].candidateIDVote == candidateID) {
                numOfVotes++;
            }
        }
        return numOfVotes;
    }
    function getNumOfCandidates() public view returns(uint) {
        return numCandidates;
    }
    function getNumOfVoters() public view returns(uint) {
        return numVoters;
    }
    function getCandidate(uint candidateID) public view returns (uint,bytes32, bytes32)
        return (candidateID,candidates[candidateID].name,candidates[candidateID].party)
    }
}

```

Normal text file | length : 1,797 | lines :

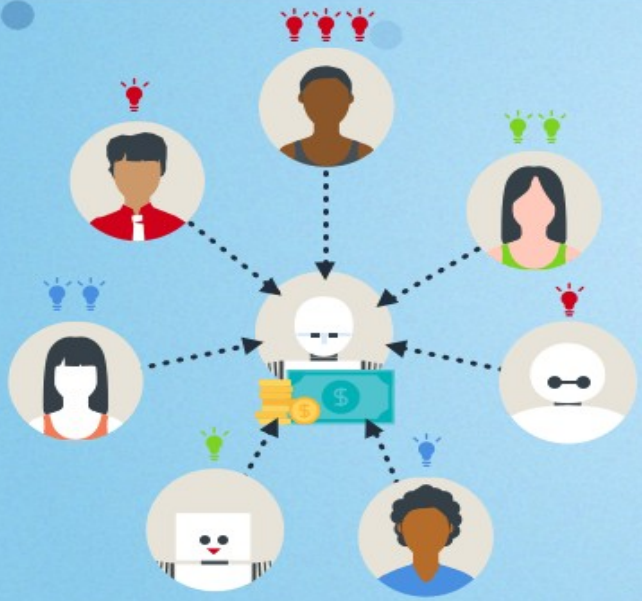
Smart Contracts

- Any routine task done by a middle man can be automated (disintermediated)
- Hard to modify since is recorded on the blockchain
- Code is law (disputed by EOS blockchain's "social arbitration")
  - Law is written in the smart contract
  - Smart contract execution enforces the law
    - Issue if smart contract has bugs since blockchain is immutable

Cypherpunk vision with Ethereum Smart Contracts

https://ethereum.org

JOIN THE COMMUNITY **DONATE**



Create a democratic autonomous organization

Now that you have developed your idea and secured funds, what's next? You have to hire managers, find a trustworthy CFO to handle the accounts, run board meetings and do a bunch of paperwork.

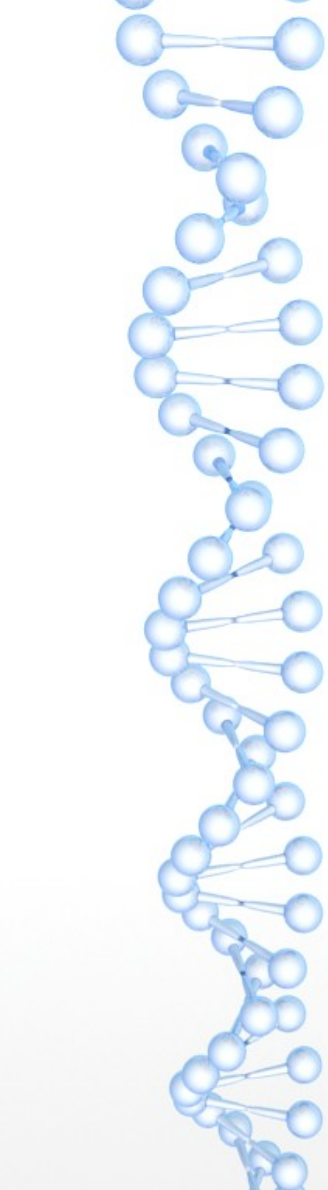
Or you can simply leave all that to an Ethereum contract. It will collect proposals from your backers and submit them through a completely transparent voting process.

One of the many advantages of having a robot run your organization is that it is immune to any outside influence as it's guaranteed to execute only what it was programmed to. And because the Ethereum network is decentralized, you'll be able to provide services with a 100% uptime guarantee.

Start your organization

YOU CAN BUILD:

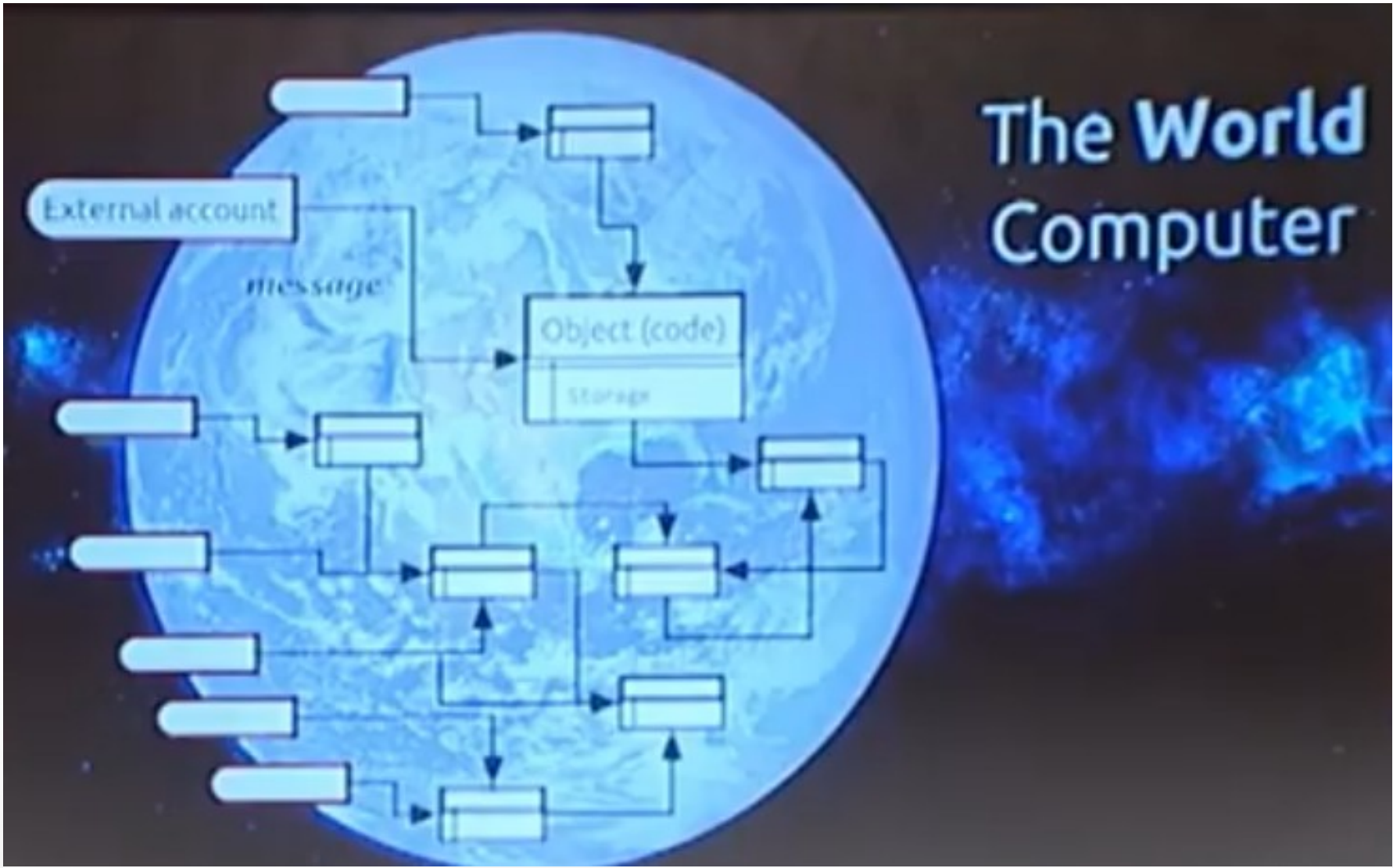
- A virtual organization where members vote on issues
- A transparent association based on shareholder voting
- Your own country with an unchangeable constitution
- A better delegative democracy

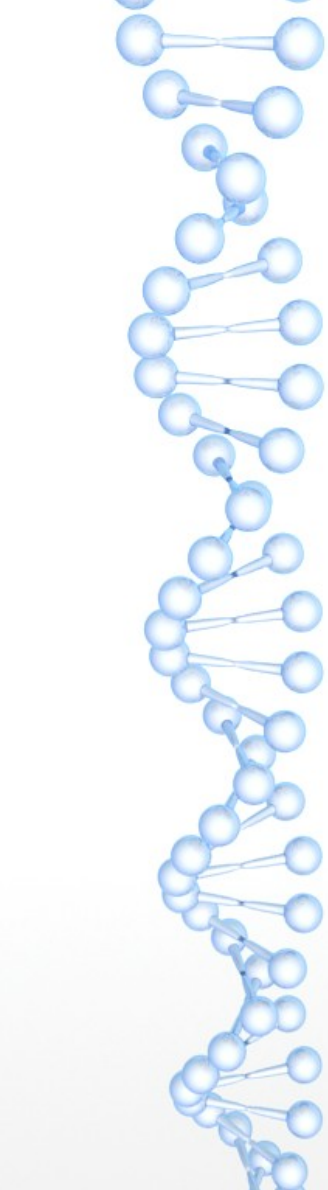


Ethereum World Computer

Ethereum is like a "World Computer", since deployed smart contracts are stored in the blockchain by all nodes and are potential executable by all (via "messages" from clients or other smart contracts).

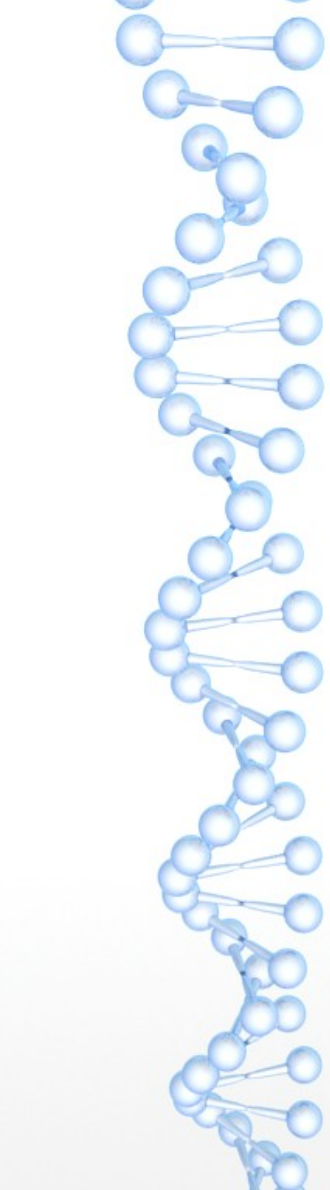
- Smart contracts change only the data on the blockchain
- Oracles are used to provide reliable data to the blockchain


The World Computer
External account
message
Object (code)
Storage



Blockchain V2 = Transfer of value blockchains

- Blockchain evolved from a payment network to a value transfer network.
- Any application with a transactional nature can be implemented or ported as a GUI interface to a collection of smart contracts on a blockchain offering the decentralization & trust features.
- Applications examples: voting, escrow, identity, crowdfunding, IoT, assets tracking (land & car registrations), real estate, auctions, supply chain tracking (diamonds, "farm to table"), insurance, proof of authenticity, etc.



Blockchain invariants

- Write in blocks of Tx to be able to crypto secure it & minimize number of links in the chain. The block size is an open debated issue.
- Tamper Resistance due to heavy cryptography use at Tx & block level
- Distributed Governed Write via consensus protocol
- Immutability (permanent write) due to all of the above

Blockchain V1/V2 variants

Deviations from Bitcoin's initial blockchain design:

- Advanced SMART CONTRACTS. Ability to store & run programs on blockchain written in modern programming languages like Solidity, Viper, C#, Java, C++, Go, JS, Haskell, etc.).
- Identity (no anonymous users).
- Private / Closed / Enterprise / Permission networks. Classical IT security (PKI) is used to allow authorized blockchain access / use
  - Allowed users can see / audit the ledger
- Consortium Enterprise blockchains
  - Encrypted ledger. Discriminate who can see the ledger
- Different Trusted Network Write (consensus) protocols

Scaling Tx/sec by faster consensus types

- PoW = hard to do, easy to verify
- Proof Stake (PoS): miners put assets at stake. Instead of using expensive hardware to solve the computation problem, PoS approach has no mining or requirement for work. It uses a pre-determined algorithm to identity the node that generates a new block.
- Proof of Authority: trusted nodes validate blocks to speed up consensus. Put reputation at stake.
- Proof of Activity (PoA): This is a combination of Proof of Work and Proof Stake. The blocks are created by the miner without any transactions using PoW. PoS is used to complete the block with the transactions.
- Proof of Burn (PoB): Instead of spending money on expensive hardware, you send the money/valuables to location that cannot be retrieved, basically burning the currency.
- Proof of Capacity (PoC): Depends on the hard disk space. The more capacity you have the more chances you will have to create the block.
- Proof of Elapsed time (PoET): Created by Intel. Like Proof of Work but consumes much less electricity. Instead of participants solving “cryptographic puzzles”, the algorithm uses a Trusted Execution Environment (TEE) to ensure blocks get produced in random lottery order, but without requiring any work.
- Practical Byzantine Fault Tolerance (PBFT): used by IBM Hyperledger consortium private blockchain & Quorum
- RAFT distributed consensus protocol: used by J. P. Morgan Quorum (Ethereum based) blockchain

(partial credit to Microsoft documentation)

Types of Blockchains

- PUBLIC - Anybody can join this network.
- PRIVATE – built locally to allow document sharing at organization level. Access to the network is controlled.
- CONSORTIUM or Federated Blockchains - More than one party controls the access to the network. Built to share documents & process automation across businesses.

Business Blockchains

- Common data layer & logic to facilitate trusted data exchange between businesses
- No currency usually, but fair play incentive is driven by identity, reputation & liability
- Solve Tx scalability & privacy
- Private (permission) blockchains secured with businesses IT PKI (Public Key Infrastructure) security infrastructure
- Examples:
  - Corda (used for privacy & finality)
  - Microsoft's Confidential Consortium (CoCo) blockchain framework
  - J.P. Morgan Quorum Ethereum blockchain
  - IBM Hyperledger open source blockchain
  - AWS Blockchain Template for Ethereum & Hyperledger

Blockchain platforms

- Andreas Antonopoulos metaphors paraphrased:
  - Blockchains are building the DIGITAL PYRAMIDS of our time writing in stone the digital truth
  - Embrace, instead of fighting them since:
    - Blockchains are in the beginning state of INFRASTRUCTURE INVERSION similar with other inventions like: cars, electricity & Internet when the new technology adversarially emerges on top of the old, becoming the future foundation for the old one.
    - Blockchain is a DISRUPTIVE technology that will enable a permissionless TSUNAMI of INNOVATION.

Takeaway summary

- Blockchain advantages:
  - is a specialized append-only, decentralized, transactional, PERMANENT, TRUSTED, fault-tolerant & shared DB, with:
    - built-in cryptographic tamper resistant security & trust
      - Chain content is fingerprinted on each block
    - multi-party consensus WRITE feature
    - and built-in incentives
      - towards correct consensus data write (TRUST MACHINE)
  - Integration platform for heterogeneous data sources
    - Enabling process automation across businesses
- Disadvantages:
  - Low transactions per second (due to consensus)
  - Not easy to input reliable data into blockchain (need for decentralized auditable oracles)
  - Not easy to query data in DB style
  - Need $$$ (or crypto coins) to use it, sometimes.

BLOCKCHAIN
Blockchain
Design + Blockchain
BLOCKCHAIN FOR GOOD
IBM
MIX
RIOT
GBBC