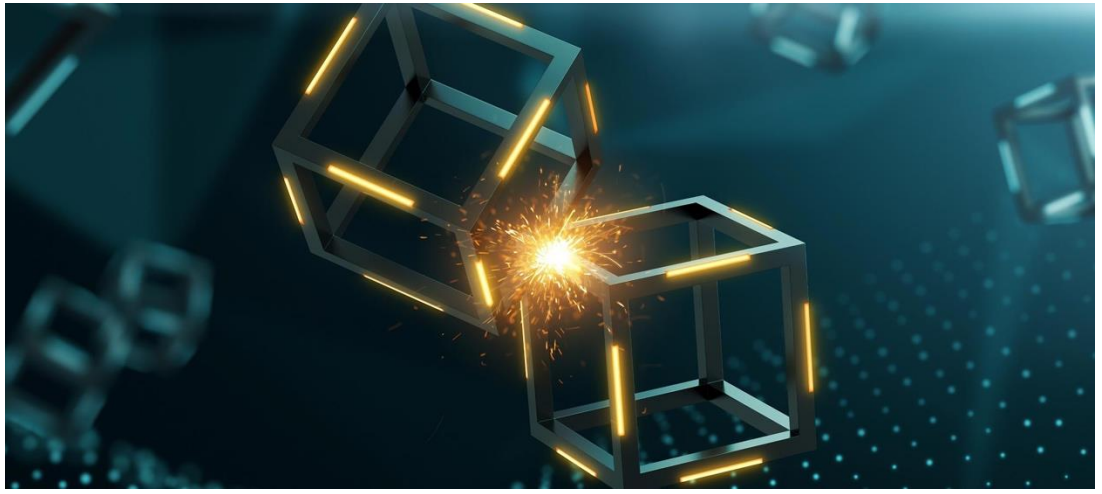
Blockchain Technologies - COMP0163

Week 1: Bitcoin blockchain: how it works



Teaching team

Silvia Bartolucci

s.bartolucci@ucl.ac.uk

Jiahua Xu

jiahua.xu@ucl.ac.uk

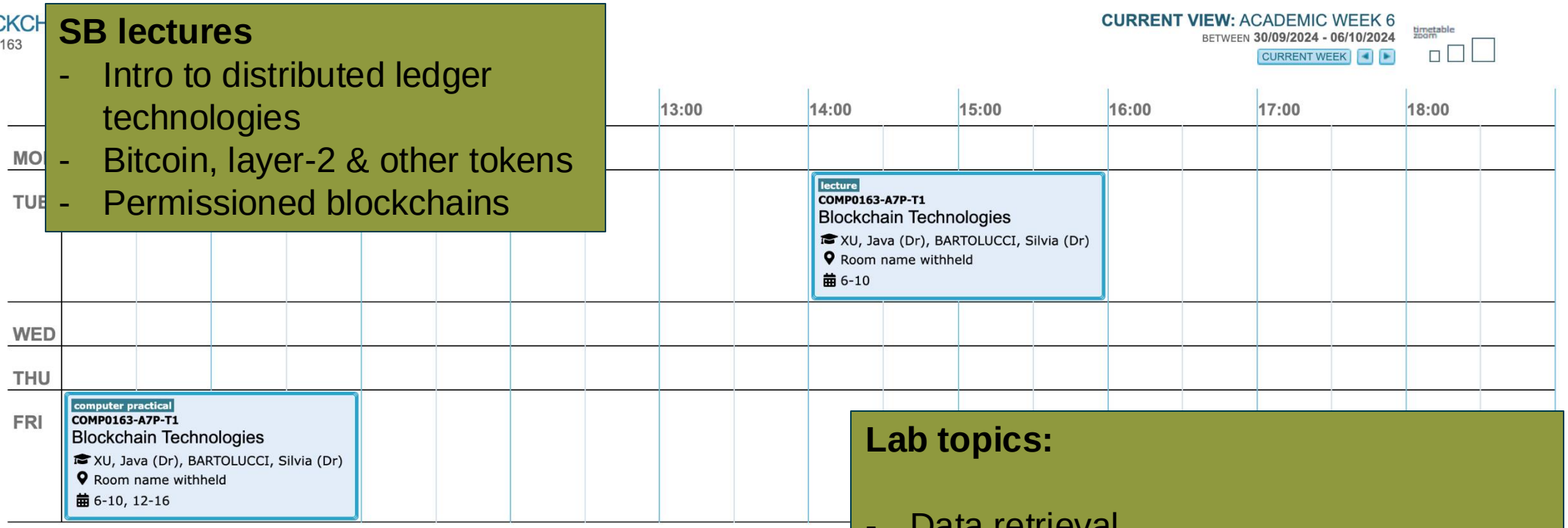
TA: Yichen Luo

yichen.luo.22@ucl.ac.uk

Timetable

SB lectures

- Intro to distributed ledger technologies
- Bitcoin, layer-2 & other tokens
- Permissioned blockchains



Lab topics:

- Data retrieval
- BTC transactions
- DAML & Permissioned blockchains
- Token smart contract
- Defi smart Contract

Support & office hours

s.bartolucci@ucl.ac.uk

Bookings: You can **book** a 20-mins slot using this link: https://calendly.com/sbartolucci/20min

Forum

- Use the forum on moodle to post your questions & doubts about the content of the lectures or admin issues.
- You can post **anonymous** queries.
- If you know the answer, contribute to the discussion and help your colleagues.

- **Pre-reading and exercise**

Read abstract and introduction from the Bitcoin Whitepaper

and identify the *core components* as well as the main features of the Bitcoin blockchain .

Bitcoin: A Peer-to-Peer Electronic Cash System

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 avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

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

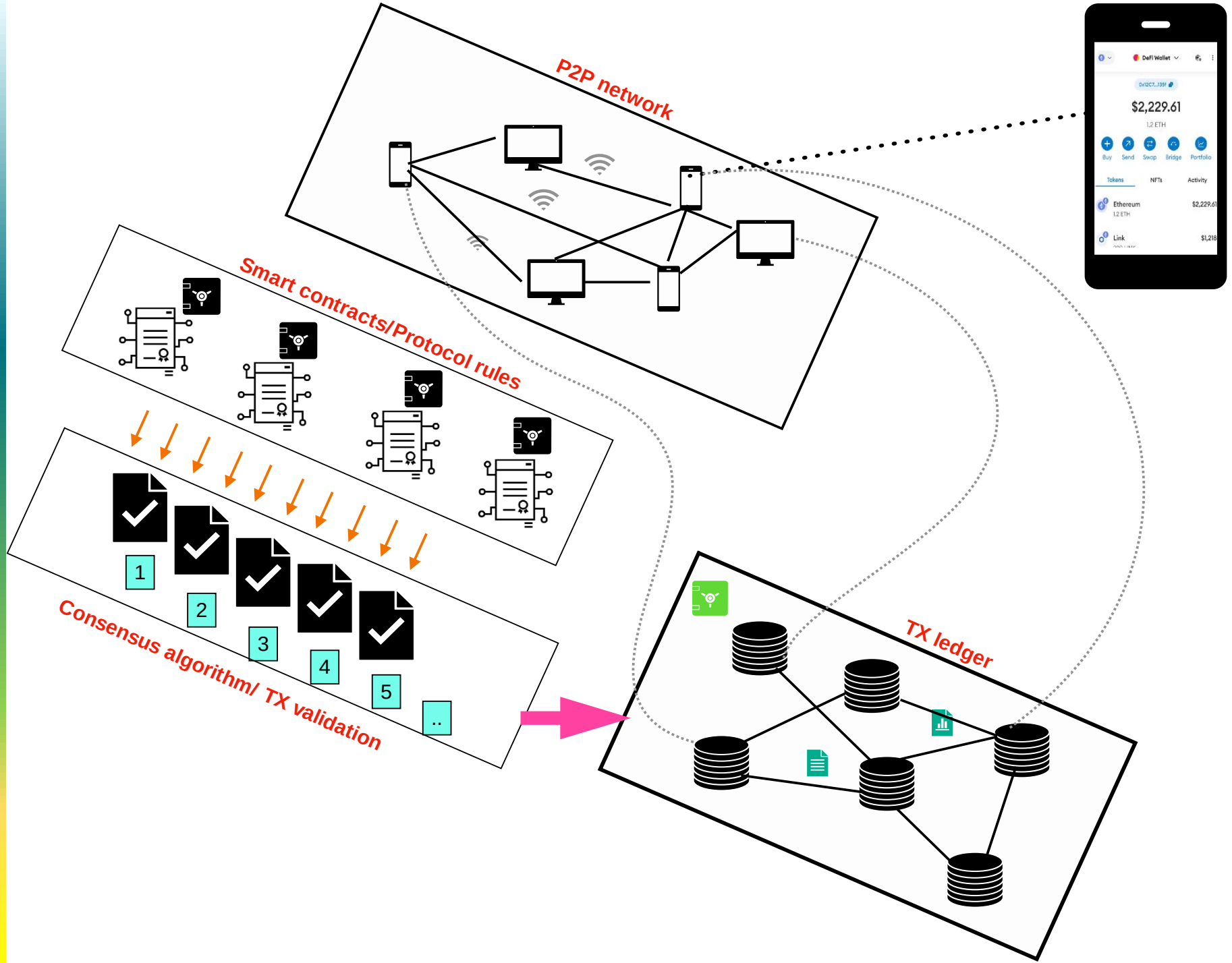
Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

Core components: P2P, Chain (Blockchain), Proof-of-Work (Consensus), Cryptography.

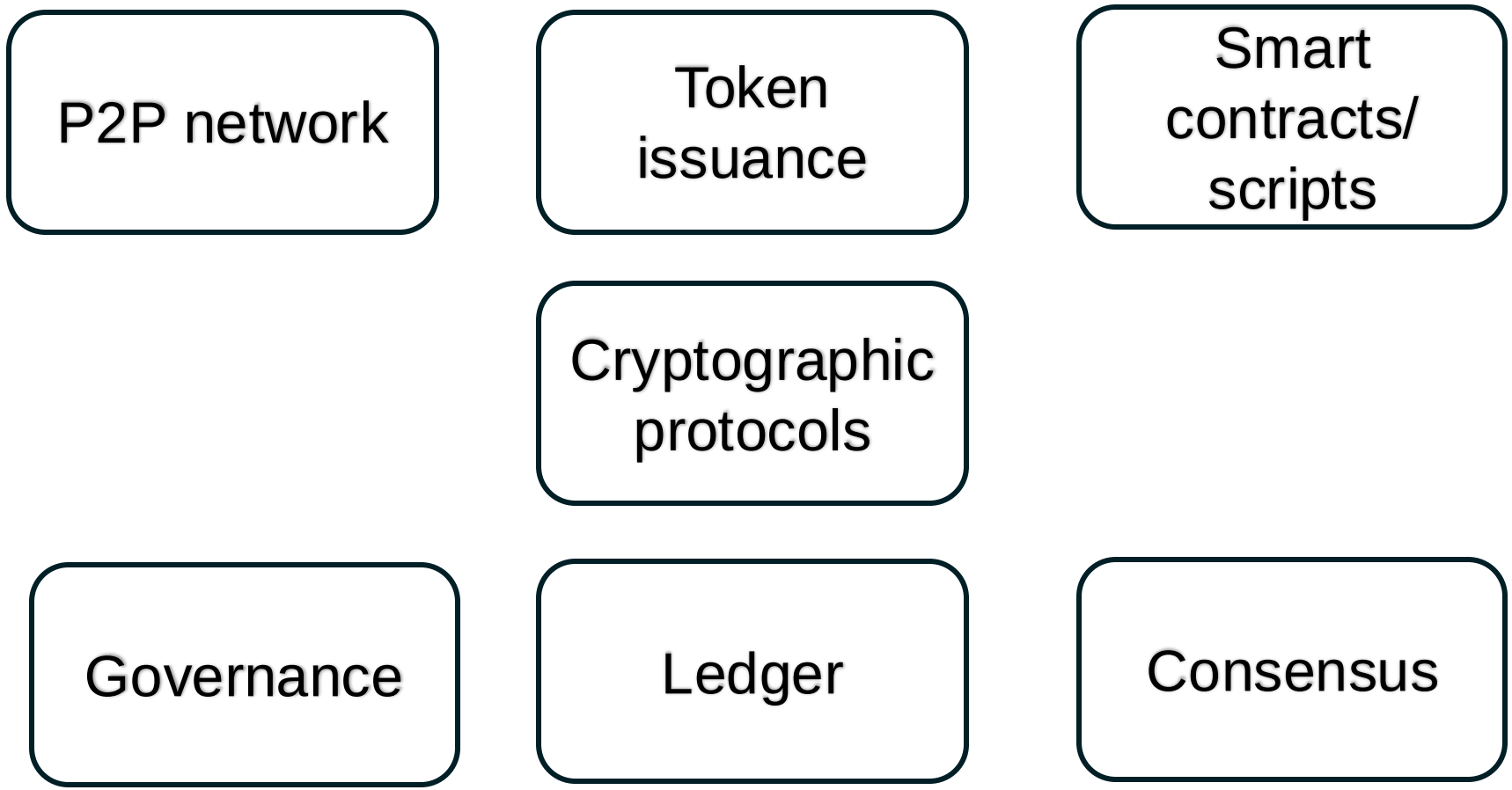
Features: Decentralisation (no trusted party), Prevents Double-Spending, Transactions Immutability.

PROTOCOL'S COMPONENTS



Core protocol components

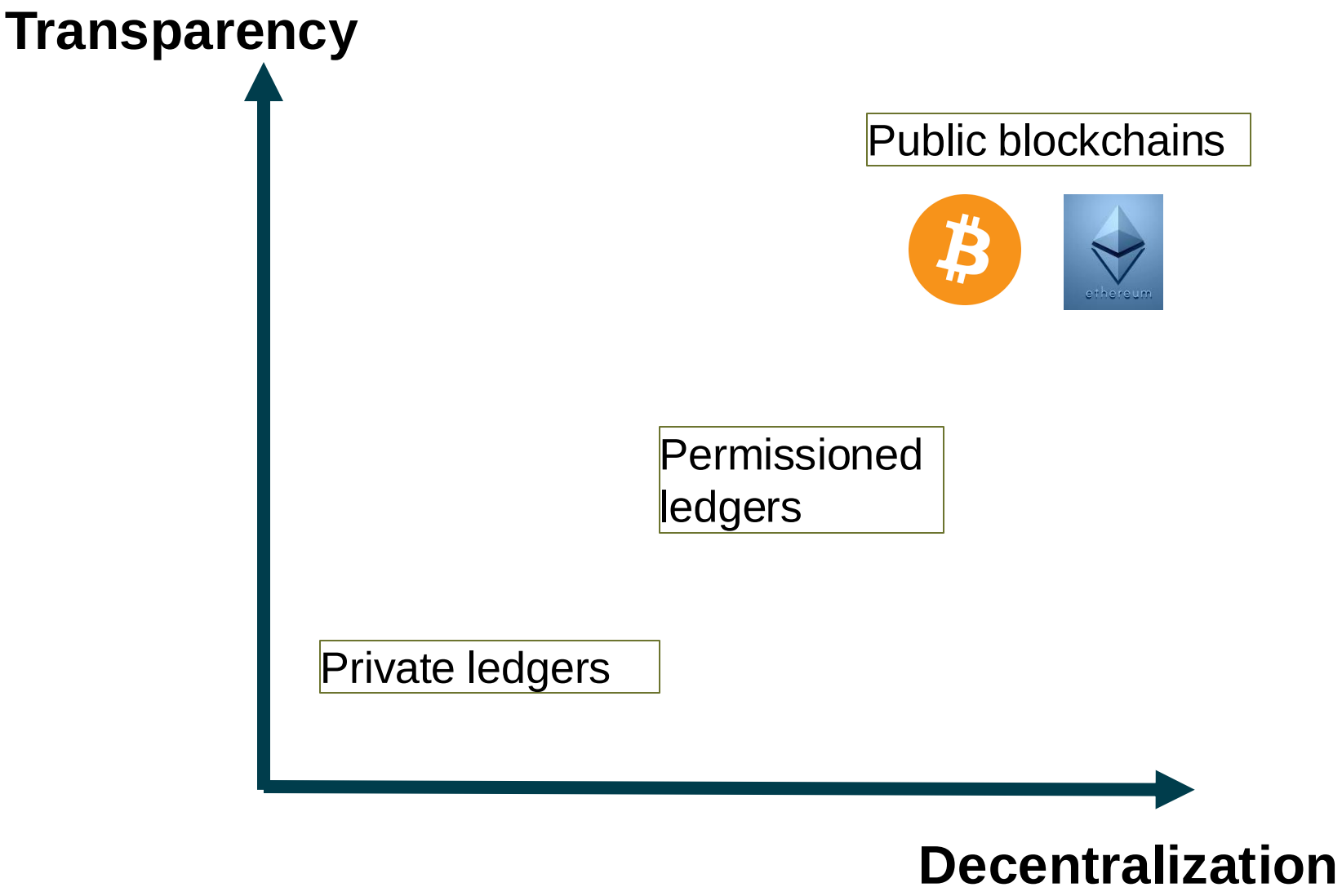
The Bitcoin protocol can be decomposed in fundamental ingredients or sub-parts that interact together to create the Bitcoin platform. Those ingredients are the baseline of **all** crypto-platforms.



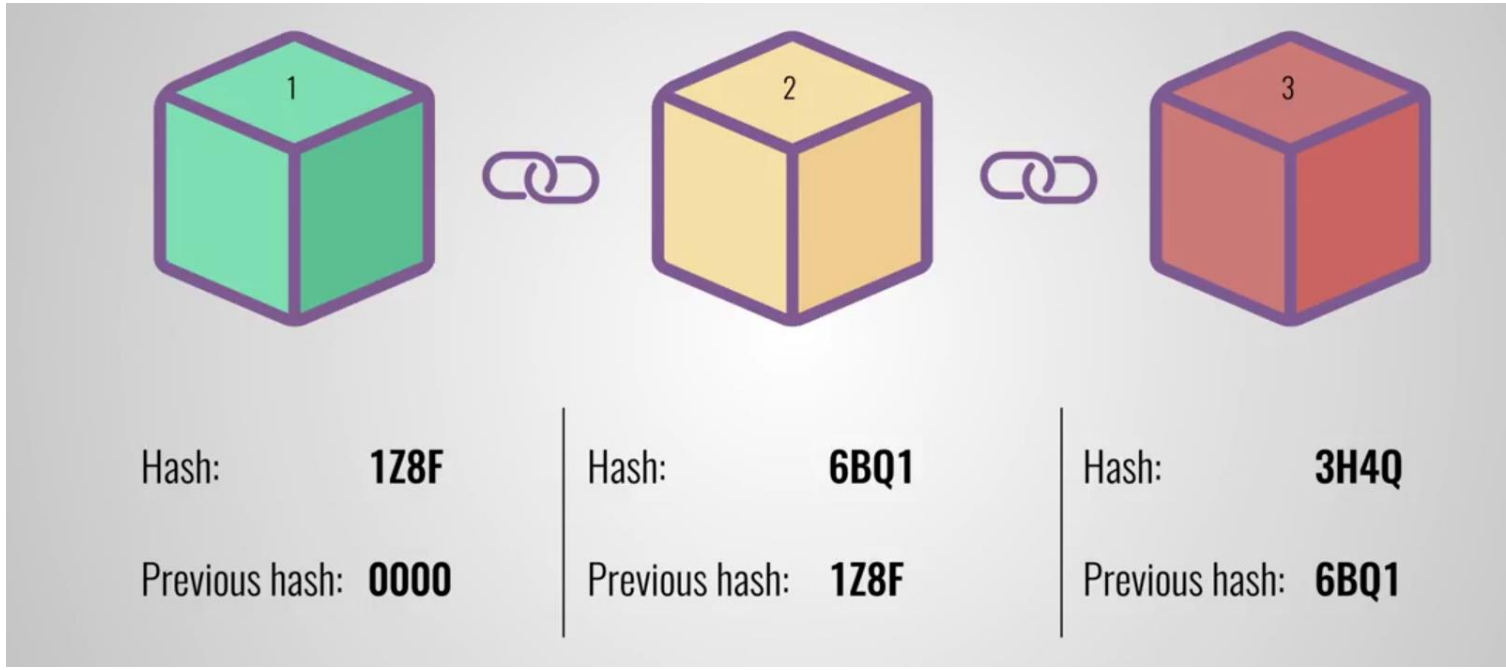
Navigating challenges

Adoption/ Investments	What drives the adoption of the technology and of the investments?
Economic Incentives design	How should rewards and rules be designed to ensure participation in the protocol?
Asset Valuation/ Market dynamics	What determines the value of digital assets?
Financial stability	What are possible spillovers effect on the traditional financial system?
Regulation	How to assess the technology? What safeguards should be in place for users and investors?
Governance	How users contribute to shaping the protocol's rules?
Financial setup	How should tokens be issued? How should parameters be set?

Type of platforms



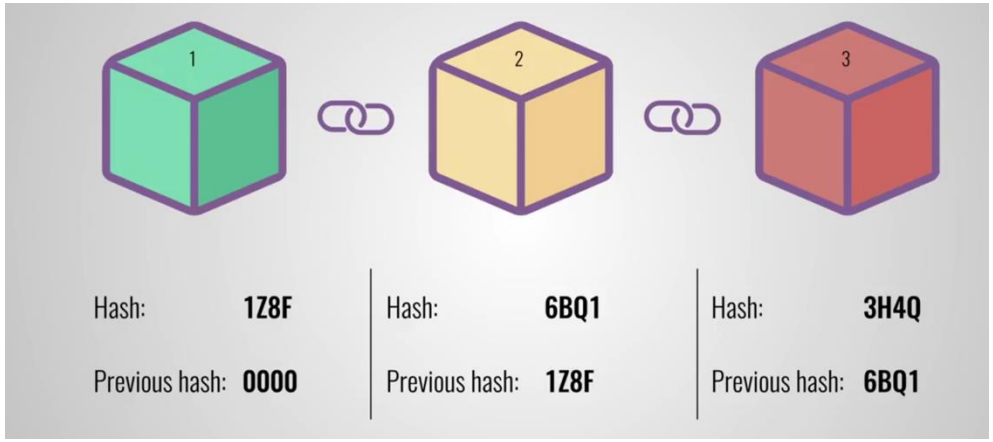
The blockchain



https://andersbrownworth.com/blockchain/

The blockchain

The blockchain data structure is an ordered, back-linked list of blocks of transactions.



- Each block is identified by a hash, generated using the SHA256 cryptographic hash algorithm on the header of the block.

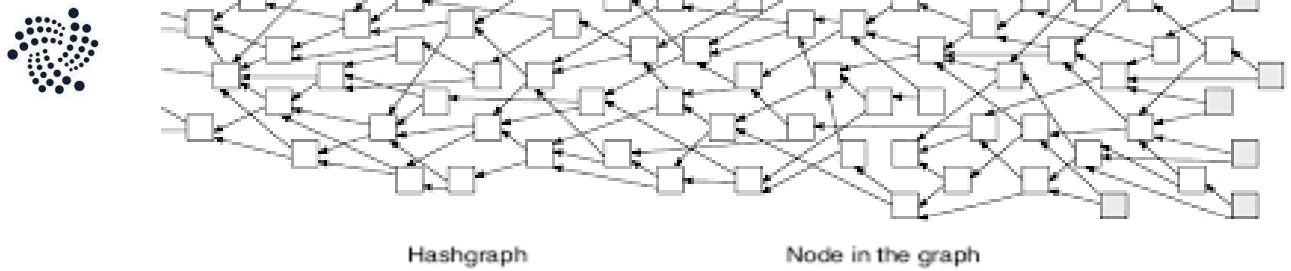
- Each block also references a previous block, known as the *parent* block, through the "previous block hash" field in the block header.

- Each block contains the hash of its parent inside its own header. The sequence of hashes linking each block to its parent creates a chain going back all the way to the first block ever created, known as the ***genesis block***.

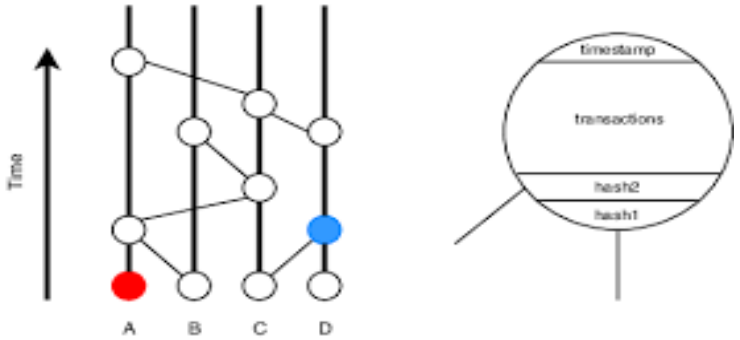
Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. " O'Reilly Media, Inc.".

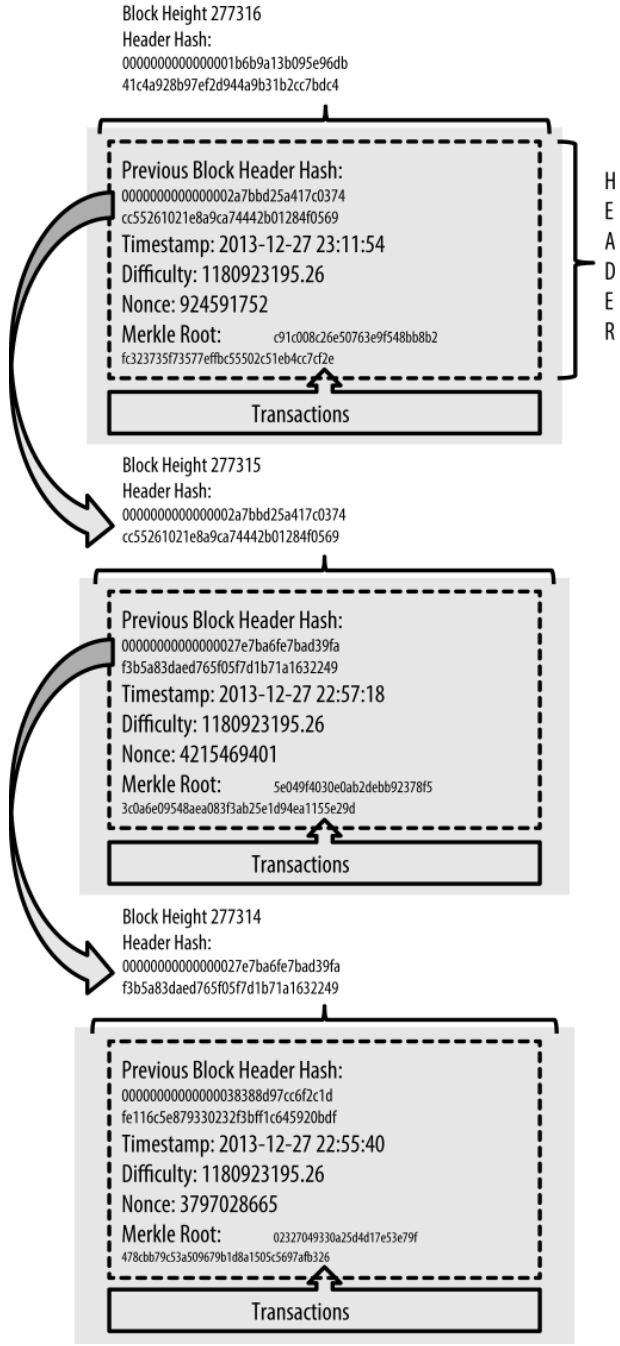
Ledger

Blockchain: It is a **shared distributed database** where transactions are recorded and cannot be altered retroactively. Transactions are grouped in block data structures linked togther *cryptographically* to form a chain**.** The blockchain is a specific realisation of a shared distributed database also generically defined as **distributed ledger**.



Hashgraph Node in the graph





The block structure

Table 7-1. The structure of a block

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1-9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

Table 7-2. The structure of the block header

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The proof-of-work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the proof-of-work algorithm

The primary identifier of a block is its cryptographic hash, a digital fingerprint, made by hashing the block header twice through the SHA256 algorithm.

A second way to identify a block is by its position in the blockchain, called the *block height*.

Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. " O'Reilly Media, Inc.".

Transactions

Hash	776587fc1059eab053e9f7f00d543d5e5a700b64b7d280078ffdfe7...				2020-03-23 11:09
	3B2j7GZwHwKiUe87VhcCBU5A4ArKGfm4Mh	0.00065383 BTC	→	3CkLTx9cVTY8NzEXCazzREjadLGnDv2B5R	0.03600000 BTC
	346tE4gJRVVufNKMQPY68wD6ELk6Fzhugp	0.03400000 BTC			
	36BM6ZszawJcPB3nmTjmGQmmQ7Q4GSPj53	0.00264930 BTC			
Fee	0.00130313 BTC (232.702 sat/B - 103.259 sat/WU - 560 bytes)				0.03600000 BTC 1 Confirmations

Hash	c0032cb46b82f91dc2ccd304395421c73be6625b8103f16ee8e806...				2020-03-23 11:10
	3J9S329rf5cDHZ49q6FyhhxUoVSKK8Ja1B	0.14663641 BTC	→	3ENmL8mUYxUinKwdwbqJ1XPf2T53rK3PsK	0.10862450 BTC
				1BFfQFBnKL8jnAwCWU3cEu43tg4L6s9JLr	0.03721191 BTC
Fee	0.00080000 BTC (196.560 sat/B - 92.700 sat/WU - 407 bytes)				0.14583641 BTC 1 Confirmations

The genesis block

```
GetHash()      = 0x000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
hashMerkleRoot = 0x4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
txNew.vin[0].scriptSig     = 486604799 4
0x736B6E616220726F662074756F6C69616220646E6F63657320666F206B6E697262206E6F20726F6C6C65636E6168432039303032 2F6E614A2F33302073656D695420656854
txNew.vout[0].nValue       = 5000000000
txNew.vout[0].scriptPubKey = 0x5F1DF16B2B704C8A578D0BBAF74D385CDE12C11EE50455F3C438EF4C3FBCF649B6DE611FEAE06279A60939E028A8D65C10B73071A6F16719274855FEB0FD8A6704
OP_CHECKSIG
block.nVersion = 1
block.nTime    = 1231006505
block.nBits    = 0x1d00ffff
block.nNonce   = 2083236893

CBlock(hash=000000000019d6, ver=1, hashPrevBlock=0000000000000, hashMerkleRoot=4a5e1e, nTime=1231006505, nBits=1d00ffff, nNonce=2083236893, vtx=1)
  CTransaction(hash=4a5e1e, ver=1, vin.size=1, vout.size=1, nLockTime=0)
    CTxIn(COutPoint(000000, -1), coinbase
04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6c6f72206f6e206272696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73)
    CTxOut(nValue=50.00000000, scriptPubKey=0x5F1DF16B2B704C8A578D0B)
  vMerkleTree: 4a5e1e
```

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

The genesis block


THE TIMES

Saturday January 3 2009 timesonline.co.uk No 69523 £1.50

Eat Out from £5

More than 900 great restaurants, including four Gordon Ramsay favourites from £15

Start collecting tokens today Pullout Inside

Israel prepares to send tanks and troops into Gaza

Chancellor on brink of second bailout for banks

Billions may be needed as lending squeeze tightens

Michael Sheen Frost, Nixon and me

Working mums So that's how she does it

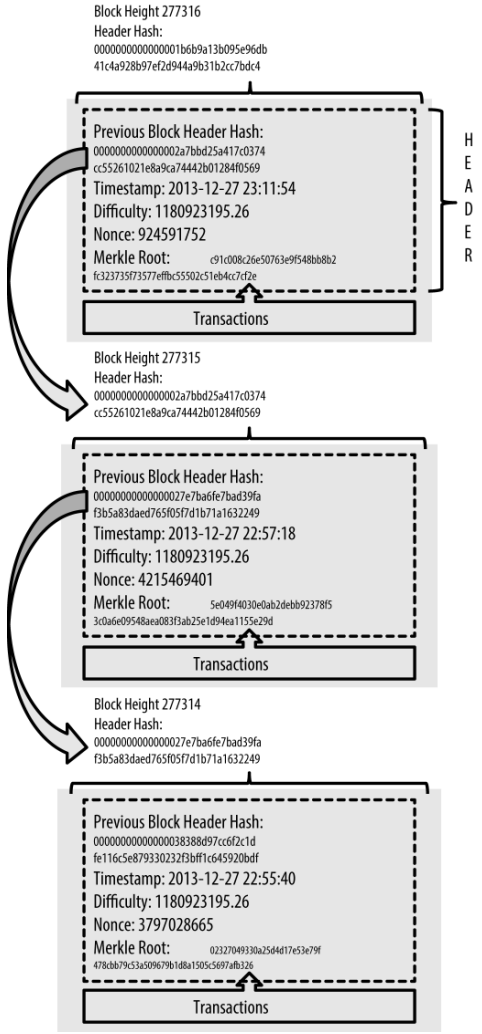
Detox in style The best spas on the planet

Giant Killing? Guide to the FA Cup Third Round

The coinbase parameter (seen above in hex) contains, along with the normal data, the following text:

The Times 03/Jan/2009 Chancellor on brink of second bailout for banks. This was probably intended as proof that the block was created on or after January 3, 2009, as well as a comment on the instability caused by fractional-reserve banking.

The block structure

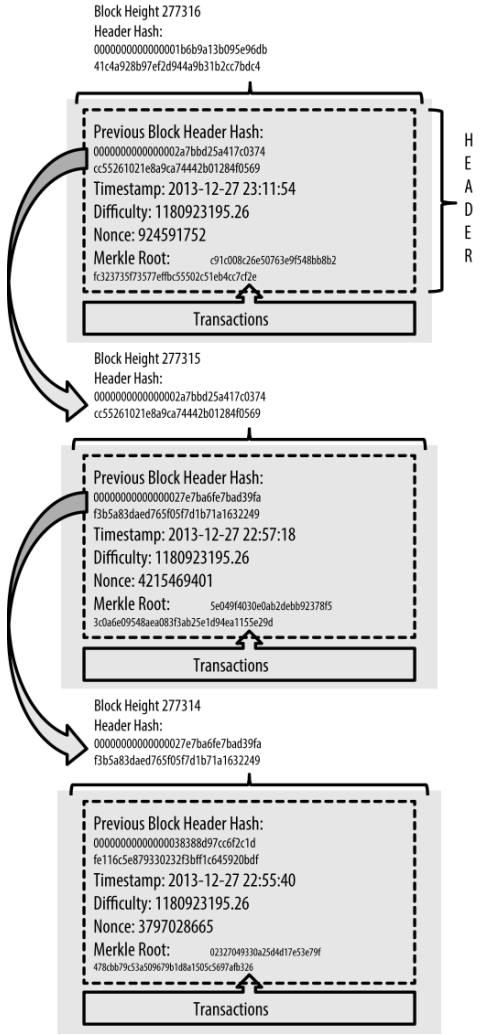


Bitcoin full nodes maintain a local copy of the blockchain, starting at the genesis block. The local copy of the blockchain is constantly updated as new blocks are found and used to extend the chain.

As a node receives incoming blocks from the network, it will validate these blocks and then link them to the existing blockchain. To establish a link, a node will examine the incoming block header and look for the “previous block hash.”

Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. ” O’Reilly Media, Inc.”.

The block structure



- Why is the blockchain using precisely the hash function to link the blocks?
- What is the most relevant feature of a hash functions that is important when building a blockchain database?

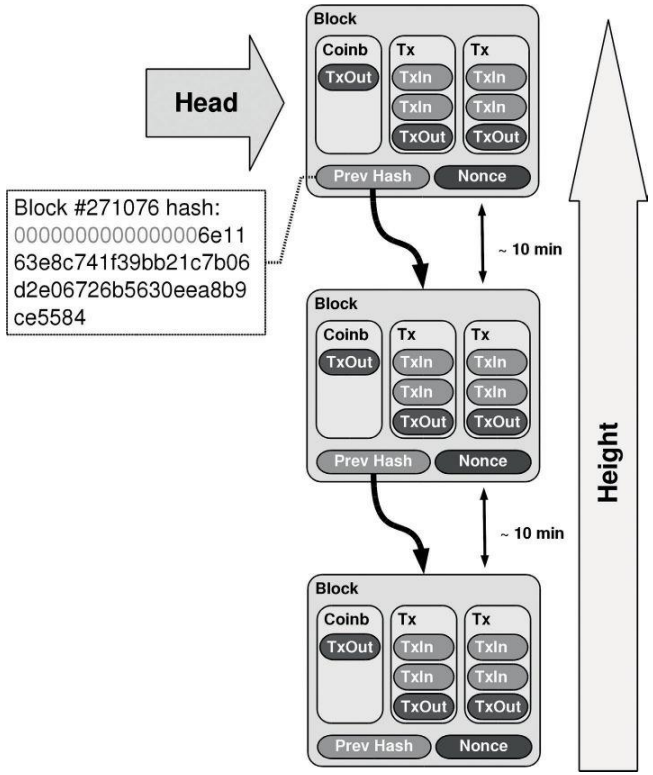
Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. ” O’Reilly Media, Inc.”.

Hash functions

Hash functions are auxiliary functions, they are used for:

- Digital signatures schemes
- Key derivation
- Random Number Generators

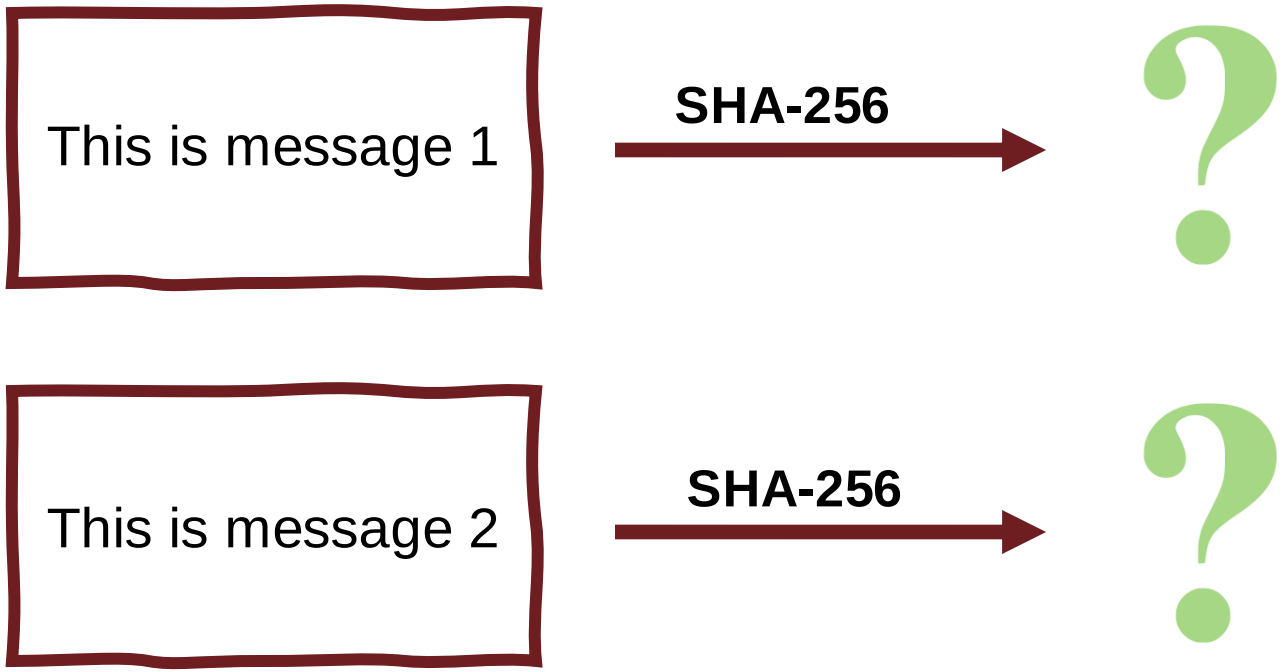
In the blockchain settings, they are the key function used to create the “chain of blocks”



Franco, P. (2014). Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons.

Experiment 1

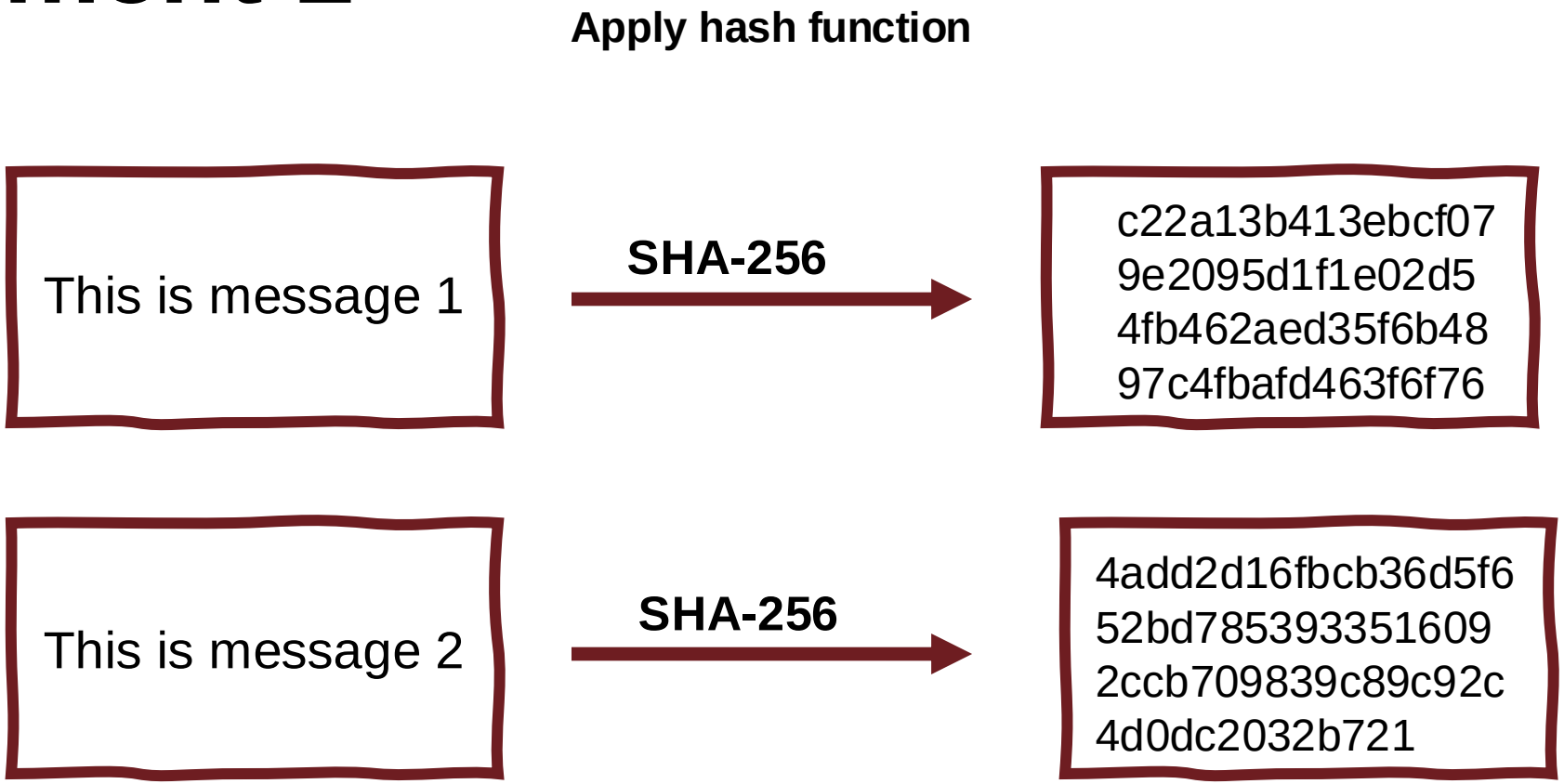
Apply hash function



- Python: import **hashlib**

  hashlib.sha256(b"This is message 2").hexdigest()
- **Go to https://andersbrownworth.com/blockchain/hash**

Experiment 1



- Python: import **hashlib**

  hashlib.sha256(b"This is message 2").hexdigest()
- **Go to https://andersbrownworth.com/blockchain/hash**

Experiment 1

Apply hash function

This is message 1

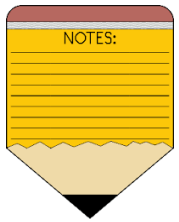
SHA-256

c22a13b413ebcf07
9e2095d1f1e02d5
4fb462aed35f6b48
97c4fbafd463f6f76

This is message 2

SHA-256

4add2d16fbcb36d5f6
52bd785393351609
2ccb709839c89c92c
4d0dc2032b721

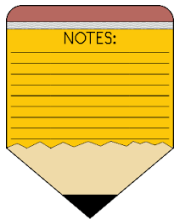


Experiment 1

Apply hash function

This is message 1 → **SHA-256** → c22a13b413ebcf07 9e2095d1f1e02d5 4fb462aed35f6b48 97c4fbafd463f6f76

This is message 2 → **SHA-256** → 4add2d16fbcb36d5f6 52bd785393351609 2ccb709839c89c92c 4d0dc2032b721



Very similar messages have very different hash!

A good hash function behaves like a random mapping

from the input value to the hash value.

- *You can try with a few more examples…*

Experiment 2

Apply hash function

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.

SHA-256 →

?

Bitcoin.

SHA-256 →

?

- Python: import **hashlib**

  hashlib.sha256(b"This is message 2").hexdigest()

- **Go to https://andersbrownworth.com/blockchain/hash**

Experiment 2

Apply hash function

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.

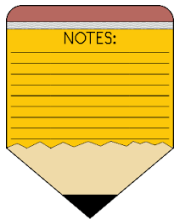
SHA-256

0fab2b7613d8fba09
716aba721eeaf25f0
2612737ba5f47e5aa
a042ecb159211

Bitcoin.

SHA-256

a9adf3c04d168153b
296083f05015f587d
7df6e0b85305b6c7b
eb2a69e3f4e75



Experiment 2

Apply hash function

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.

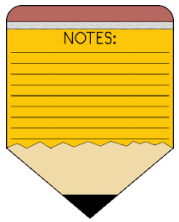
SHA-256

0fab2b7613d8fba09
716aba721eeaf25f0
2612737ba5f47e5aa
a042ecb159211

Bitcoin.

SHA-256

a9adf3c04d168153b
296083f05015f587d
7df6e0b85305b6c7b
eb2a69e3f4e75



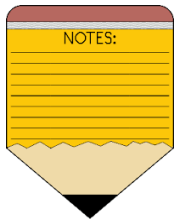
The output of a hash function is of **fixed length** and independent of the input length. Practical hash functions have output lengths between 128–512 bits. SHA-256 = 256 bit.

Experiment 3

Can you reverse it?

a0dc65ffca799873cbea
0ac274015b9526505d
aaaed385155425f7337
704883e

(SHA-256)^-1

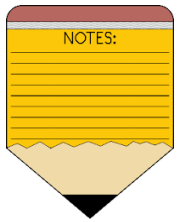


Experiment 3

Can you reverse it?

a0dc65ffca799873cbea
0ac274015b9526505d
aaaed385155425f7337
704883e

(SHA-256)^-1



Given a hash output, called “digest”, it is computationally infeasible to retrieve the input.

Properties of hash functions

Hash functions compute a *digest* of a message which is a short, fixed-length bit-string. For a particular message, the message digest, or *hash value*, can be seen as the fingerprint of a message, i.e., a unique representation of a message.

Properties of Hash Functions

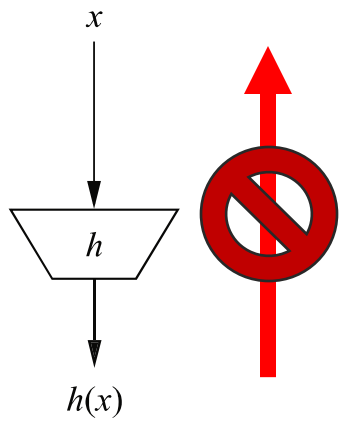
1. **Arbitrary message size** $h(x)$ can be applied to messages x of any size.
2. **Fixed output length** $h(x)$ produces a hash value z of fixed length.
3. **Efficiency** $h(x)$ is relatively easy to compute.
4. **Preimage resistance** For a given output z, it is impossible to find any input x such that $h(x) = z$, i.e, $h(x)$ is one-way.
5. **Second preimage resistance** Given x_1, and thus $h(x_1)$, it is computationally infeasible to find any x_2 such that $h(x_1) = h(x_2)$.
6. **Collision resistance** It is computationally infeasible to find any pairs $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$.

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Properties of hash functions

4. **Preimage resistance** or **one-wayness**: given a fingerprint, we cannot derive a matching message.

Very important feature when hash functions are used to derive a key!



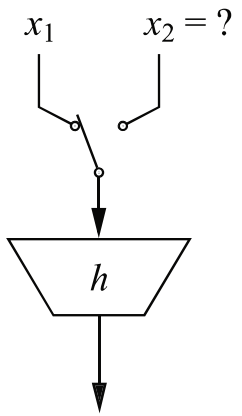
preimage resistance

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

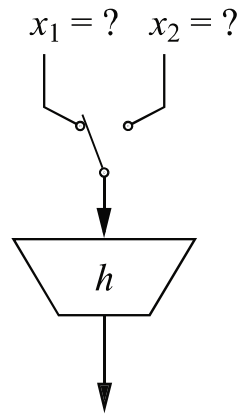
Properties of hash functions

5. **Second preimage resistance or weak collision resistance**: preimage resistance means that it should be computationally infeasible to create two different messages with the same hash values. We can differentiate two types of collision: weak and strong.

x_1 is given and the attacker tries to find x_2



second preimage resistance



collision resistance

An attacker is free to choose both x_1 and x_2.

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Properties of hash functions

We show now how Oscar could turn his ability to find collisions (modifying two messages) into an attack. He starts with two messages, for instance:

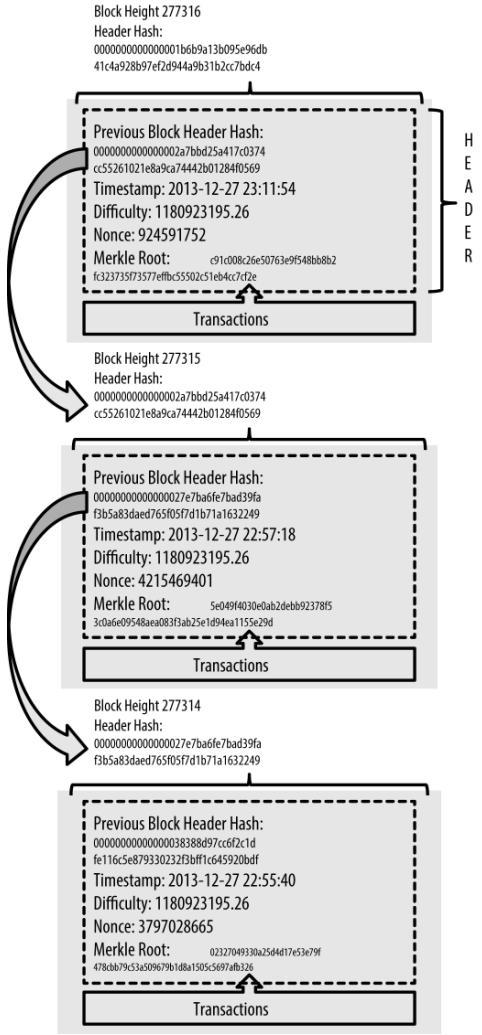
x_1 = `Transfer $10 into Oscar's account`
x_2 = `Transfer $10,000 into Oscar's account`

He now alters x_1 and x_2 at "nonvisible" locations, e.g., he replaces spaces by tabs, adds spaces, etc. The meaning of the message is the same (e.g. for a bank) but the hash changes.

Oscar tries until the condition $h(x_1) = h(x_2)$. Note that if an attacker has e.g., 64 locations that he can alter or not, this yields 2^{64} versions of the same message with 2^{64} different hash values.

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

The block structure



- Why is the blockchain using precisely the hash function to link the blocks?
- What is the most relevant feature of a hash functions that is important when building a blockchain database?

Important property: the hash function is useful to verify that a given input has not been altered, as any little modification of the input would produce a different hash.

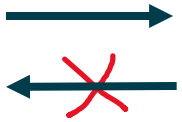
Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. ” O’Reilly Media, Inc.”.

Private Key, Public Key and Bitcoin addresses

https://etherscan.io/ : Block explorer (public)

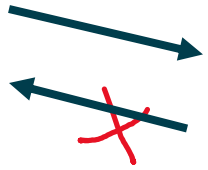
Private key

(256-bit random number)

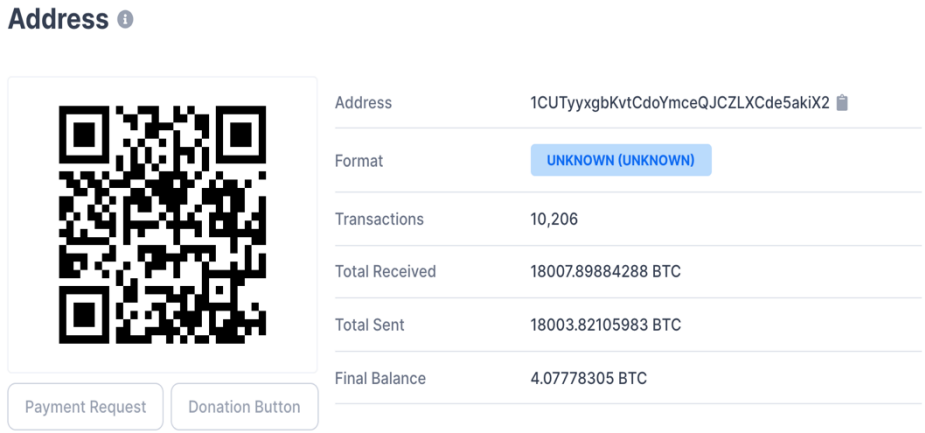


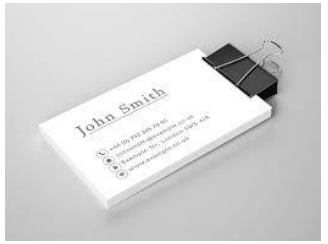
Elliptic curve Cryptography

Public Key



Hash + other





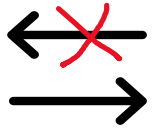
- Data Encryption/Decryption
- Transactions authorizations via digital signature

Private Key, Public Key and Bitcoin addresses

Public key cryptography is used to generate a key pair (**private + public**) that controls access to Bitcoins.

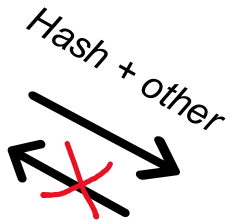
Private key

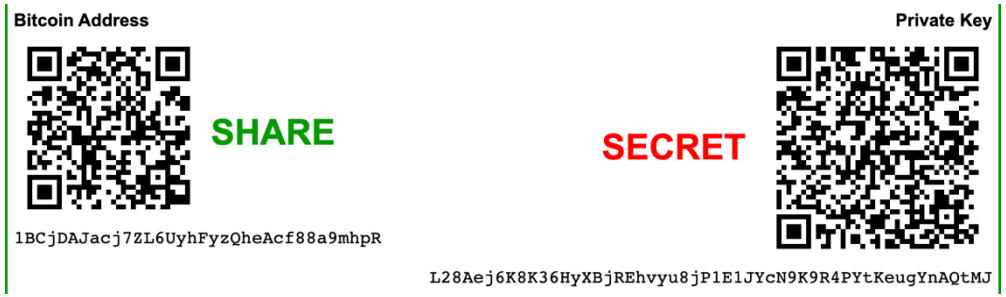
(256-bit random number)



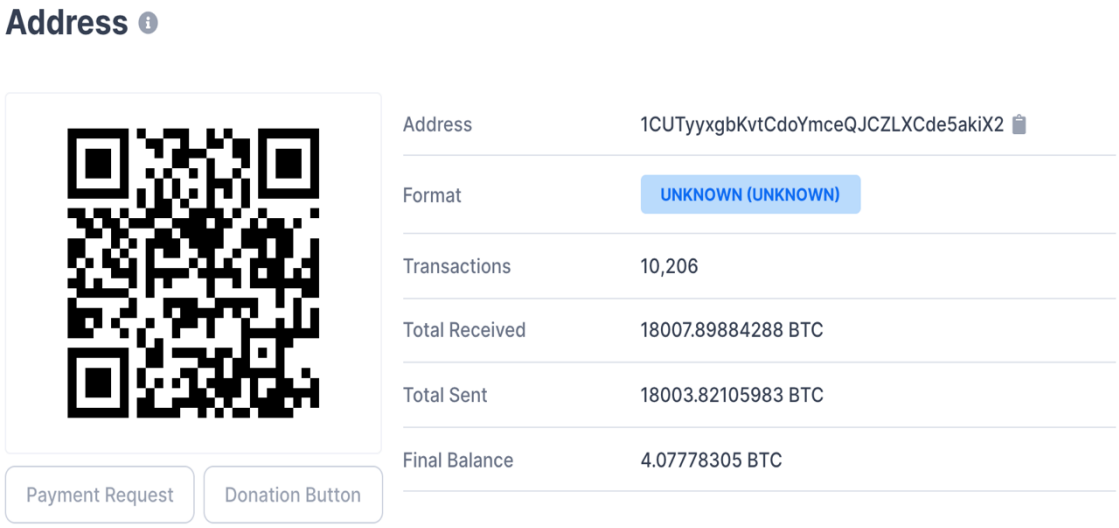
Elliptic curve Cryptography

Public Key





https://www.bitaddress.org/



Properties of hash functions

What is the likelihood of finding a **collision**?

SHA256 = a0dc65ffca799873cbea0ac274015b9526505daaAed385155425f7337704883e

- 64 entries
- Each entry can take 1 out of 16 possible values

TOTAL NUMBER OF POSSIBLE DIFFERENT HASHES

$$16^{64} = 1.15792 * 10^{77} =$$

$$115792089237316195423570985008687907853269984665640564039457584007913129639936 = 2^{256} \text{ (in bits)}$$

- We might think that to find a collision we need to try 10^77 messages, but, in reality, the probability is much smaller…

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Collision probability and birthday problem

What is the likelihood of finding a hash **collision**? = What is the likelihood that out of k people at least 2 have the same birthday (same day - month)?

The goal is to compute *P*(*A*), the probability that at least two people in the room have the same birthday. However, it is simpler to calculate *P*(*A*′), the probability that no two people in the room have the same birthday. We assume that all dates are equally likely (uniform).

N=365 (birthday)

N= $\mathbf{2^{256}}$ (hash)

Prob that at least **two** out of **k** people **do not** have their birthdays on the same day

$$= \frac{N-1}{N} \times \frac{N-2}{N} \times \cdots \times \frac{N-(k-2)}{N} \times \frac{N-(k-1)}{N}$$

$$= \left(1-\frac{1}{N}\right)\left(1-\frac{2}{N}\right) \ldots \left(1-\frac{k-2}{N}\right)\left(1-\frac{k-1}{N}\right)$$

$e^x \sim 1+x$ ➡

$$= \left(e^{-\frac{1}{N}}\right)\left(e^{-\frac{2}{N}}\right) \ldots \left(e^{-\frac{k-2}{N}}\right)\left(e^{-\frac{k-1}{N}}\right) =$$

Gauss formula ➡

$$= e^{-\frac{(1+2+..k-2+k-1)}{N}} = e^{-\frac{k(k-1)}{2N}}$$

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Collision probability and birthday problem

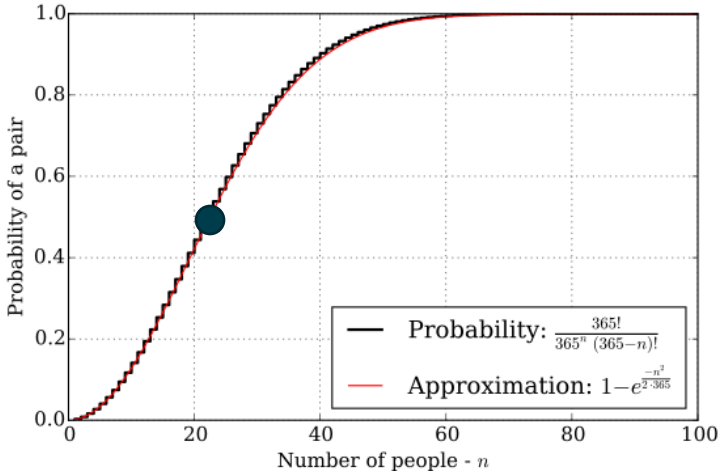
What is the likelihood of finding a hash **collision**? = What is the likelihood that out of N people at least 2 have the same birthday (same day - month)?

N=365 (birthday)

N= 2^{256} (hash)

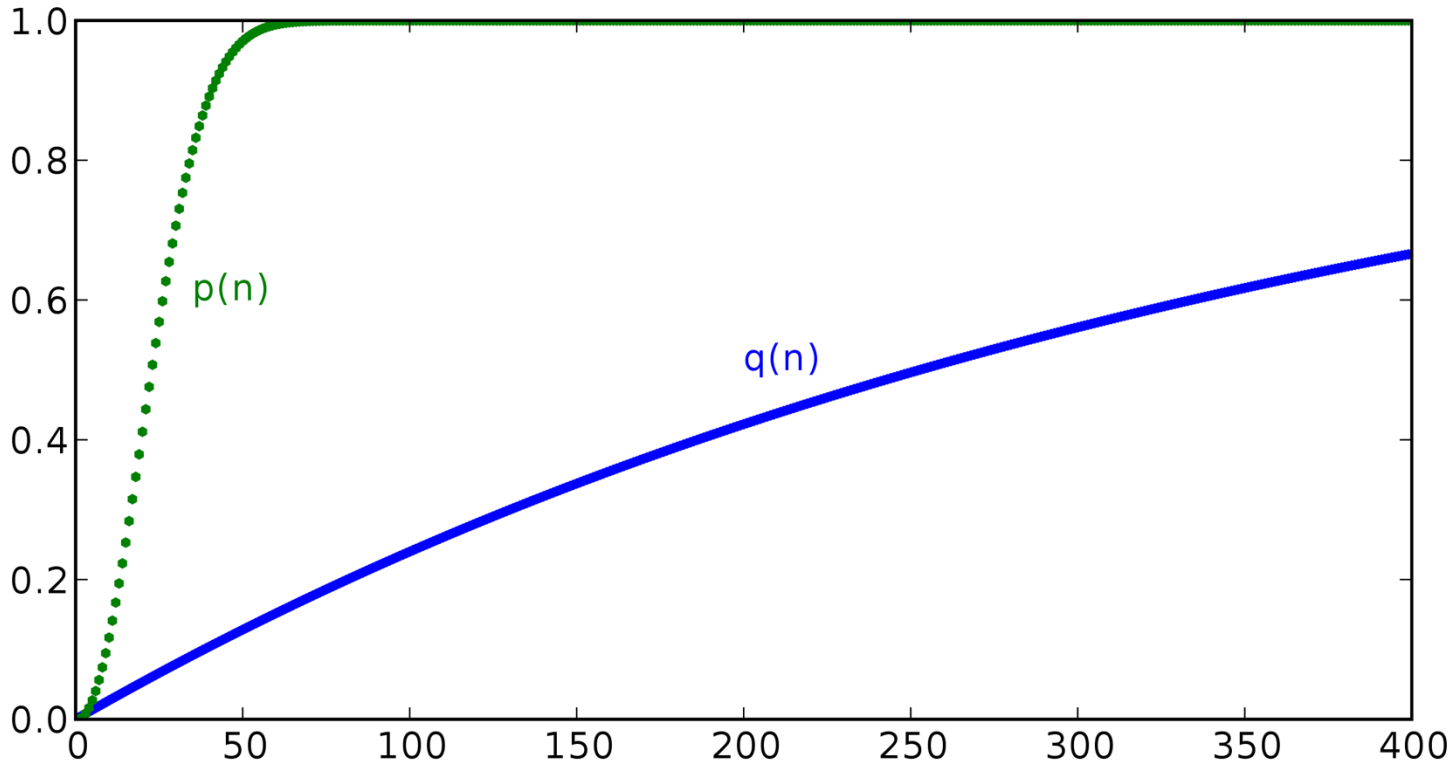
Prob that at least **two** out of **k** people have their birthdays on the same day

$$= 1 - e^{-\frac{k(k-1)}{2N}}$$



Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Collision probability and birthday problem



Comparing $p(n)$ = probability of a birthday match
with $q(n)$ = probability of matching *a fixed* birthday

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

Collision probability

The most important consequence of the birthday attack is that the number of messages we need to hash to find a collision is roughly equal to the square root of the number of possible output values, i.e., about $\sqrt{2^n} = 2^{\frac{n}{2}}$. Hence, for a security level of *x* bit, the hash function needs to have an output length of 2*x* bit.

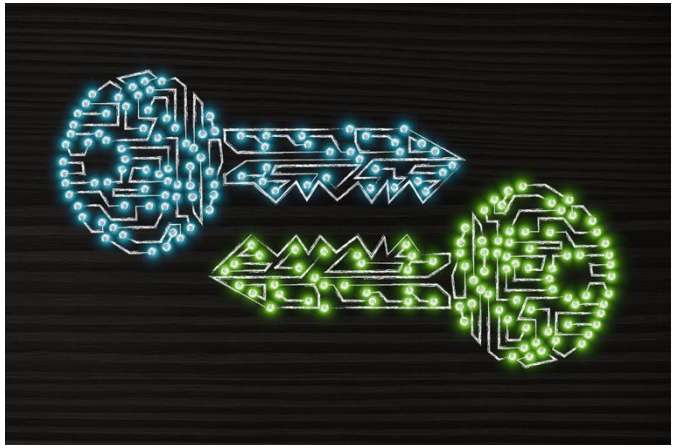


Table 11.1 Number of hash values needed for a collision for different hash function output lengths and for two different collision likelihoods

	Hash output length				
λ	128 bit	160 bit	256 bit	384 bit	512 bit
0.5	2^{65}	2^{81}	2^{129}	2^{193}	2^{257}
0.9	2^{67}	2^{82}	2^{130}	2^{194}	2^{258}

For the moment, those attack are unfeasible for most cryptocurrencies, but the situation may change…

Paar, C., Pelzl, J. (2009). Understanding cryptography: a textbook for students and practitioners. Springer Science & Business Media.

Hash functions and proof of work

Hash functions are also an important part of the so-called **consensus algorithm.**

The consensus algorithm is fundamental so that everyone in the network agrees on a universal “truth” about the state of the system (i.e. who owns what) without the need of a central authority.

The consensus is achieved via an asynchronous interaction of independent nodes that follow the same set of simple rules.

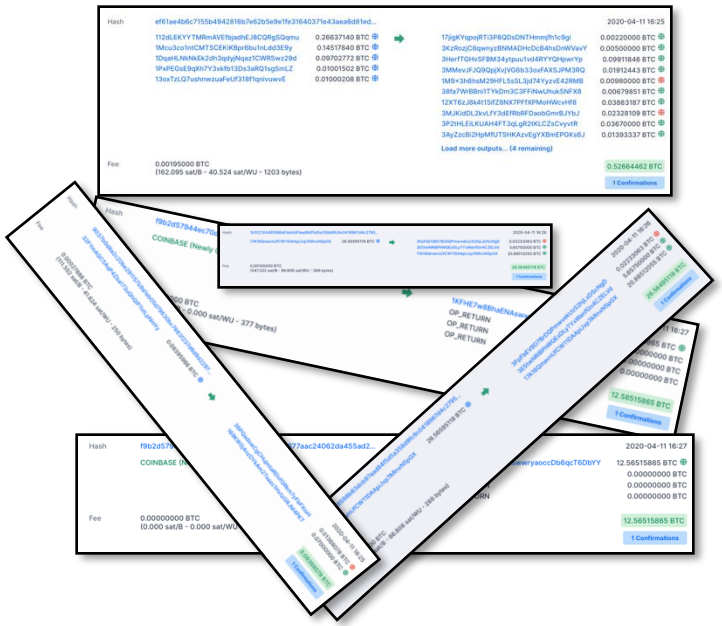
Hash functions and proof of work

Hash functions are also an important part of the so-called **consensus algorithm.**

Important steps:

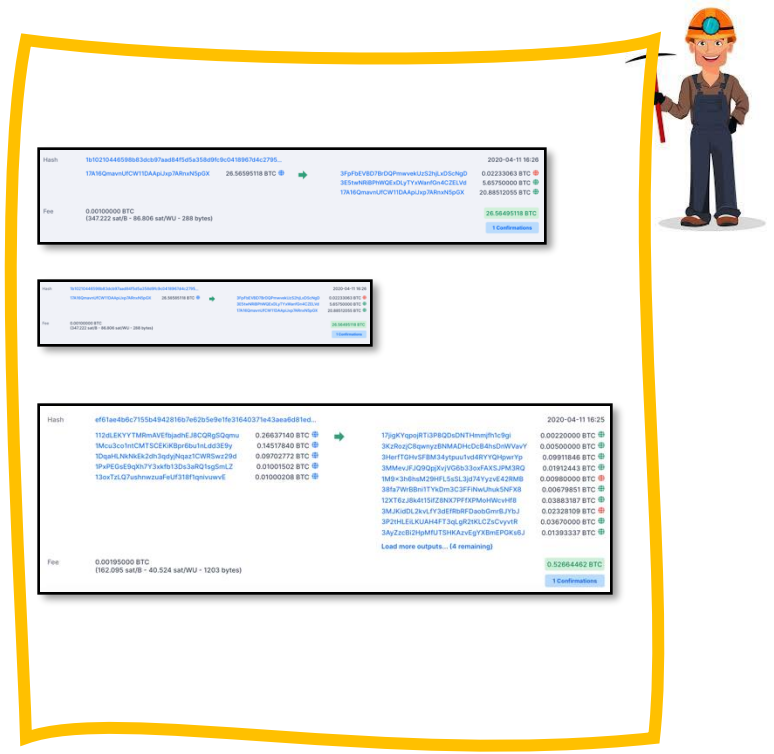
- Independent verification of each transaction based on a comprehensive list of criteria.
- Independent aggregation of those transactions into new blocks by mining nodes, coupled with computation performed via a Proof-of-Work algorithm.
- Independent verification of the new blocks by every node and assembly into a chain.
- Independent selection, by every node, of the chain with the most cumulative computation demonstrated through Proof-of-Work (longest chain).

Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. “O’Reilly Media, Inc.”.

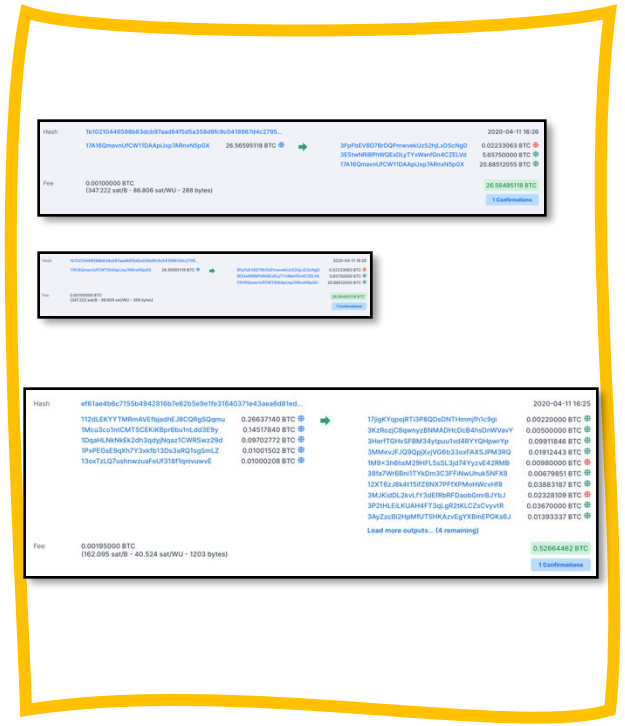


1- Set of unconfirmed transactions (mempool)

2- Creation of the block



3- Solve the task



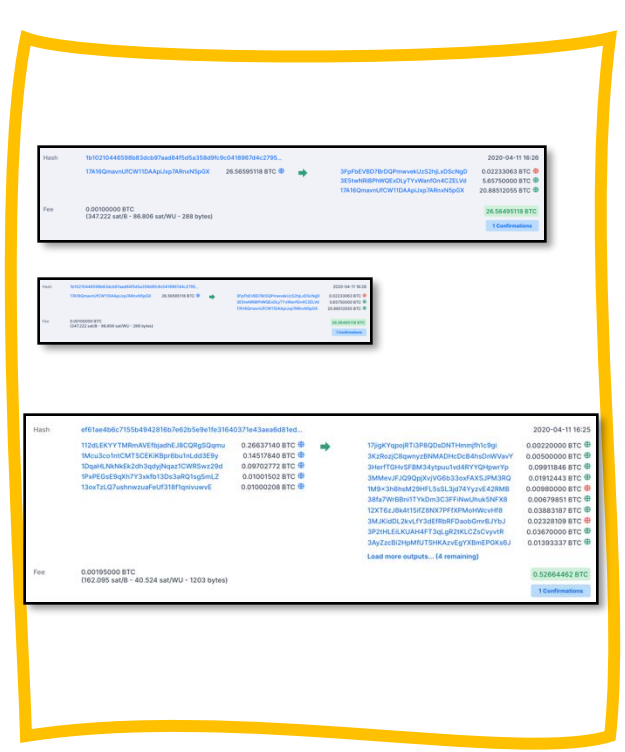
+

Number (the so-called *nonce*)

Hash them together…

00009a88797e94c123bb0cf0166b0918e67f4fef9a1931101dcbf80141a13baf

Challenge - set *target* difficulty: produce a hexadecimal hash that starts with *x* zeros.



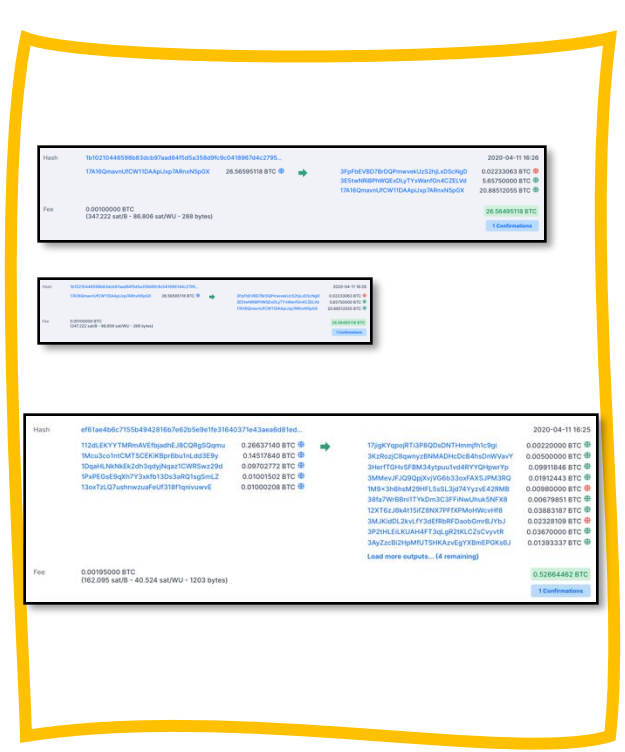
+

Number (the so-called *nonce*)

Hash them together…

00009a88797e94c123bb0cf0166b0918e67f4fef9a1931101dcbf80141a13baf

Challenge - set *target* difficulty: produce a hexadecimal hash that starts with *x* zeros.



+

Number (the so-called *nonce*)

Hash them together…

x

00009a88797e94c123bb0cf0166b0918e67f4fef9a1931101dcbf80141a13baf

…Increasing x (decreasing the target value) makes the search harder.

Try to hash the following sentence and find the nonce that gives you a hash starting with one 0:

UCLCS[add nonce]

- Python: import **hashlib**

hashlib.sha256(b"This is message 2").hexdigest()

- **Go to https://andersbrownworth.com/blockchain/hash**

Try to hash the following sentence and find the nonce that gives you a hash starting with two 0s:

UCLCS[add nonce]

- Python: import **hashlib**

hashlib.sha256(b"This is message 2").hexdigest()

- **Go to https://andersbrownworth.com/blockchain/hash**

Examples..

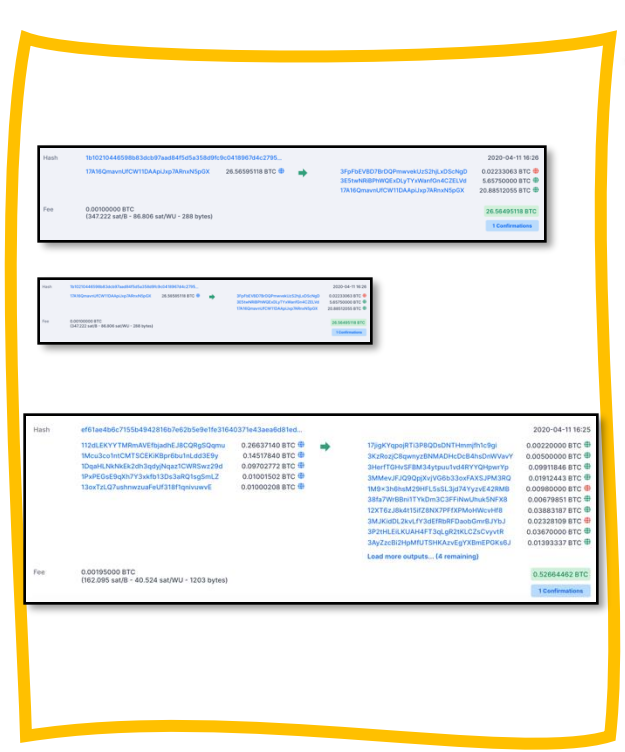
- Let us imagine to flip a coin 100 times trying to get *n* heads in a row. The higher is *n* the lower is the probability to get it.

n	$p(n)$
2	1
3	0.9997
4	0.97
5	0.81
6	0.55
7	0.32
8	0.17
9	0.088
10	0.044
11	0.022
12	0.011
13	0.0053
14	0.0027
15	0.0013
16	0.00063

Examples..

- Consider throwing two dice, trying to throw less than a target. If the target is 12, unless you throw a double six you win. If the target is lowered to 5 more than half of the throws will be invalid.

Challenge - set *target* difficulty: produce a hexadecimal hash that starts with *x* zeros.



+

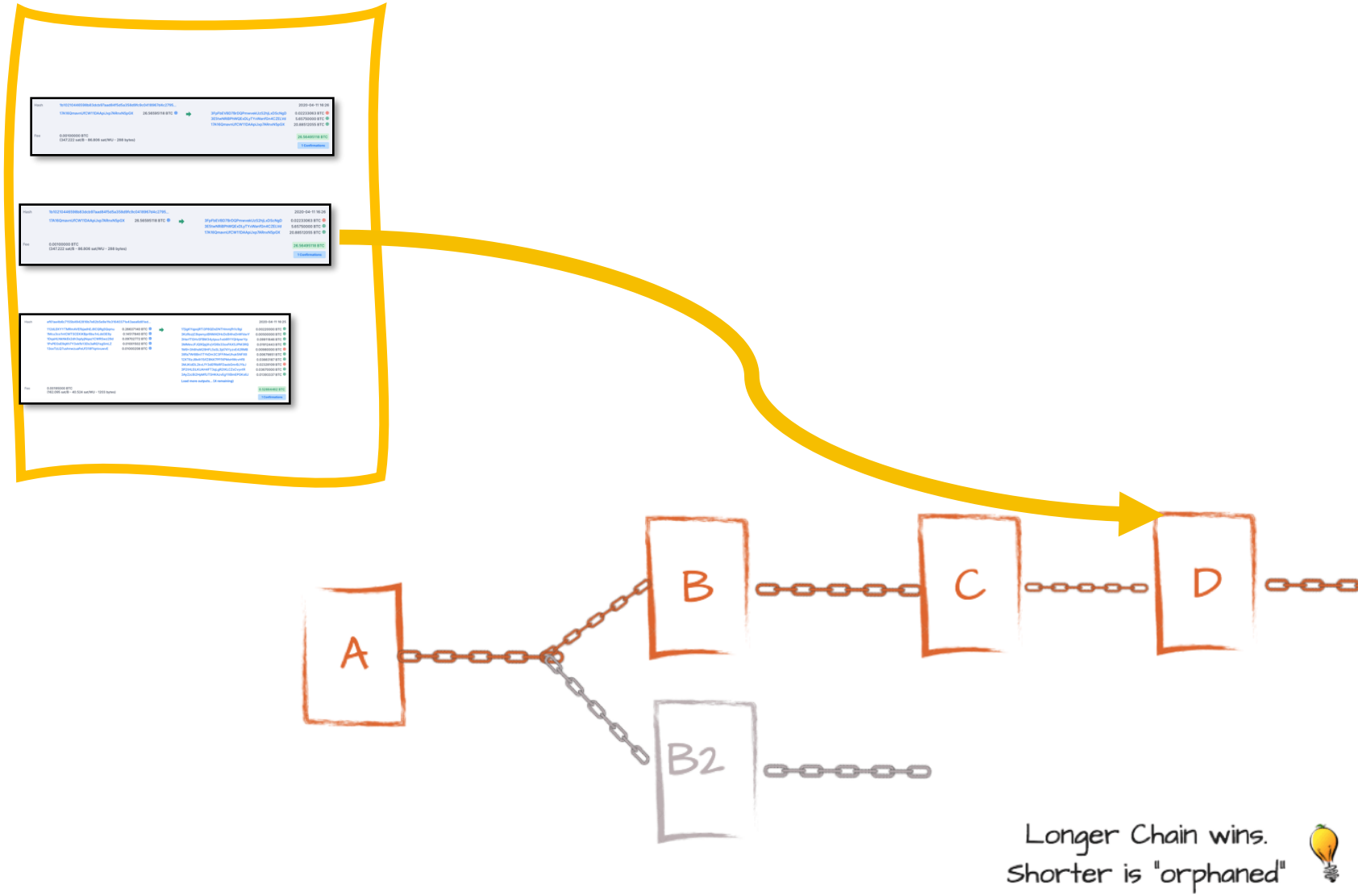
Number (the so-called *nonce*)

Hash them together…

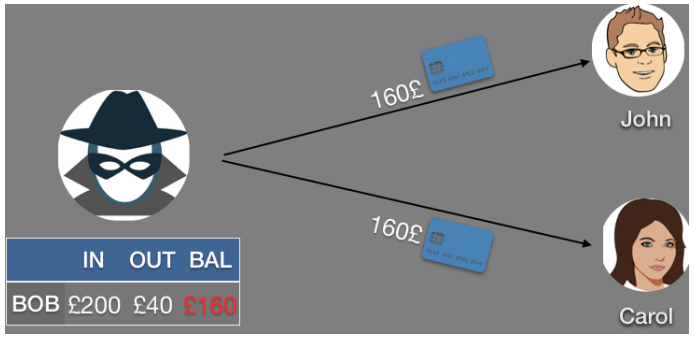
x

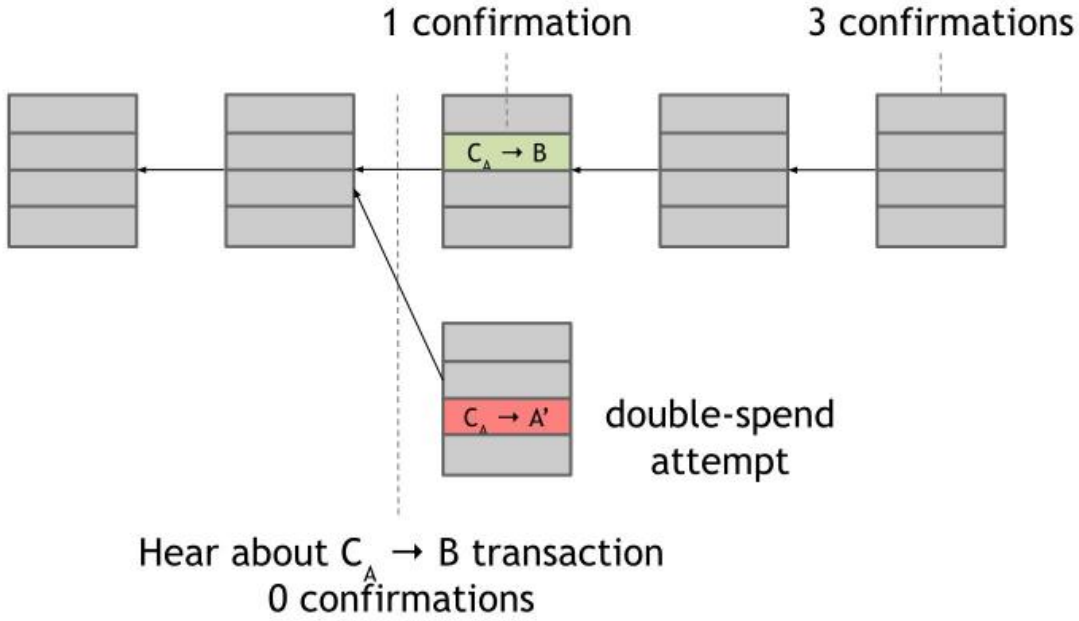
00009a88797e94c123bb0cf0166b0918e67f4fef9a1931101dcbf80141a13baf

Longest chain rule



Side note: double spending attack





- The more confirmations a transaction gets, the higher the probability that it is going to end up on the longest chain.
- A transaction is currently considered confirmed if the block it is included in has at least 6 mined block on top.
- Mining creates trust by ensuring that transactions are only confirmed if enough computational power was devoted to the block that contains them. More blocks mean more computation, which means more trust.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2015). Bitcoin and cryptocurrency technologies.

Adjusting the target difficulty…

- On average, every **10 minutes** a new block is validated. Why 10 minutes?

Ten minutes was specifically chosen as a trade-off between first confirmation time and the amount of work wasted due to chain splits.

After a block is mined, it takes time for other miners to find out about it, and in the meantime, they are competing against the new block instead of adding to it.

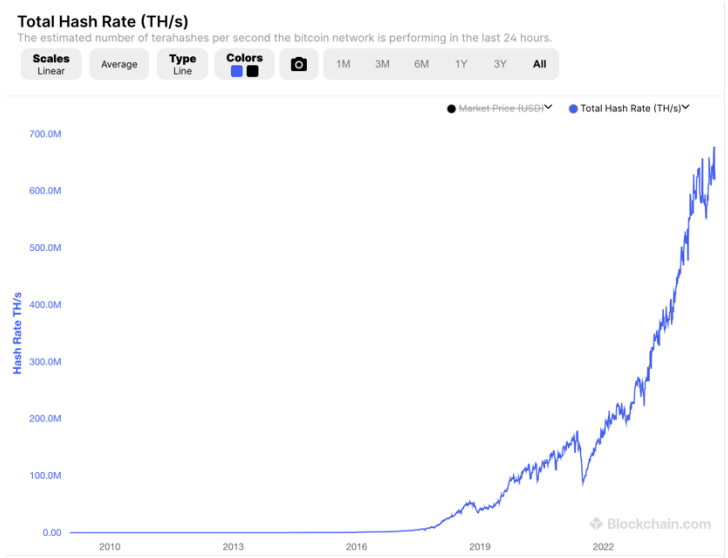
If two blocks are mined simultaneously, the network can only accept one of the two wasting all the work that went into the other block.

Adjusting the target difficulty…

The **hashing power** of miners has increased exponentially every year since Bitcoin was created in 2008.

The growth has also reflected a complete change of technology:

- In 2010 and 2011 many miners switched from using CPU mining to GPU mining.
- In 2013 ASIC mining was introduced with silicon chips specialized for the purpose of mining.



https://www.blockchain.com/charts/hash-rate

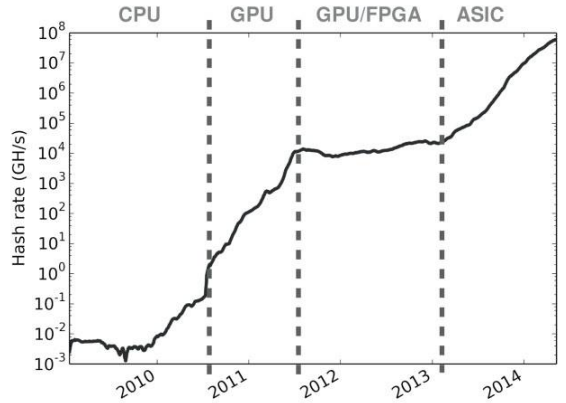
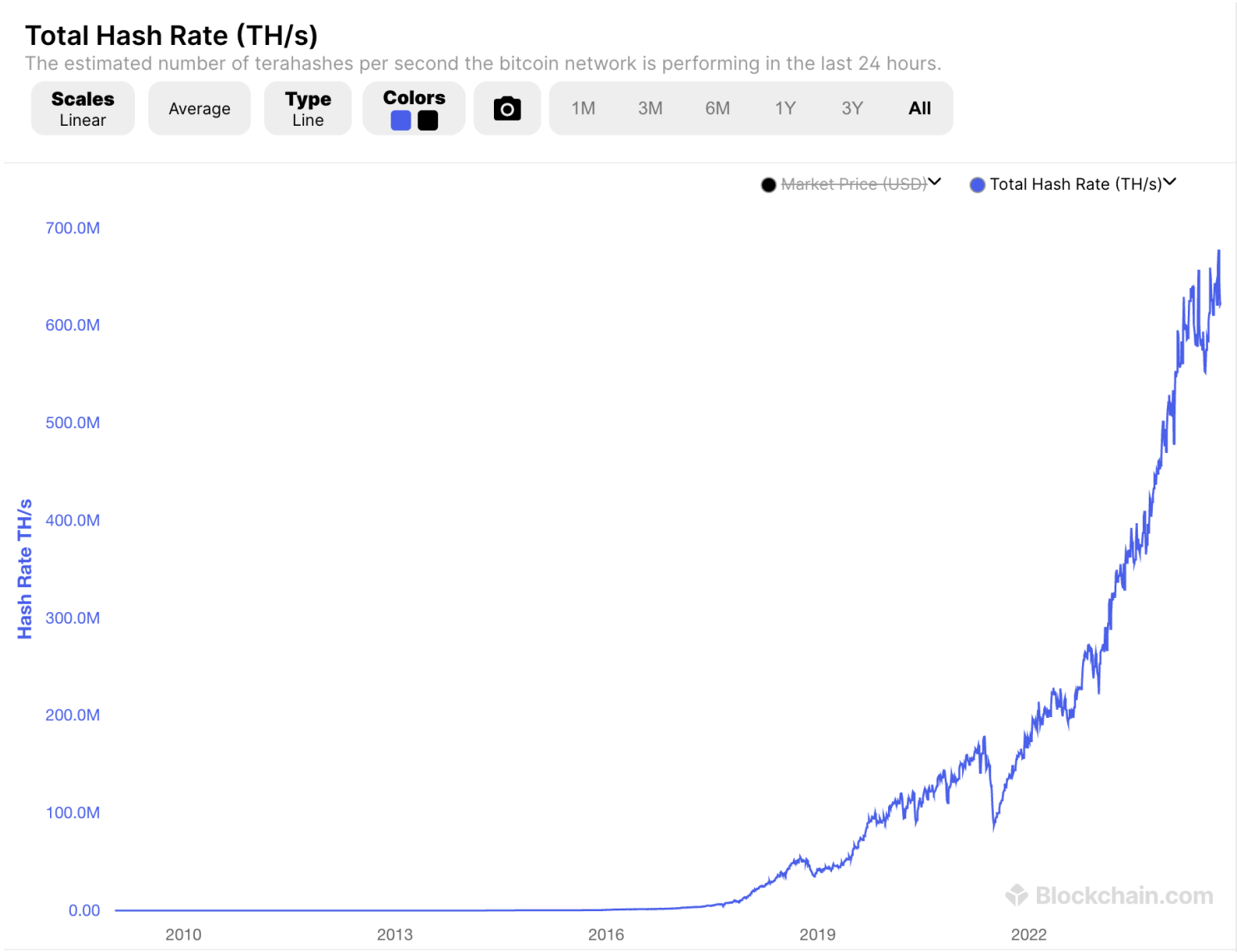


FIGURE 9.2 Hash rate of the Bitcoin network. Hash rate data from blockchain.info

Franco, P. (2014). Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons.



https://www.blockchain.com/charts/difficulty

Mining farms

Key economic issues

A Brief Introduction to Blockchain Economics

Long Chen*, Lin William Cong[†,§] and Yizhou Xiao[‡]

Network security

- Denial of service
  Selfish mining
- 51% attack

Key economic issues

A Brief Introduction to Blockchain Economics

Long Chen*, Lin William Cong†,§ and Yizhou Xiao‡

Network security

- Denial of service
  Selfish mining
- 51% attack

Overconcentration

- Mining pools take gradually over mining.
- No single mining pool takes over the ecosystem.

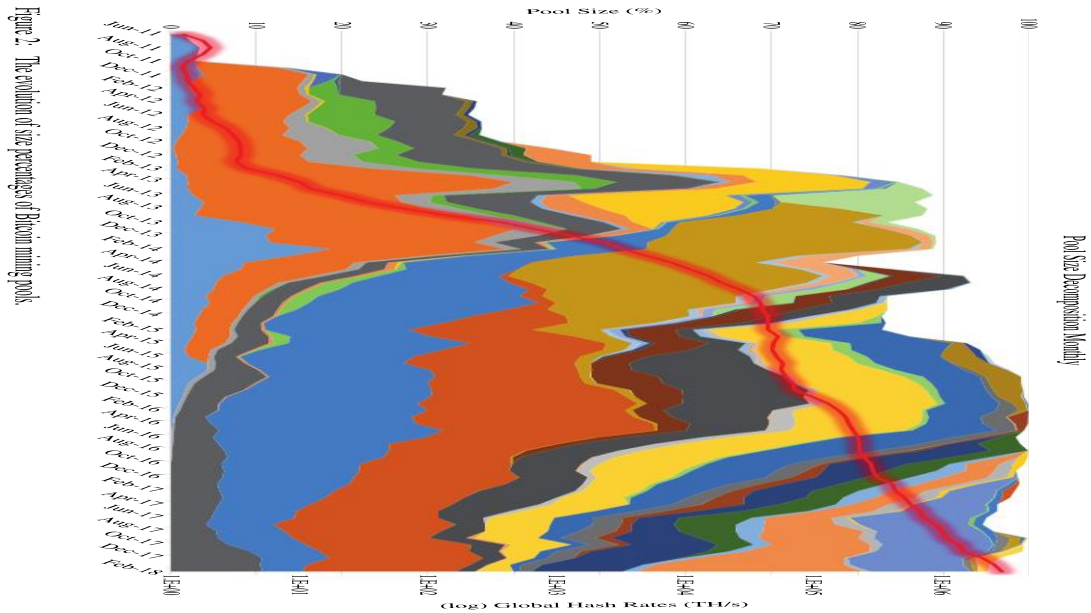
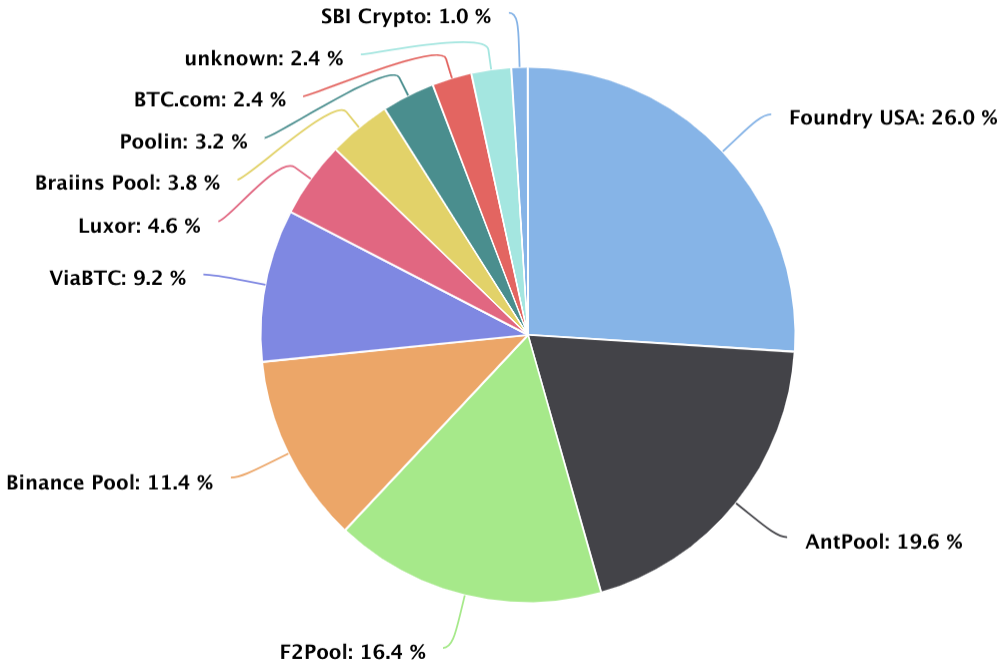
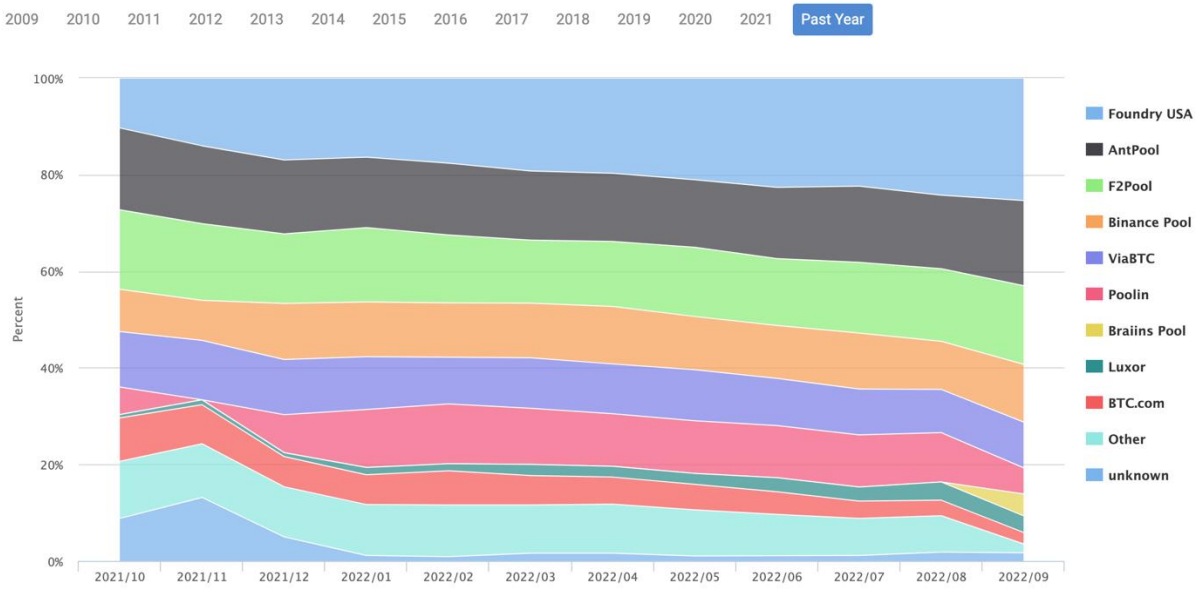


Figure 2. The evolution of size percentages of Bitcoin mining pools.

Mining pools

Miners now collaborate to form mining pools, pooling their hashing power and sharing the reward among thousands of participants. By participating in a pool, miners get a smaller share of the overall reward, but typically get rewarded every day, reducing uncertainty.



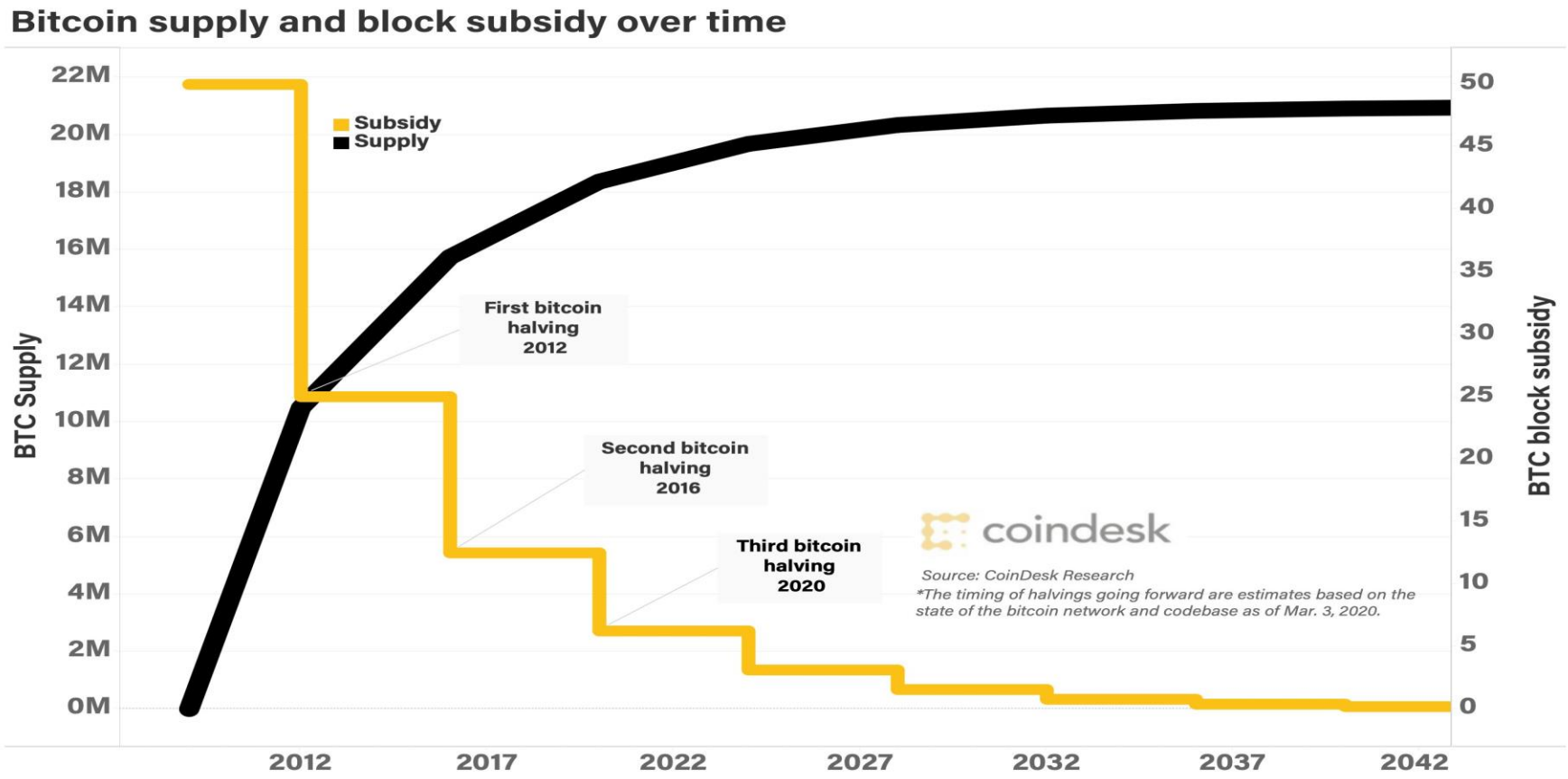


Antonopoulos, A. M. (2014). Mastering Bitcoin: unlocking digital cryptocurrencies. ” O’Reilly Media, Inc.”.

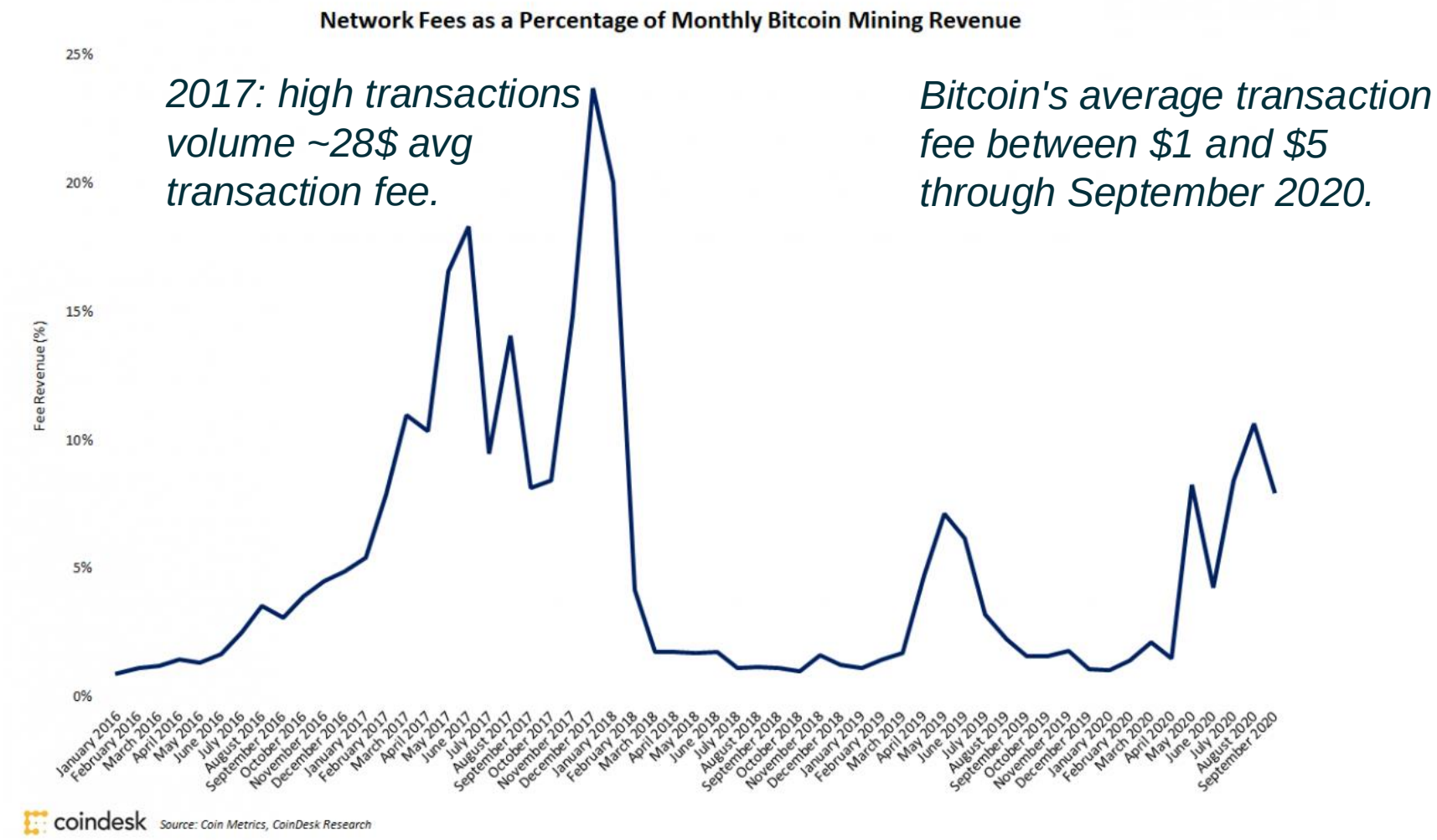
Mining reward

Miners receive two types of rewards in return for the service provided via mining:

- **new coins** created with each new block,
- **transaction fees** from all the transactions included in the block.



Increase in fee revenue important to sustain the network's security as the block reward decreases every four years.



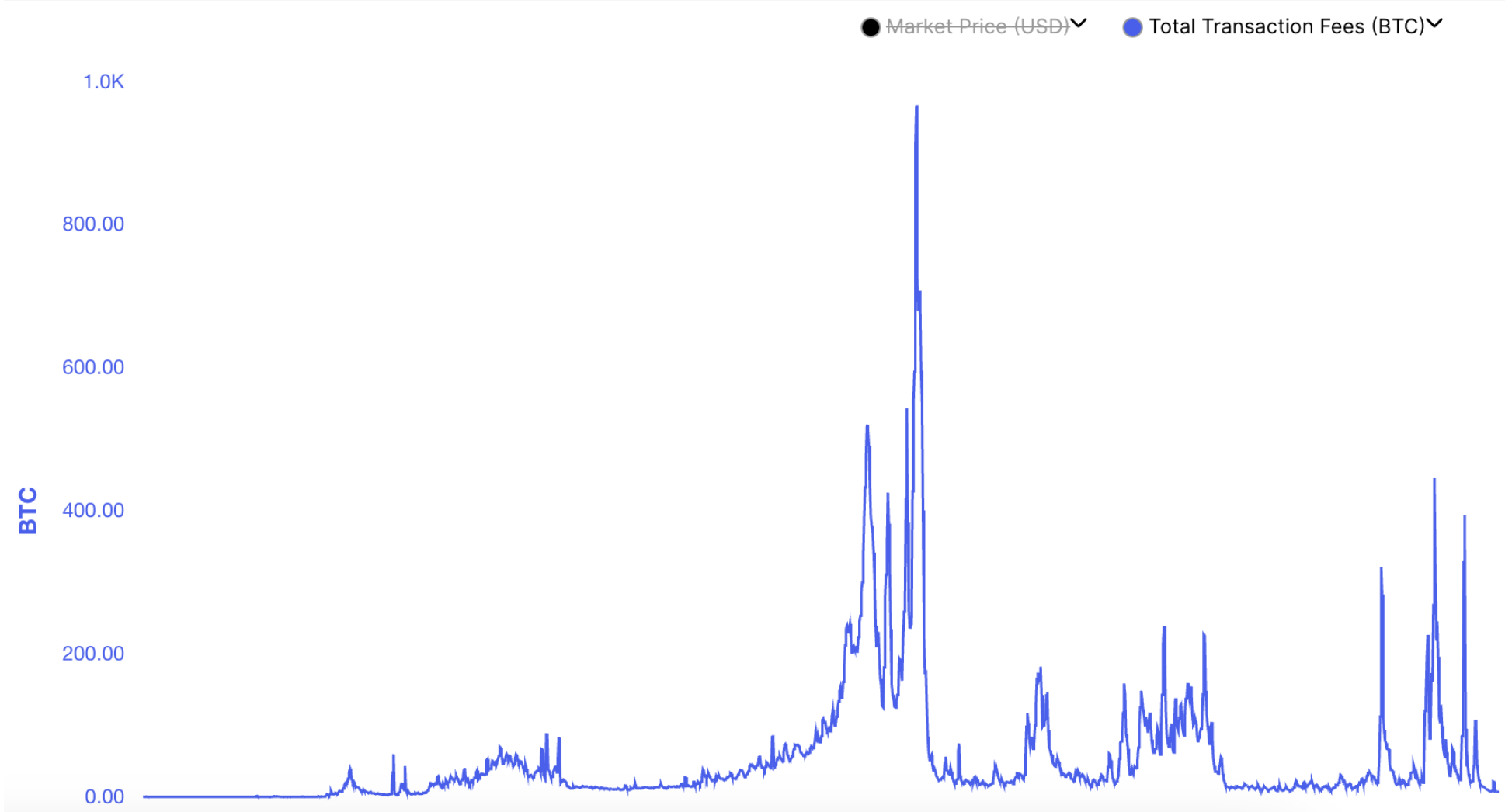
2017: high transactions volume ~28$ avg transaction fee.

Bitcoin's average transaction fee between $1 and $5 through September 2020.

Total Transaction Fees (BTC)

The total BTC value of all transaction fees paid to miners. This does not include coinbase block rewards.

Scales Linear | Average | **Type** Line | **Colors** | 1M | 3M | 6M | 1Y | 3Y | **All**



Solutions for miners' profitability..

- Upgrade their systems to next-generation processor chips that are *faster* and more *energy-efficient*.
- Cutting operational costs: looking at energy prices around the world.
- Increase fees (not convenient for users..).
- Decrease network difficulty (a block can be mined wasting less energy).

If miners cannot cover costs they may decide to drop out of the system impacting the security of the network

Mining pools, revenues & co

- Approximate geographic distribution of miners

  https://cbeci.org/mining_map

- Miners reward/security

  https://www.blockchain.com/charts/miners-revenue

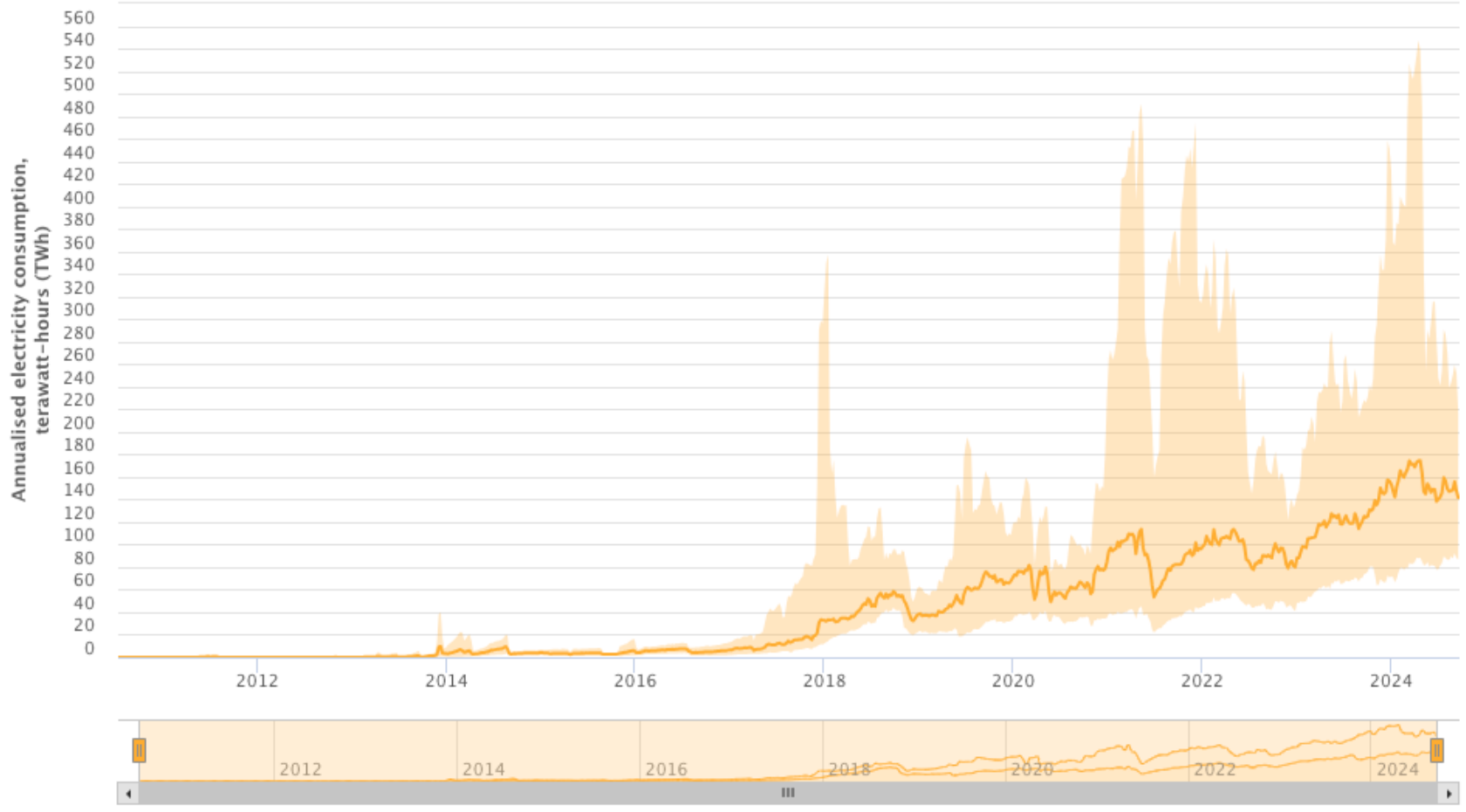
Historical annualised electricity consumption

Select a time window by clicking and dragging to define the start and end dates on the timeline below

Annualised consumption | Power demand

7d 1m 3m 6m YTD 1y All Custom

2010-07-12 - 2024-09-21



Note: a 7-day moving average is applied to reduce the effect of short-term hashrate volatility. The model begins on July 18th 2010 due to a lack of available price data for the prior period.

https://cbeci.org

Navigating challenges

Scalability

- Transaction validation
- Network growth
- Data management

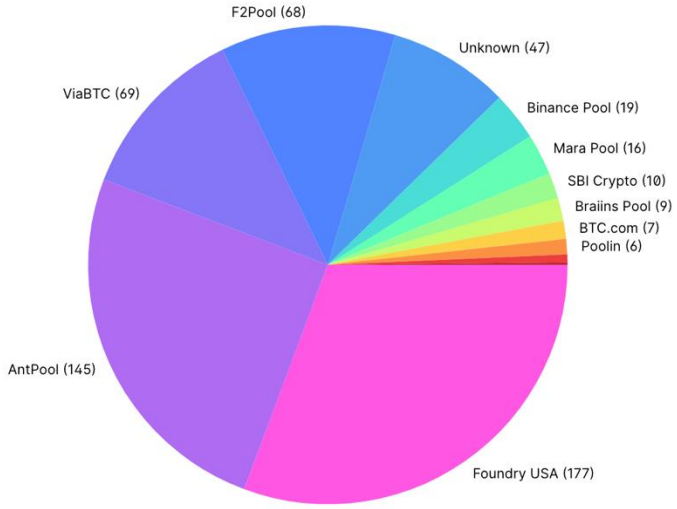
Can validate ~10 transaction per second

Network security

- Denial of service
- Security breaches

Overconcentration

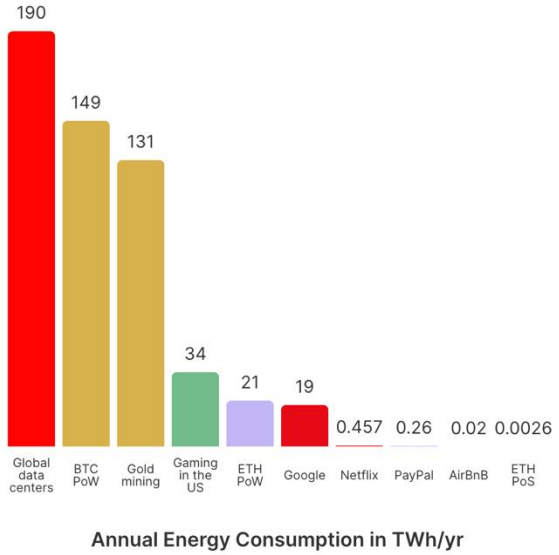
- Growth of hubs and service providers



Source: Blockchain.com

Sustainability

- Decentralised consensus is costly.
- Carbon footprint and energy usage.
- Governance issues



Source: Ethereum Foundation

Table 1. Examples of Bitcoin ASIC Miner Machine Types

Machine	Producer	Hashrate (TH/s)	Power Efficiency (J/TH)	Net Weight (kg)	Weight/Hashrate (kg/TH/s)	Released
Antminer S15	Bitmain	28	57	7	0.25	December 2018
Antminer S9	Bitmain	14	98	4.2	0.30	June 2016
Antminer T9	Bitmain	12.5	126	4.2	0.34	January 2017
Antminer T9+	Bitmain	10.5	127	4.2	0.40	January 2018
Antminer S7	Bitmain	4.73	273	4.5	0.95	September 2015
Antminer S5	Bitmain	1.155	510	2.5	2.16	December 2014
Antminer S4	Bitmain	2	690	7.3	3.65	September 2014
AvalonMiner 821	Canaan	11	109	4.7	0.43	February 2018
AvalonMiner 761	Canaan	8.8	150	5.8	0.66	July 2017
AvalonMiner 741	Canaan	7.3	160	4.3	0.59	April 2017
Bitfury B8	Bitfury	47	130	37	0.79	December 2017

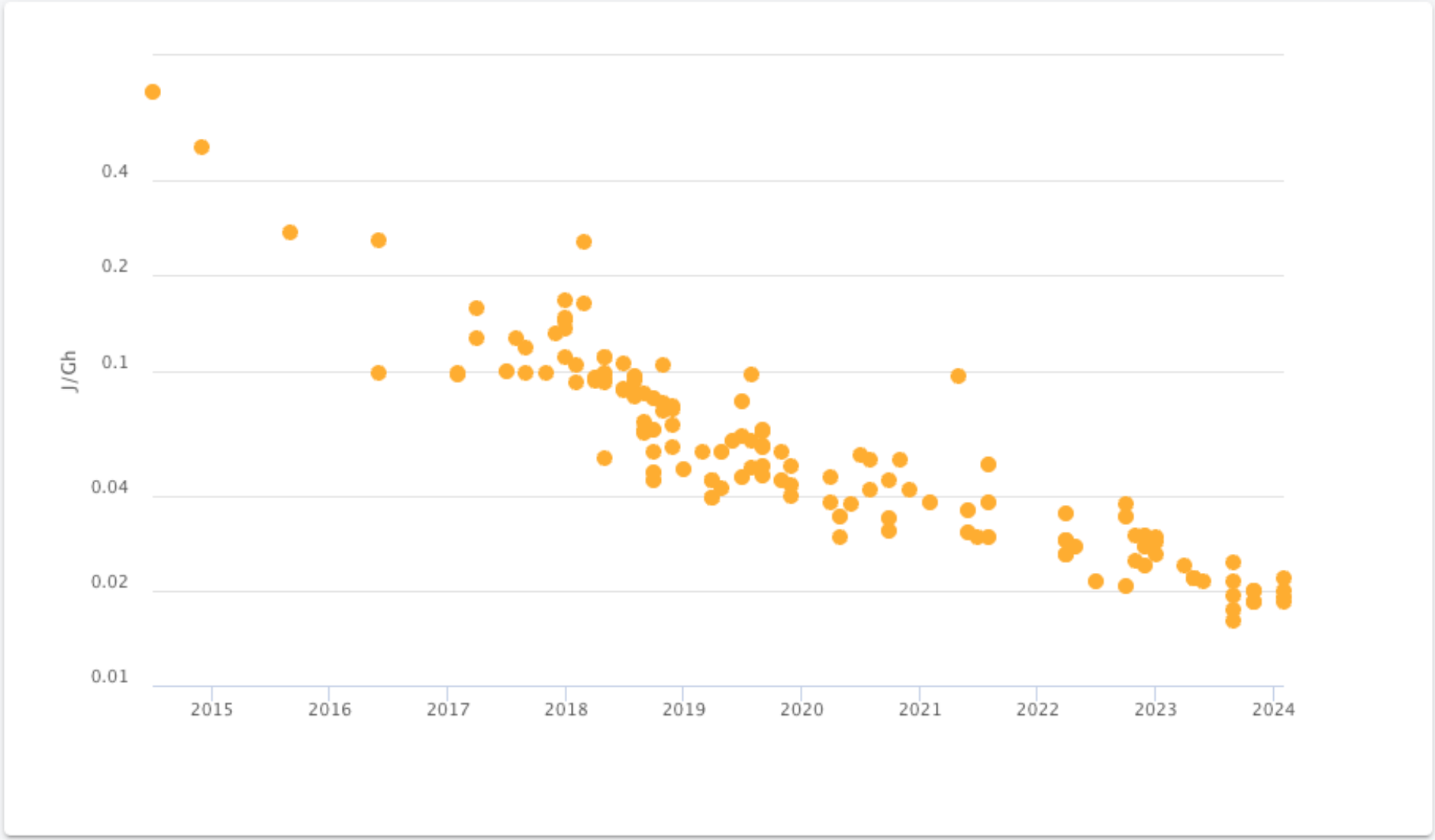
Source: Bitmain, Bitfury, and Canaan.

Mining equipment becomes obsolete every ~1.5 years.

For ASIC mining machines, there is no purpose beyond the singular task they were created to do, meaning they immediately become electronic waste (e-waste) afterward.

de Vries, A. (2019). Renewable energy will not solve bitcoin's sustainability problem. *Joule*, *3*(4), 893-898.

Figure 1: Evolution of bitcoin mining equipment efficiency



Note: *A 1,000 W mining device that generates 10 Th/s has an efficiency of 100 J/Th. This chart is based on a list of 100+ SHA-256 mining equipment.*

https://cbeci.org

Bitcoin & renewables

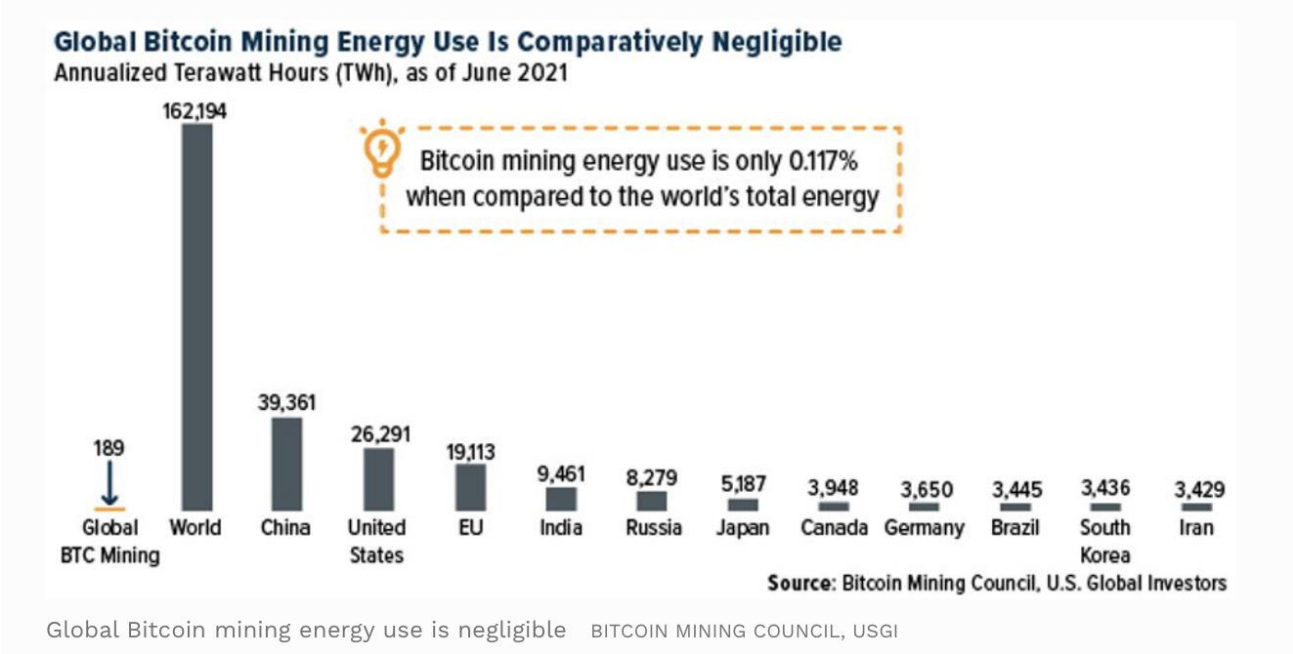
Bitcoin enthusiasts have long maintained that bitcoin mining could drive a clean energy transition.

Bitcoin miners buy the cheapest sources of energy available -> renewable (wind and solar) sources of energy are getting progressively cheaper and will soon outmatch thermal energy on cost -> bitcoin miners will therefore subsidize the buildout of renewable energy, benefiting everyone.

MARKETS

Bitcoin Mining Uses A Higher Mix Of Sustainable Energy Than Any Major Country Or Industry

Frank Holmes Contributor
Great Speculations Contributor Group



Global Bitcoin mining energy use is negligible BITCOIN MINING COUNCIL, USGI

TECH NEWS

Bitcoin miners align with fossil fuel firms, alarming environmentalists

CNBC news

“Miners don’t just need cheap energy, but a stable source of power because their machines need to run 24/7, and fossil fuel sources are best suited for it […] Miners are reviving gas plants and idle coal mines in places like New York and Montana.”

Cryptocurrencies

Bitcoin miners revived a dying coal plant – then CO_2 emissions soared

The Guardian

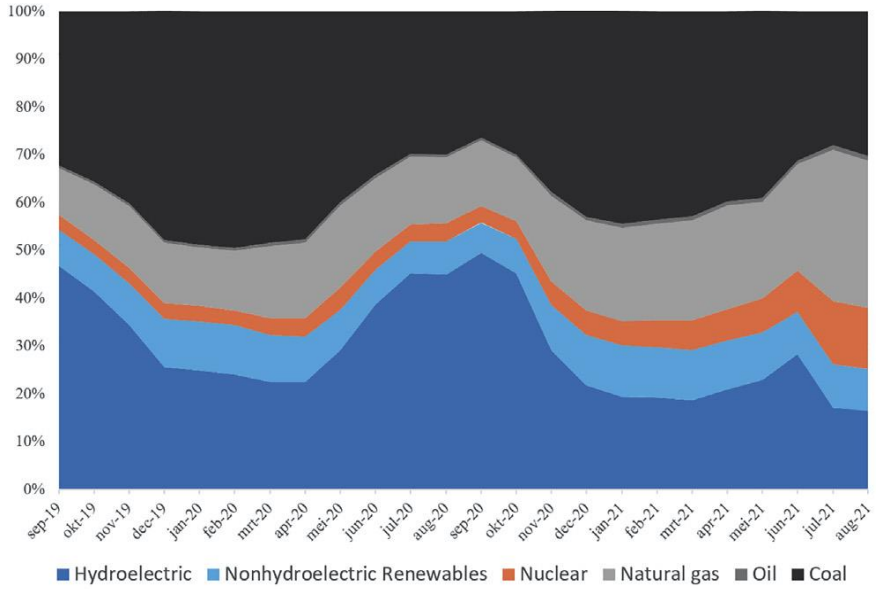
China Banished Cryptocurrencies. Now, 'Mining' Is Even Dirtier.

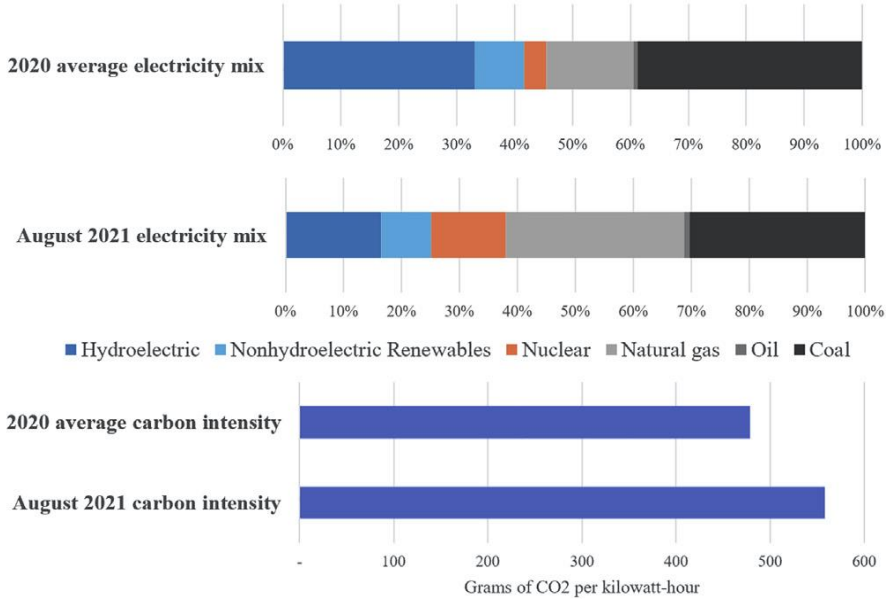
New research shows that China's Bitcoin ban has sent the process of creating new coins, known as mining, to countries where it uses far less renewable energy.

NY Times

China's Bitcoin mining hash rate share fell from over 60% in May to near zero in August, with miners moving to the United States, Russia and Kazakhstan.

Consequently, the share of natural gas in the electricity mix nearly doubled from 15% to 30.8% and the emission factor of coal-fired power generation potentially increased due to high-emitting plants in Kazakhstan compared to China.





de Vries, A., Gallersdörfer, U., Klaaßen, L., & Stoll, C. (2022). Revisiting Bitcoin's carbon footprint. *Joule*, *6*(3), 498-502.

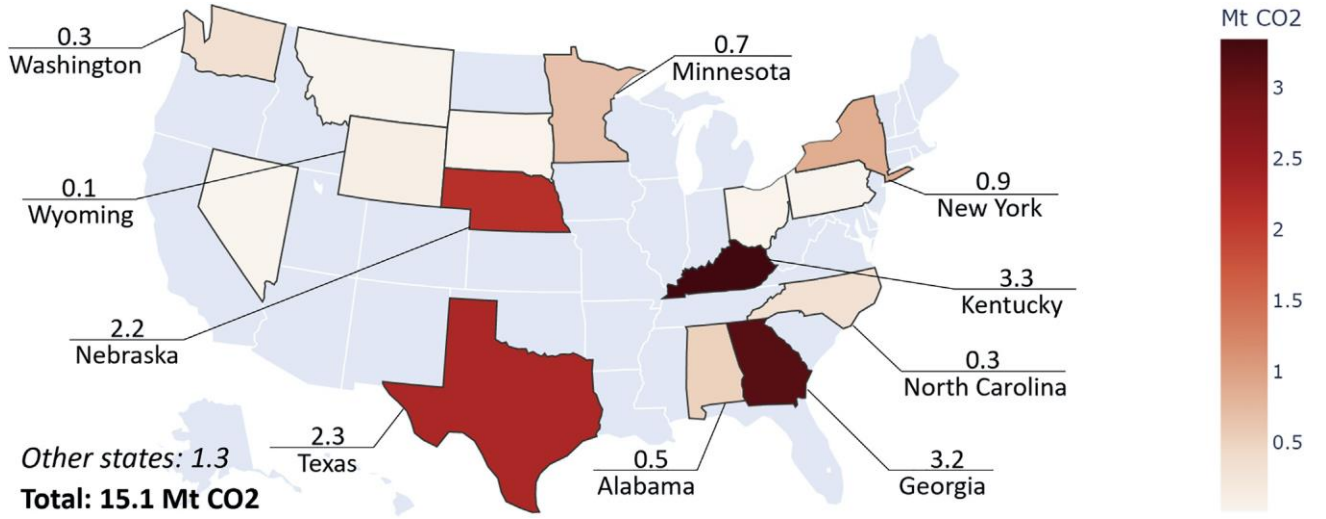


Figure 3. Estimated carbon footprint of the Bitcoin network in the United States, as of August 2021

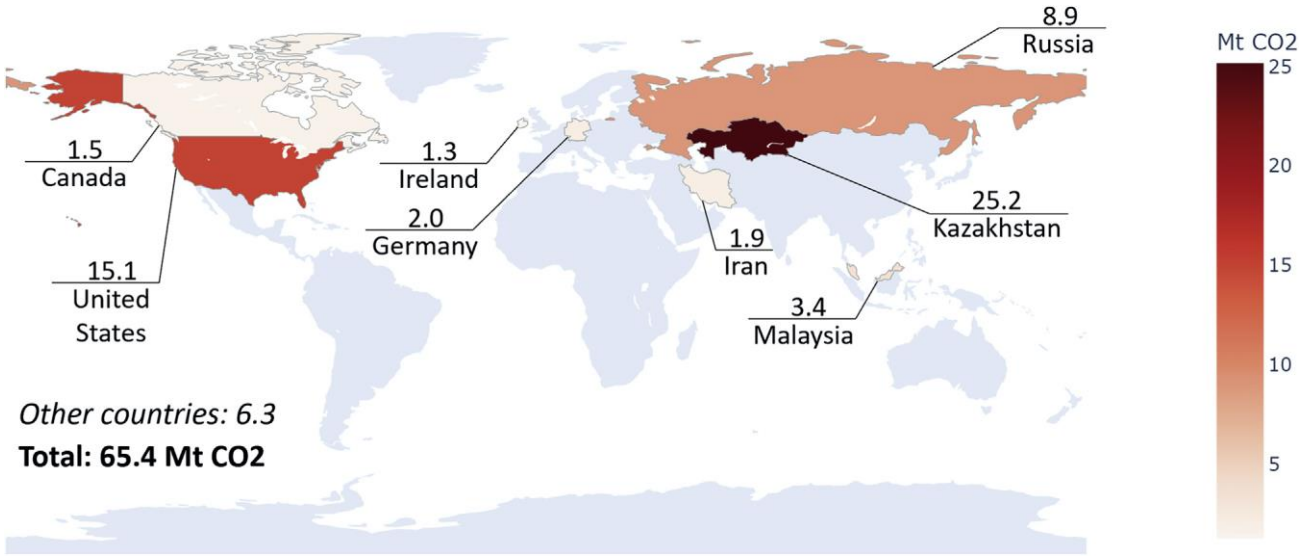
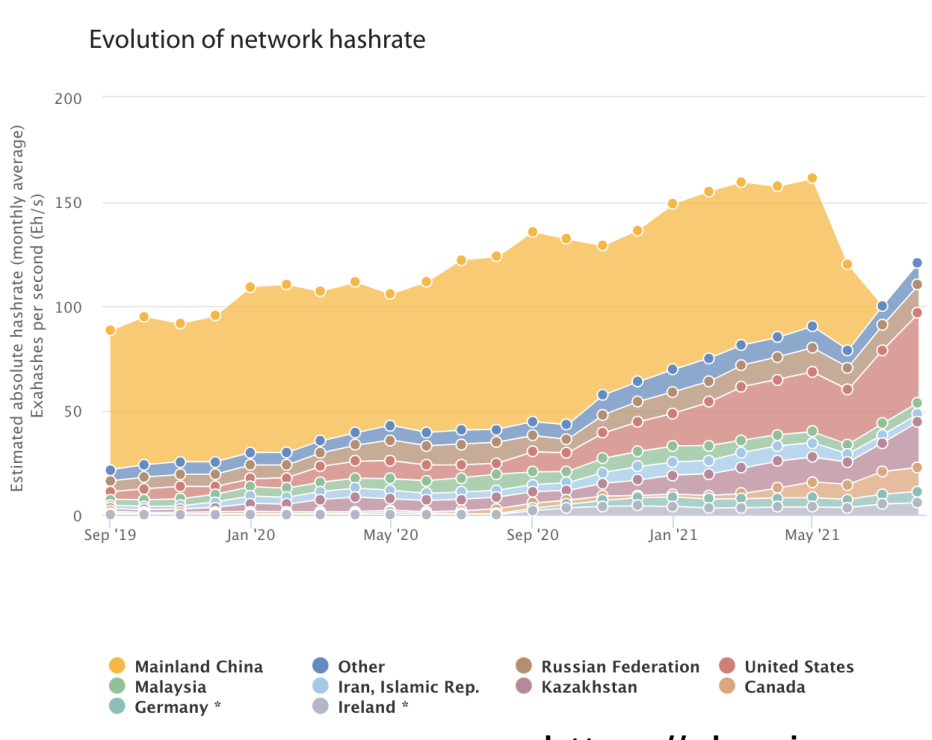


Figure 2. Estimated global carbon footprint of the Bitcoin network, as of August 2021



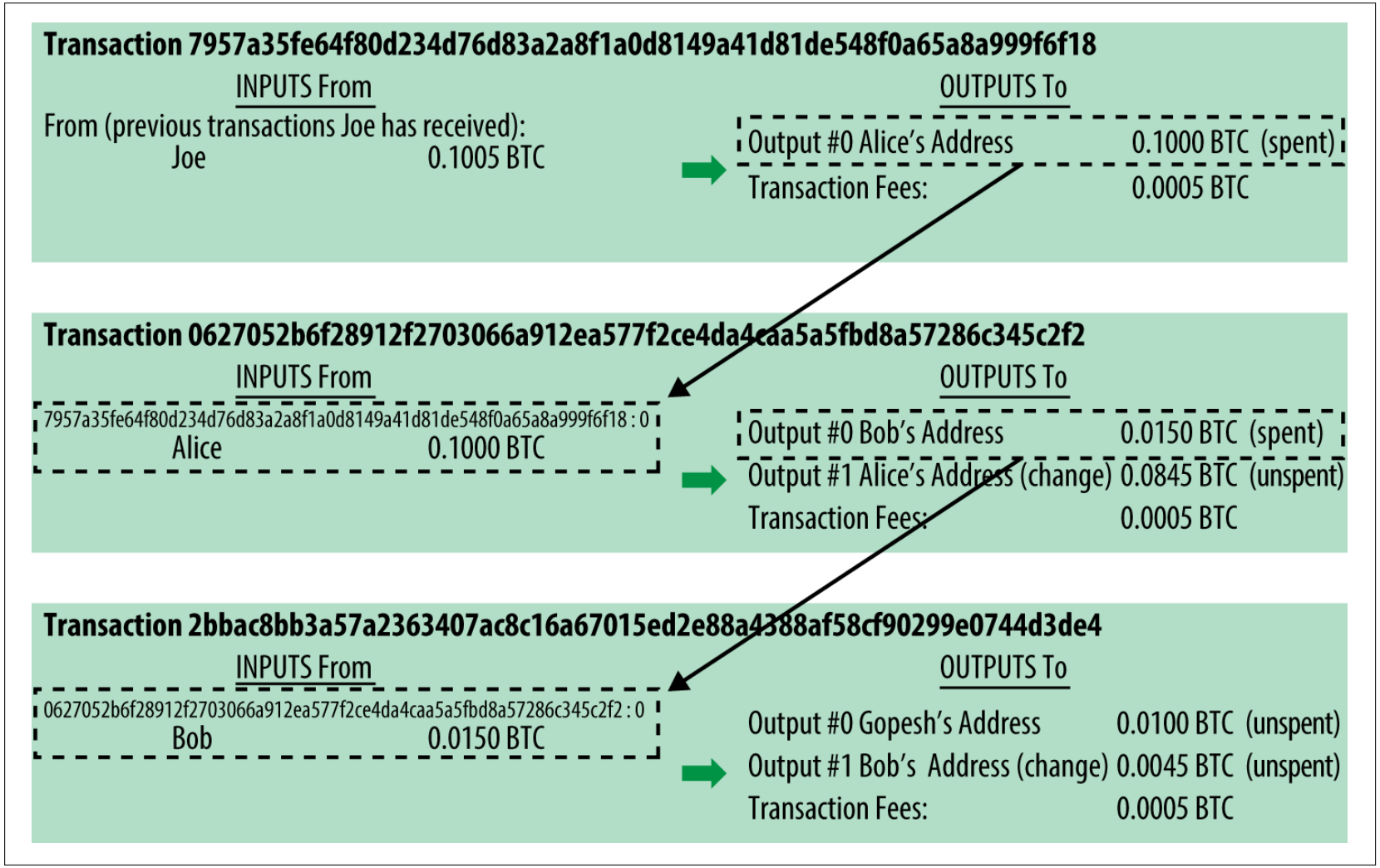
https://cbeci.org

de Vries, A., Gallersdörfer, U., Klaaßen, L., & Stoll, C. (2022). Revisiting Bitcoin's carbon footprint. *Joule*, *6*(3), 498-502.

Transactions format & scripts

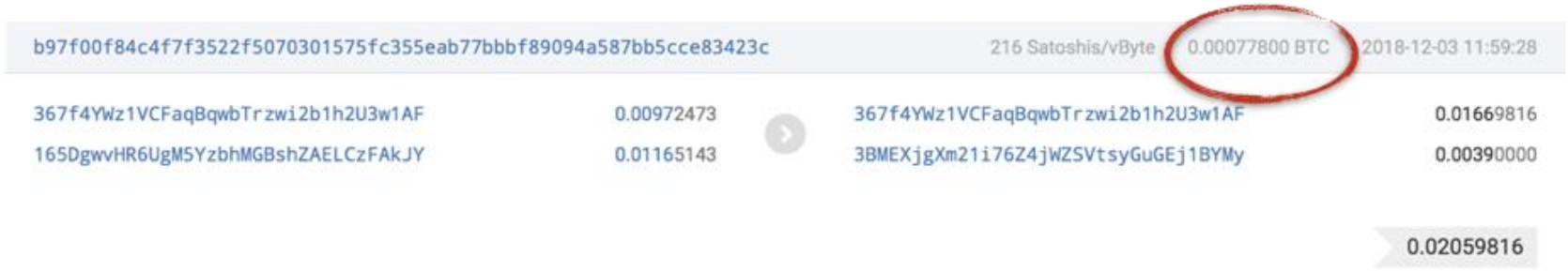
Transactions are like lines in a **double-entry bookkeeping ledger**. In simple terms, each transaction contains one or more “inputs,” which are debits against a bitcoin account. On the other side of the transaction, there are one or more “outputs,” which are credits added to a bitcoin account.

Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.".


Transaction 7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18
INPUTS From
From (previous transactions Joe has received):
Joe 0.1005 BTC
OUTPUTS To
Output #0 Alice's Address 0.1000 BTC (spent)
Transaction Fees: 0.0005 BTC
Transaction 0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2
INPUTS From
7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18 : 0
Alice 0.1000 BTC
OUTPUTS To
Output #0 Bob's Address 0.0150 BTC (spent)
Output #1 Alice's Address (change) 0.0845 BTC (unspent)
Transaction Fees: 0.0005 BTC
Transaction 2bbac8bb3a57a2363407ac8c16a67015ed2e88a4388af58cf90299e0744d3de4
INPUTS From
0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2 : 0
Bob 0.0150 BTC
OUTPUTS To
Output #0 Gopesh's Address 0.0100 BTC (unspent)
Output #1 Bob's Address (change) 0.0045 BTC (unspent)
Transaction Fees: 0.0005 BTC

Transaction fees

Transactions are added to the block prioritising them by the amount of fees offered.



The data structure of transactions does not have a field for fees. Fees are implied as the difference between the sum of inputs and the sum of outputs:

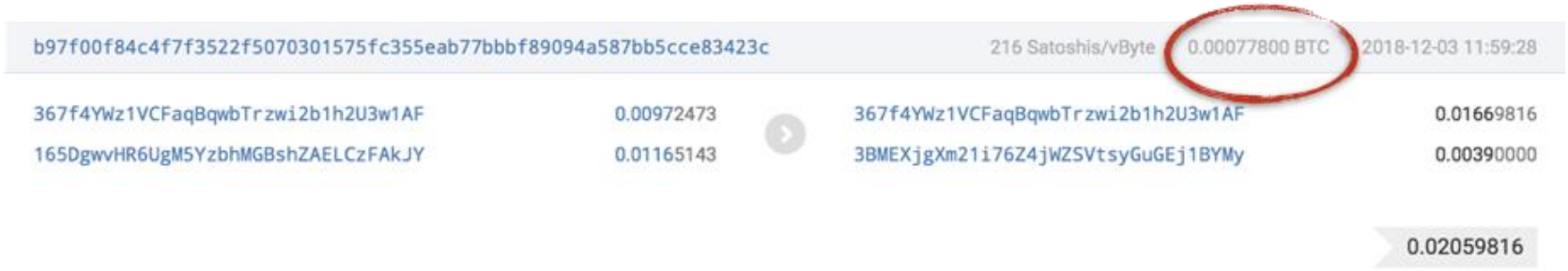
```
Fees = Sum(Inputs) - Sum(Outputs)
```

! If you consume a 10-bitcoin UTXO to make a 1-bitcoin payment, you must include a 9-bitcoin change output back to your wallet. Otherwise, the 9-bitcoin “leftover” will be counted as a transaction fee and will be collected by the miner who mines your transaction in a block.

Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.".

Transaction fees

Transactions are added to the block prioritising them by the amount of fees offered.



The data structure of transactions does not have a field for fees. Fees are implied as the difference between the sum of inputs and the sum of outputs:

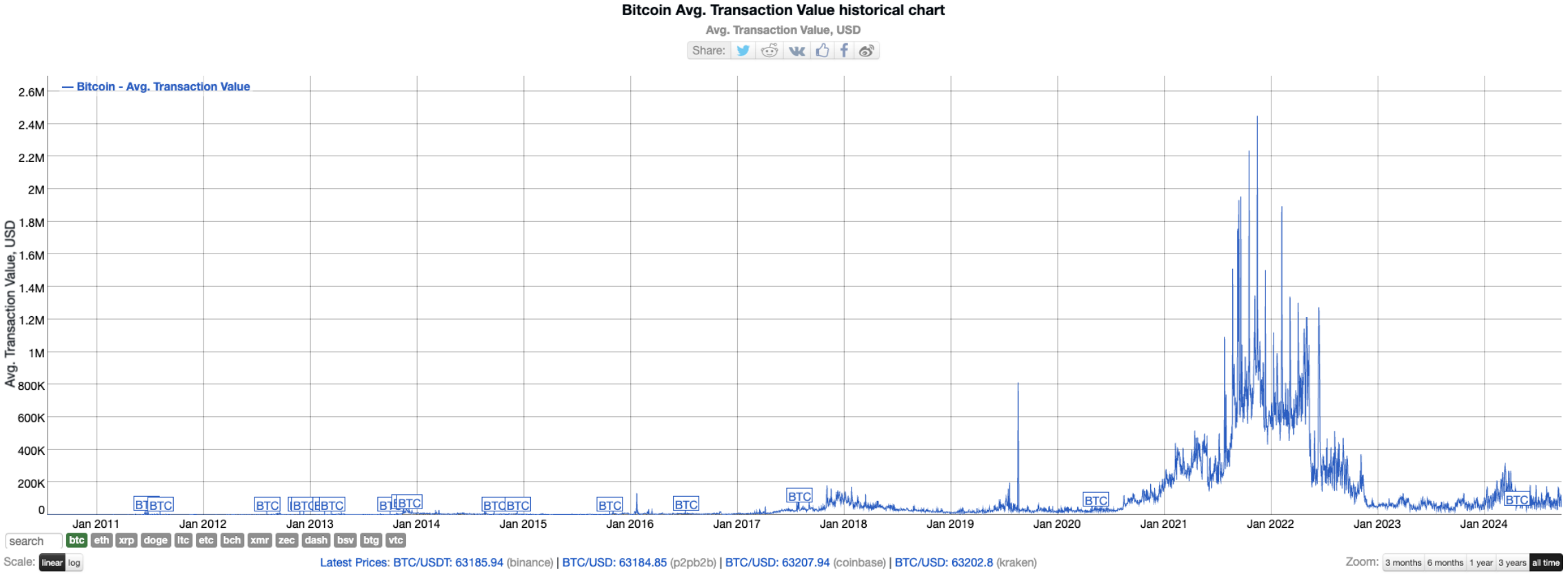
```
Fees = Sum(Inputs) - Sum(Outputs)
```

! If you consume a 10-bitcoin UTXO to make a 1-bitcoin payment, you must include a 9-bitcoin change output back to your wallet. Otherwise, the 9-bitcoin “leftover” will be counted as a transaction fee and will be collected by the miner who mines your transaction in a block.

Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.".

How to choose the right fee?

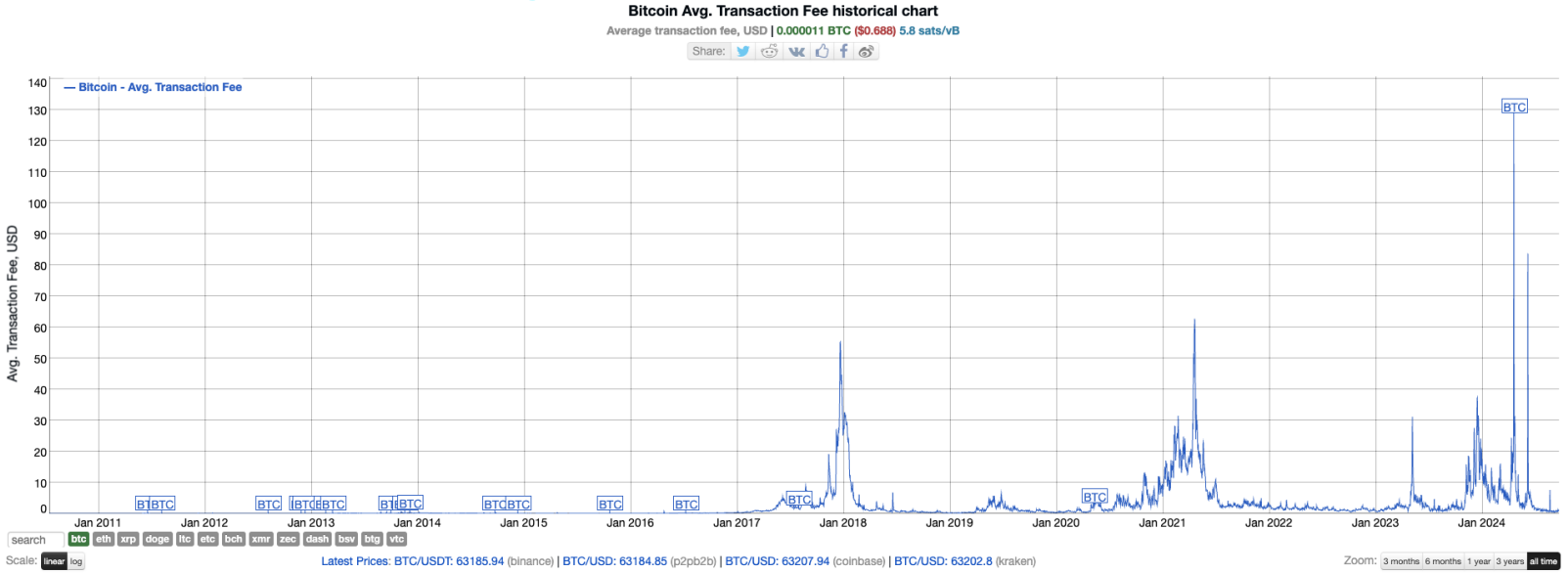
Fees are normally calculated **per byte of transaction data**: they are not proportional to the amount of Bitcoin transferred but to the size/complexity of the transaction.

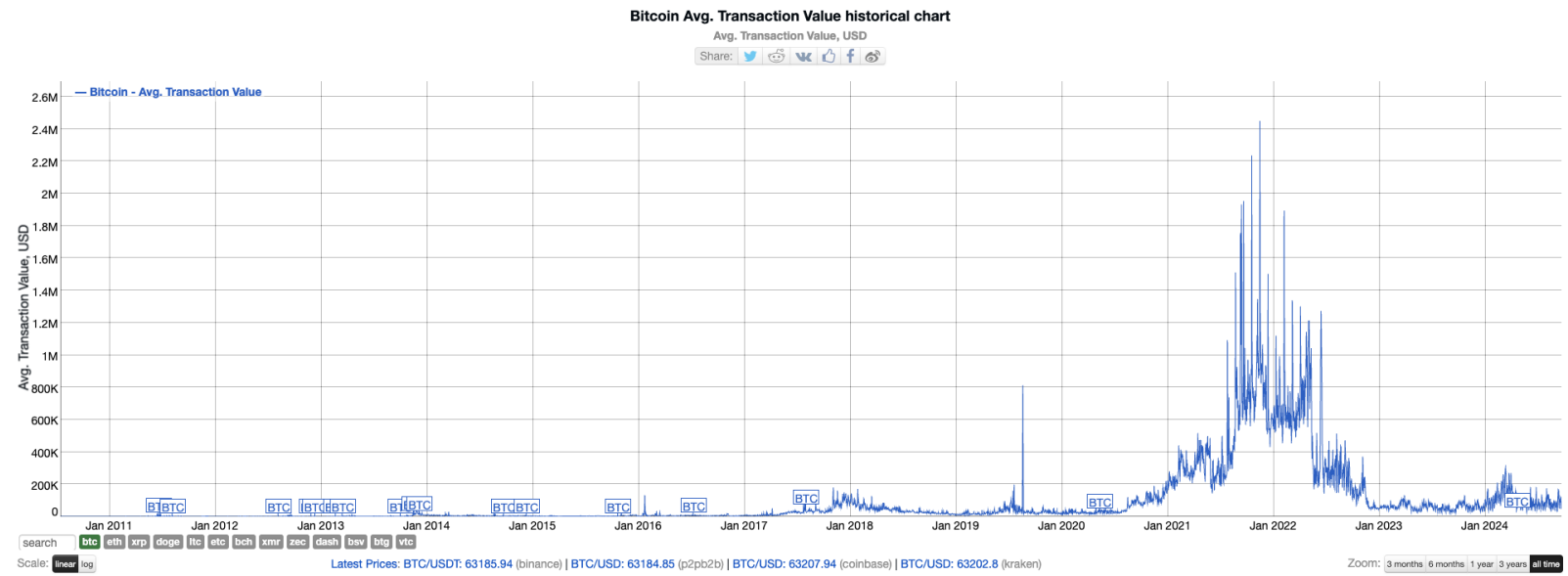


https://bitinfocharts.com/

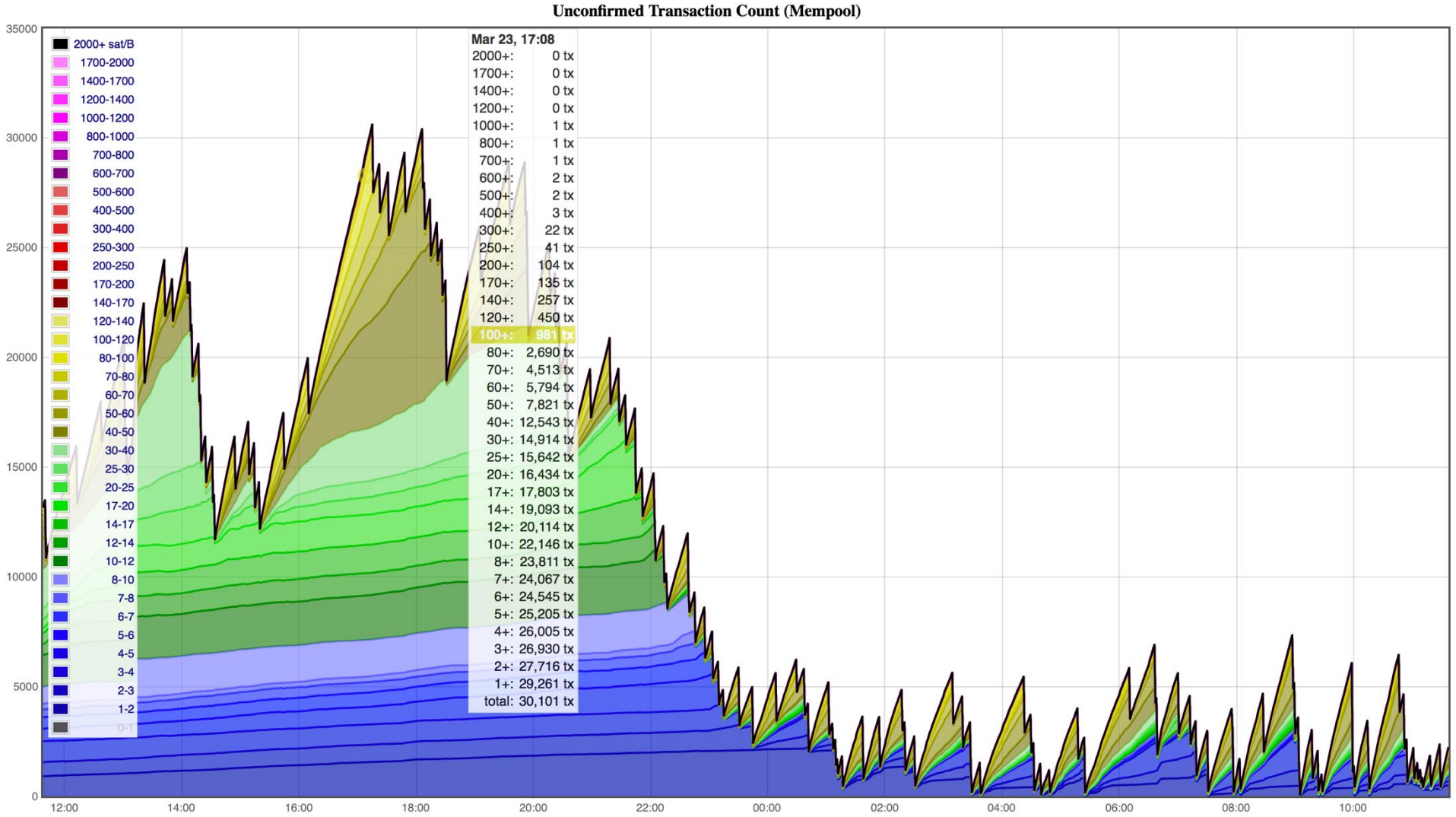
https://bitcoinfees.earn.com

How to choose the right fee?





Mempool status



https://jochen-hoenicke.de/queue/#0,24h

Ordering transactions

Transactions are ordered by their **fee-per-kilobyte ratio** in descending order and chosen from this list.

This is the current implementation in Bitcoin Core, but of course miners are free to tweak this implementation or to use other mining software with a different algorithm.

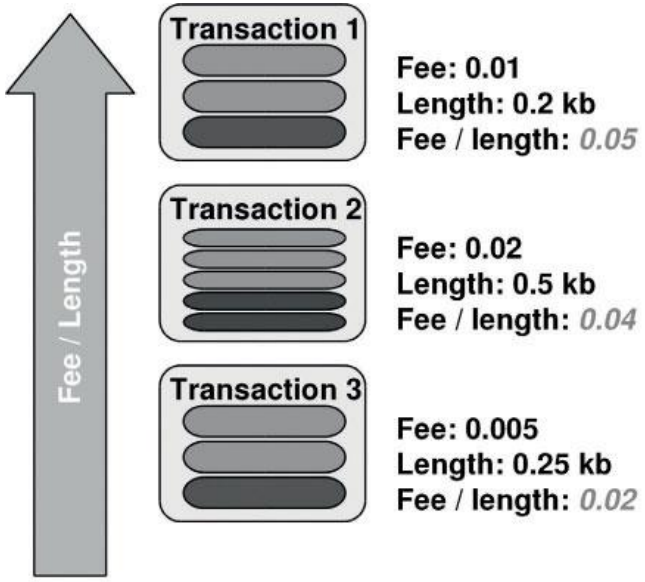


FIGURE 9.7 Prioritization of transactions by miners

Franco, P. (2014). Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons.

Ordering transactions

There is a **minimum transaction fee** in the protocol. The goal of this fee is to prevent denial-of-service attacks that could flood the network with transactions that pay no fee. The minimum transaction fee is currently set to 1,000 satoshis or 0.00001.

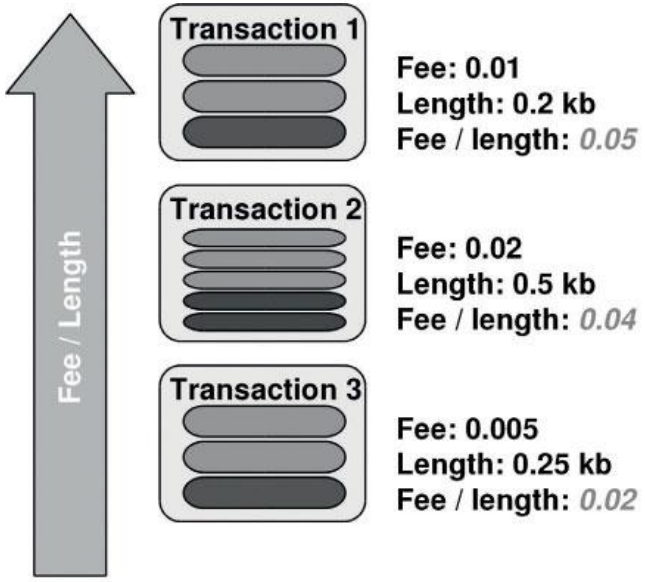
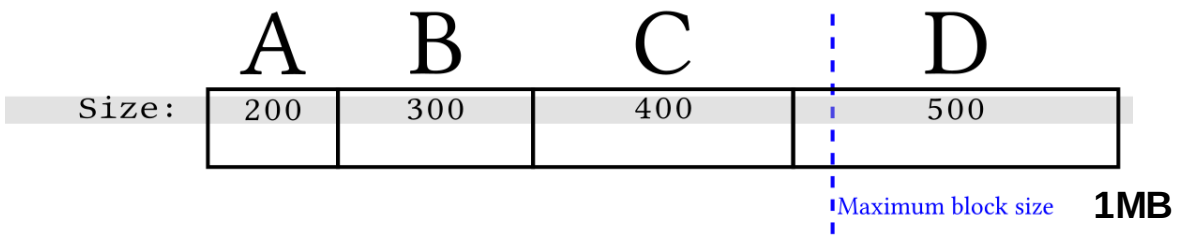


FIGURE 9.7 Prioritization of transactions by miners

Franco, P. (2014). Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons.

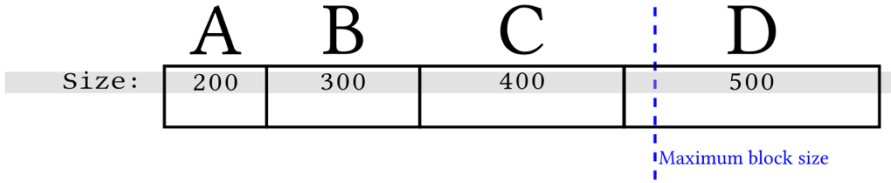
Fee rate = fee per kb ratio



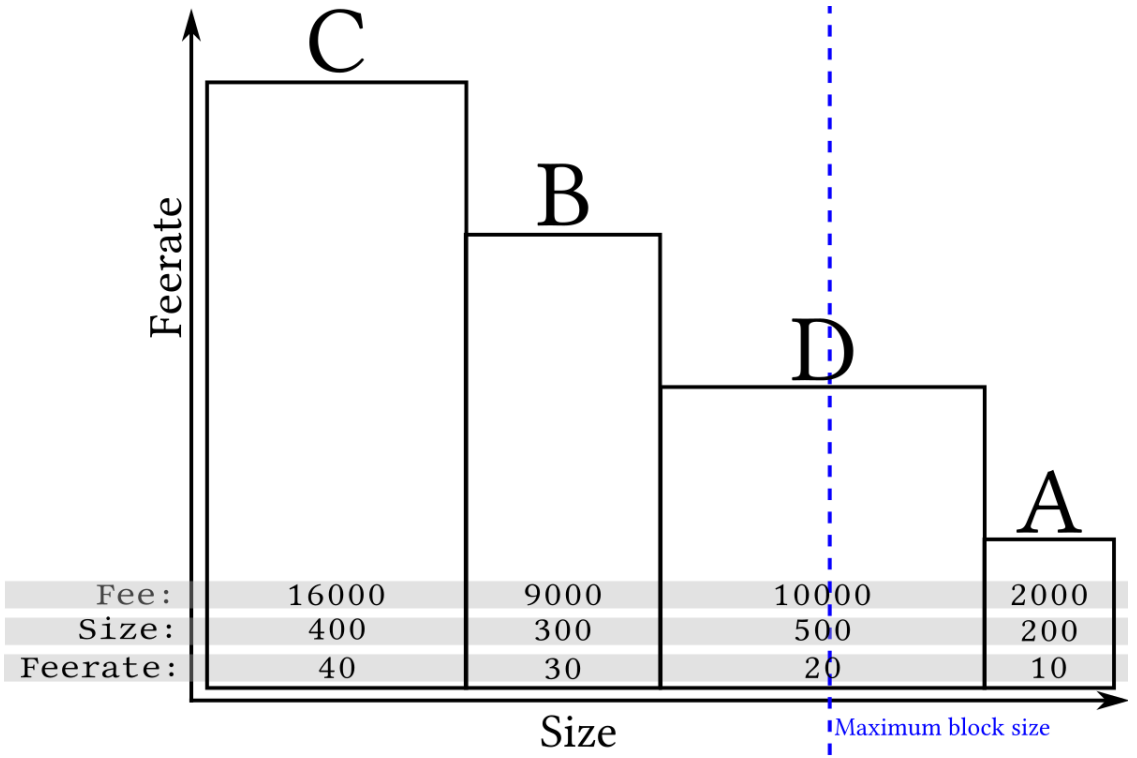
Which transactions should we fit in the block to maximise revenues?

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

Fee rate = fee per kb ratio



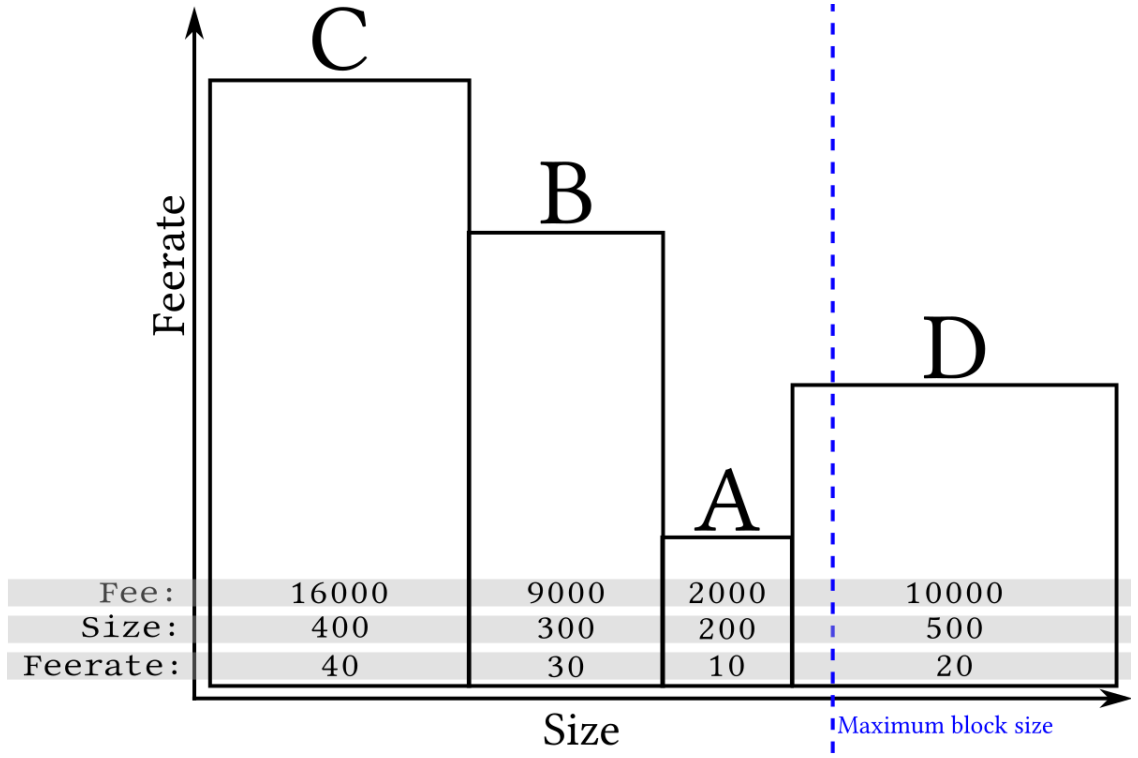
Although long (wide) transactions may contain more total fee, the high fee rate transactions are the most profitable to mine because their area is greatest compared to the amount of space they take up in a block.



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

Fee rate = fee per kb ratio

Sometimes it may be more convenient to leave some space for smaller transactions…



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

Transaction priority

Historically it was not required to include a fee for every transaction.

A large portion of miners would mine transactions with no fee given that they had enough "priority". Every block had some space reserved, called the **priority block**, for priority transactions.

```
Priority = Sum (Value of input * Input Age) / Transaction Size
```

Transactions needed to have a priority above **57,600,000.** For instance:

```
High Priority > 100,000,000 satoshis * 144 blocks / 250 bytes = 57,600,000
```

Today, low priority is mostly used to detect spam transactions and almost all miners expect every transaction to include a fee

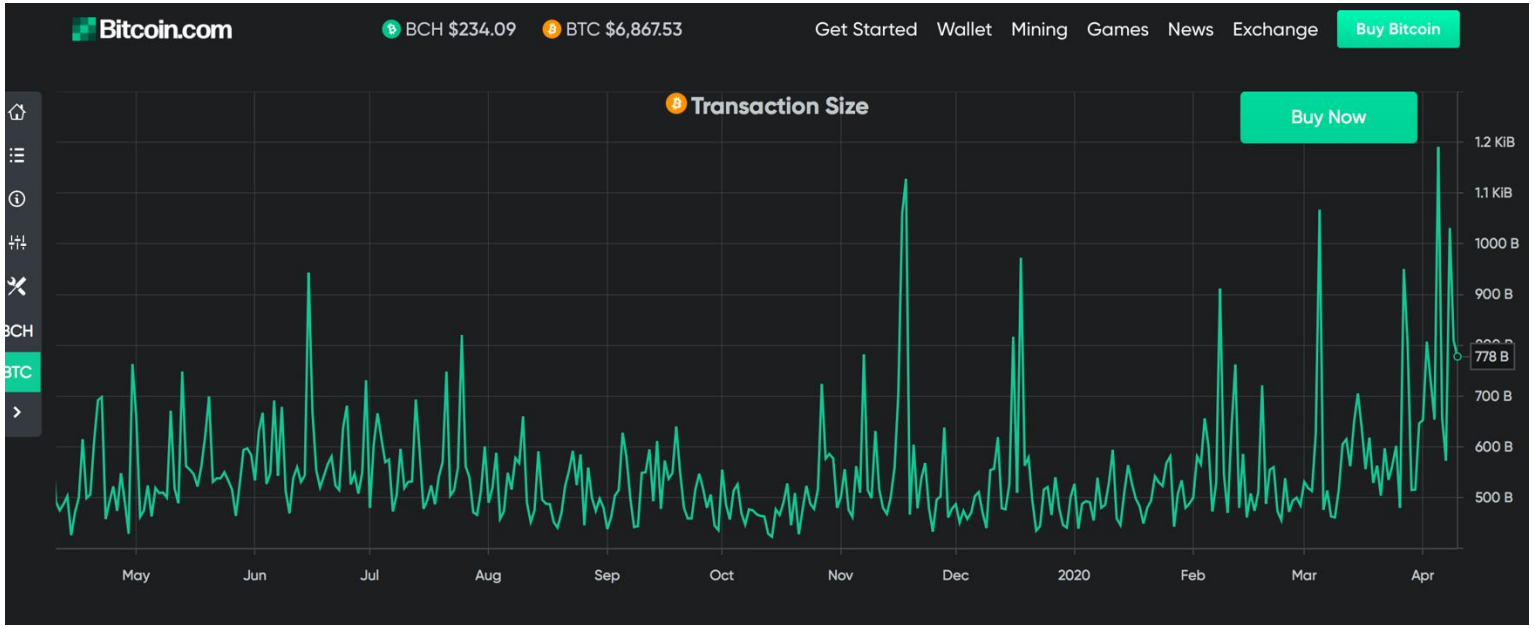
Antonopoulos, A. M. (2014). *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.".

Transaction size

- The size of a transaction depends on the number of addresses that the transaction draws funds from and on the number of addresses that the funds are sent to.

- The size of a transaction with only one input and one output is in the order of **157 bytes**.

- Each additional input to a transaction adds **113 bytes** to its size, while an additional output adds 34 bytes.



Recap exercise – transaction fees

A Bitcoin miner Mempool contains the following eight transactions. The maximum block size is set to 700 Bytes**.** Which transactions will the miner select to be included in the next block and why?

	Tx1	**Tx2**	**Tx3**	**Tx4**	**Tx5**	**Tx6**	**Tx7**	**Tx8**
Size (in Bytes)	130	200	400	150	70	350	170	180
Fees (in Bitcoins)	1	0.5	1.5	2	1.3	2.5	2.3	0.7

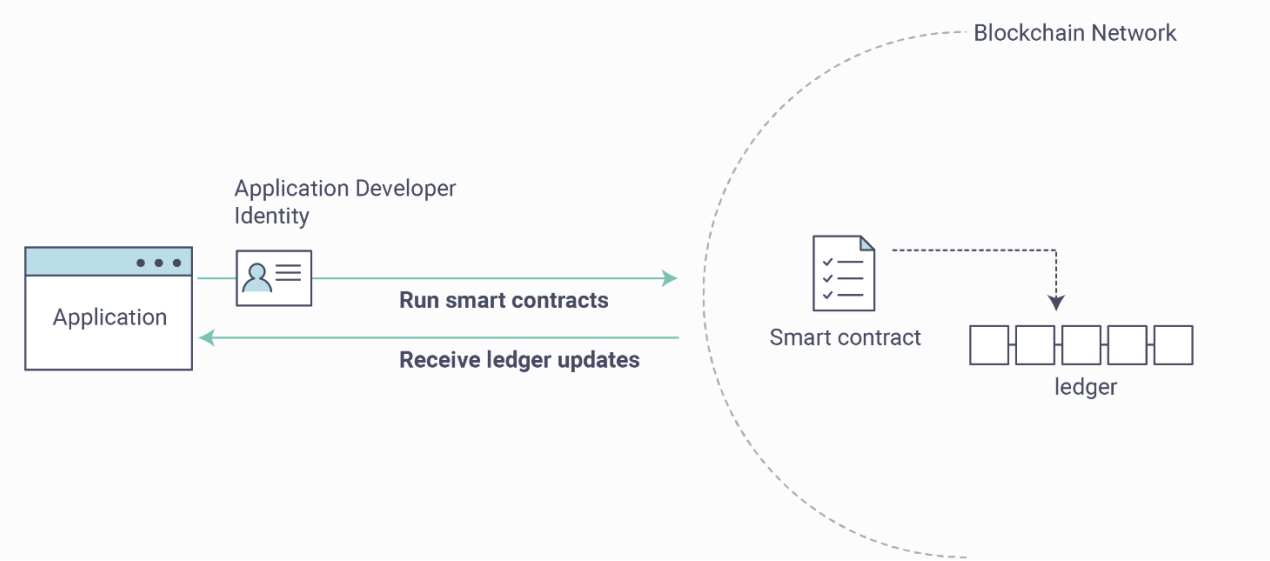
What is a (smart) contract?

A contract is a legally-binding agreement which governs the rights and duties of the parties to the agreement. A contract is legally enforceable because it meets the requirements and approval of the law. An agreement typically involves the exchange of goods, services, money, or promises of any of those. In the event of breach of contract, the law awards the injured party access to legal remedies such as damages and cancellation. A contract generally requires an offer, acceptance, consideration, and a mutual intent to be bound.

Smart contracts: Smart contracts are self-executing digital contracts between peers immutably stored on a blockchain. They are automatically executed when algorithmically enforced predetermined terms and conditions are met.

Language purpose

- Blockchain systems allow to modify the conditions under which certain information (e.g. transactions) will be included into the public record.
- These conditions must be specified in an algorithmic manner *(smart contracts).*
- These algorithms are run in a *scripting language* designed purely for this purpose.
- The flexibility given to the users impacts the ability to create conditions for certain actions to occur and the hypothetical computational effort to verify the conditions included in the contracts.



Source:Hyperledger

Programming languages

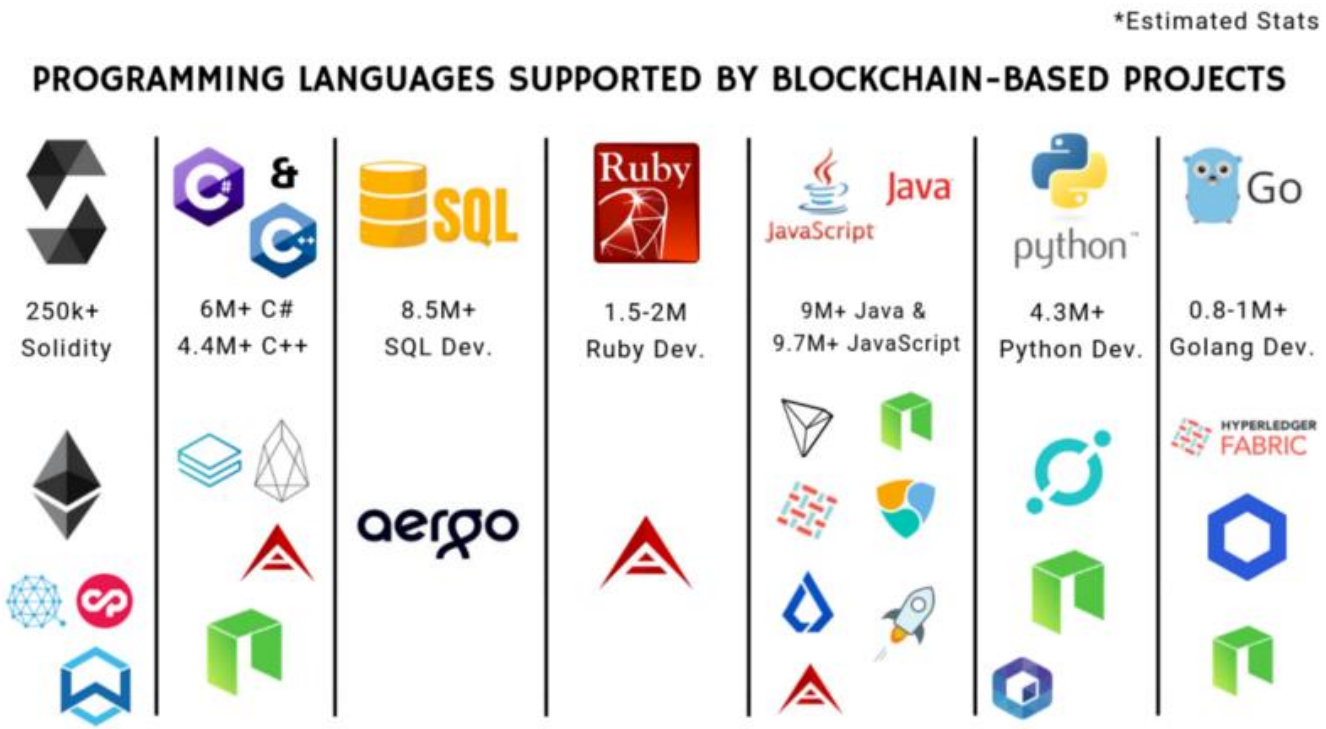
Conditions on how to spend tokens, enforce smart contract clauses are programmed using different programming languages offering diverse functionalities.

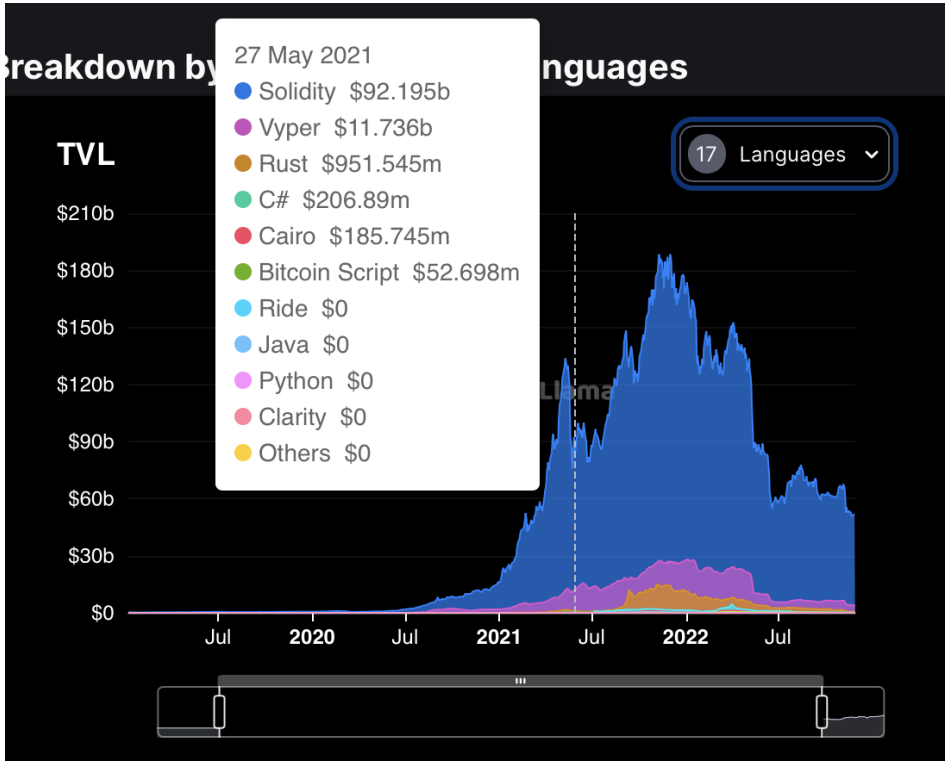
- Turing complete languages are general purpose languages, in which arbitrary computations can be performed.

- Non-Turing complete languages have scripts with limited complexity and predictable execution times.

- Languages that are not of this kind, are so because of design reasons which aim at prevent specific behaviors of code execution, like undefined termination.

Smart contract languages





https://www.freecodecamp.org/news/the-most-popular-programming-languages-used-in-blockchain-development-5133a0a207dc

https://defillama.com/languages

Locking script

<Public Key A> OP_CHECKSIG

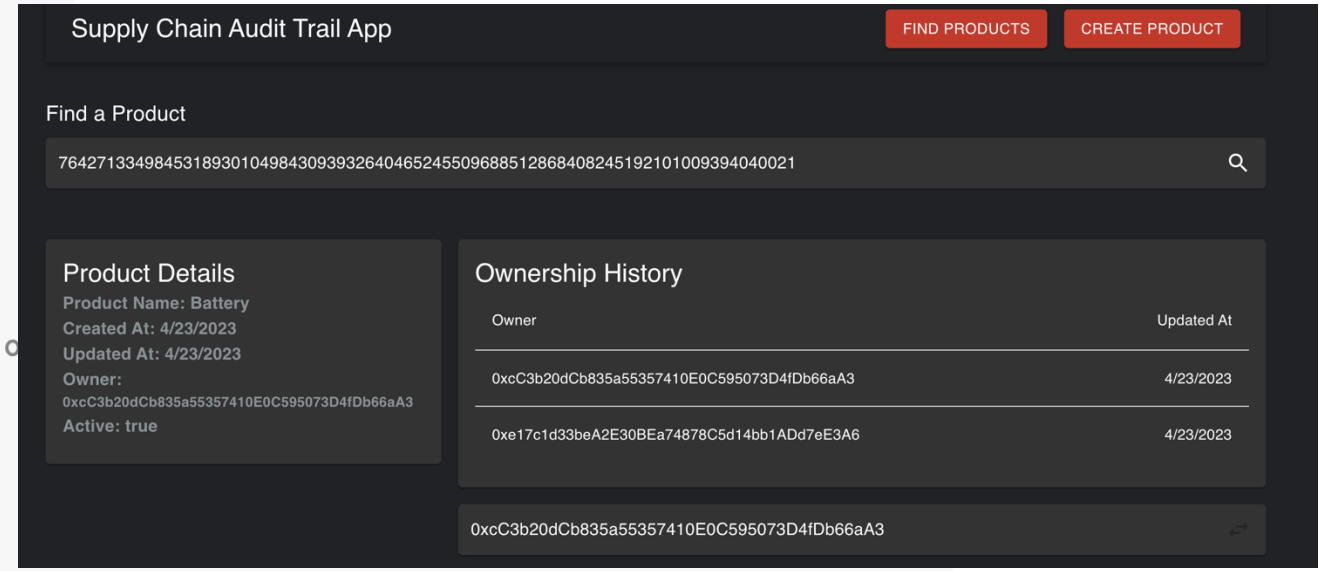
Unlocking script

<Signature from Private Key A>

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

```
contract SupplyChain {

    // Map of product IDs to product information
    mapping(uint => Product) public products;

    // Struct to represent a product
    struct Product {
        // Name of the product
        string name;
        // Timestamp of when the product was created
        uint createdAt;
        // Timestamp of when the product was last updated
        uint updatedAt;
        // Current owner of the product
        address owner;
        // Is the product active
        bool isActive;
    }

    // Event to be emitted when a product transfers o
    event ProductOwnershipTransfered(
        uint indexed id,
        string name,
        address newOwner,
        uint updatedAt
    );

    // A modifier that only allows the product owner to interact with a specific function
    modifier onlyProductOwner(uint _id) {
        Product storage product = products[_id];
        require(product.owner == msg.sender, "Function can only be called by product owner!");
        _;
    }
```

Create - view - track ownership of products



https://github.com/cabeywic/supply-chain-contract

```
daml 1.2 module Main.CWspace where
import Main.DataTypes
import Main.Cash

template CWspace
    with
        price: Decimal
        currency: Currency
        manager: Party
        tenant: Party
        loc_aut: Party
        address: Text
        tenancy_length: Decimal
        square_meters: Decimal
        description: Text

    where
     signatory manager
     observer loc_aut

     controller manager can
        UpdateListing: (
            ContractId CWspace
            )
            with
                newTenant : Party
                newPrice: Decimal
                newtenancy_length : Decimal
            do
                create this with
                    tenant=newTenant
                    price=newPrice
                    tenancy_length = newtenancy_length

        RemoveListing: ()
            do
                assert(
                    manager == tenant
                    )
                return()
```

Next week..

- P2P network
- Scalability
- Scripting Language
- Protocol’s improvements & forks