Bitcoin Evolution and Challenges

Öz, B., Hoops, F., Gebele, J., Gallersdörfer, U., & Matthes, F. (2024). “Blockchain-based Systems Engineering”. Lecture Slides. TU Munich.

Chair of Software Engineering for Business Information Systems (sebis)
Department of Computer Science
School of Computation, Information and Technology (CIT)
Technical University of Munich (TUM)
wwwmatthes.in.tum.de

Outline

1. Evolution of the Bitcoin Network
- Protocol Update
- Design
- Signaling
- Potential Results
- SegWit and Taproot

2. Attacks on the Bitcoin Network
- Double Spending Attack
- Replay Attack
- 51% Attack
- Selfish-mining Attack

3. Limitations & Challenges of Bitcoin
- Transaction Throughput
- Energy Consumption

Evolution of the Bitcoin Network

As any other software, blockchains also require **updates**.

These updates affect two parts of the network:

- The **software relying on full nodes** (wallets, etc.)
- The **blockchain network** (the full node implementations)

Considering wallets and other software, updates have well-known issues.

1. Incompatibility between old and new software components
   ➔ Old and new software components have to check the version available at runtime
2. Incompatibility between historic data and current data schema expected by the software components
   ➔ Database schema changes and data migration

Therefore, the Bitcoin network inherits these standard problems.

Additionally, the immutable blockchain data structure and the decentralized P2P-network lead to evolutionary issues. ➔ Process for protocol update

Process of Bitcoin Protocol Update

1. Design

- Layers
- Hard fork
- Soft fork

The proposal targets different layers, possibly resulting in a hard fork or a soft fork.

2. Signaling

- Miners vote

Once a proposal is final, the mining community votes on the proposal.

3. Results

- Acceptance
- Rejection
- Chain split

Depending on the votes, the proposal is implemented, rejected, or creates a chain split.

1. Design

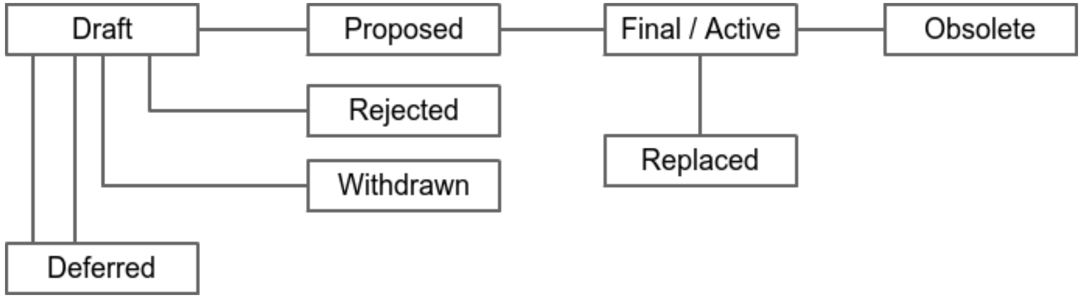
- Proposals can target different layers:
  - Consensus Layer (how to validate states and history)
  - Peer Services Layer (propagation of messages)
  - API/RPC Layer (high-level calls accessible by apps)
  - Applications Layer (high-level structures)

- All proposals in Bitcoin are referred to as **Bitcoin improvement proposals (BIP)**.

- A Github-repository maintained by the core-developers contains all BIPs.

- A BIP contained in the repository is not automatically accepted. Furthermore, the miner community decides whether a BIP is implemented. (through **signaling**)

- A final BIP contains a detailed description as well as a **reference implementation**. Developers of different clients should be able to adopt the BIP.



Possible BIP status paths.

Find out more: https://github.com/bitcoin/bips
Process of creating BIPS and status changes https://github.com/bitcoin/bips/blob/master/bip-0002.mediawiki
Different layers in BIPS https://github.com/bitcoin/bips/blob/master/bip-0123.mediawiki

1. Design (cont.)

The terms hard fork and soft fork describe changes within the **consensus layer**.

- **Hard fork**[1]: Structures, that are invalid under old rules become valid under new rules.
- **Soft fork**: Some structures, that were valid under the old rules are no longer valid under the new rules.
- Other changes applying to other layers are not classified as a hard fork or a soft fork. E.g., if a new RPC/API-call is introduced, the consensus layer is not affected.

Examples

The Bitcoin core client specifies a maximum block size of 1MB.

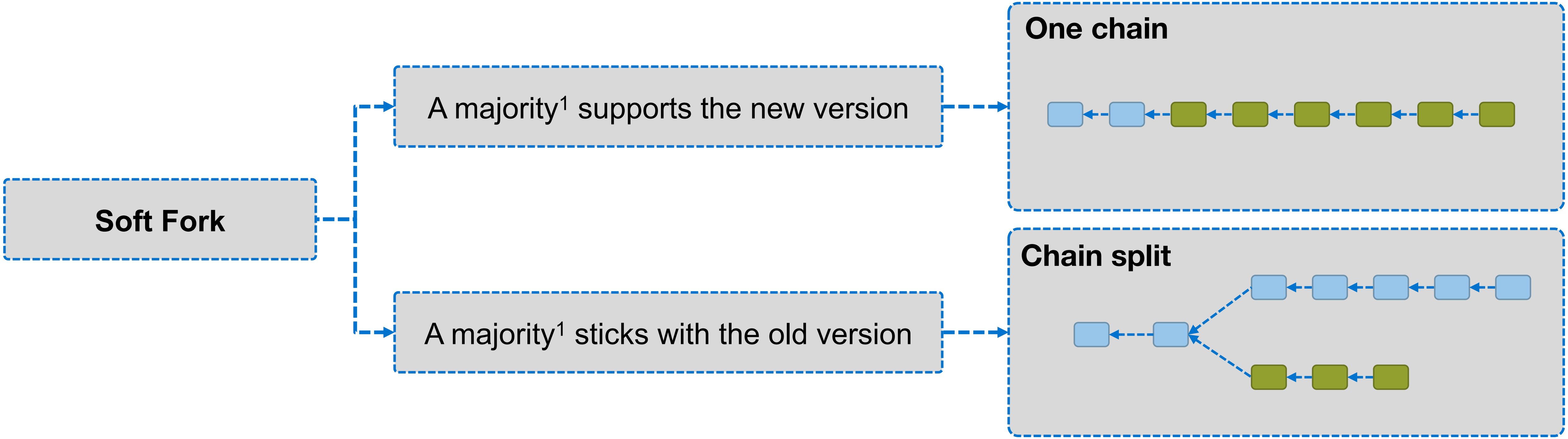
- An update enabling block sizes up to 8MB is considered a **hard fork**.
- An update restricting block sizes up to 0,5MB is considered a **soft fork**.

Why do we need signaling?

[1] *Please note, that we use the definition of a hard fork from BIP123. The creation of new chains sharing history is considered as chain split. https://github.com/bitcoin/bips/blob/master/bip-0123.mediawiki*

Why do we need signaling?

- The following diagram shows that a soft fork that is not accepted by the majority of miners leads to two chains.



- The disadvantage of a chain split is that both chains compete for users.
  - A split could be the goal of the designer or not.
- To find out whether a chain split would occur, **signaling** is used.

[1]The majority of the hash power in the overall network.

2. Signaling (cont.)

Signaling is the process that is used for voting on proposals.

Miners are the only network participants who can cast their votes. They gain the right to vote only when they mine a new block. To signal their vote, they include special values in the header of the blocks they mine.

How does it work?

- The **version** field in the block header contains 32 bits.
- Each proposal author (reference implementation) selects a bit in this version field, a start, and an end time.
- If miners update, their node software uses the bit to signal the support of this miner for this proposal between start time and end time.

- **Option A**: The overall support for this proposal is higher than a certain threshold (usually 1916 out of 2016 blocks ➔ 95%) in one difficulty period. The proposal is **accepted (locked in)**, and the rules apply after further 2016 blocks to allow the remaining miners to update as well.

- **Option B**: The overall support for this proposal is lower than the threshold until the end time. The proposal is **rejected**.

- As of the signaling, miners can vote up to 29 different proposals at the same time. Bits are reused after the end time.

3. Results

- If the signaling leads to **acceptance**, the proposal is automatically implemented.
- Upon a rejection, the community can **split** (separate the development and go into different directions).

➔ This is called a “Chain split”

Obviously, both communities want to keep the history.

Two main problems arise with the creation of a second chain:

1. If the community behind the hard fork has less than 50% of the computational power, the new chain will not work, as the new software will switch back to the old chain, as the old chain will probably have the higher weight.
2. The same transaction can be executed on both chains ➔ **Replay attacks**

To prevent both problems, the community has to make the **new chain/software incompatible with the old chain**. This is done via the creation and adaption of new parameters, such that the old chain is not accepted by the new software and the other way round. Another possibility is to define a second “genesis” block which has to be contained in the longest chain. If the second block contains the new rules (rejected by the old software), the chains are split indefinitely and can not merge together.

3. Results (cont.)

- There have been numerous successful Bitcoin soft forks like *Pay to Script Hash* (P2SH).
- We are not aware of a successful Bitcoin hard fork.
- Some chain splits were executed with *varying* success, e.g.,
  - Bitcoin Cash[1] (8MB blocks instead of 1MB blocks)
  - Bitcoin Gold[2] (changes in the PoW-algorithm for ASIC-resistance)

[1]Bitcoin Cash blocks are on average smaller than the blocks of Bitcoin, see https://thenextweb.com/hardfork/2019/01/17/bitcoin-cash-block-size/
[2]Bitcoin Gold was subject to a 51%-attack, see https://bitcoinist.com/51-percent-attack-hackers-steals-18-million-bitcoin-gold-btg-tokens/

SegWit and Taproot

- **Segregated Witness** (SegWit) (2017) and **Taproot** (2021) are two major upgrades that occurred on the Bitcoin network. Both are **soft forks**.

SegWit

- Aimed to increase the transaction throughput by reducing the weight of transactions in a block
- Segregates the transaction signature into a separate **witness** component
- Introduced the concept of weight units (WU)
  - Block size limit (1,000,000 bytes) is replaced with a block weight limit (4,000,000 WU)

$$\boldsymbol{weight(tx) = 4 * (tx_{non\_witness\_bytes}) + 1 * (tx_{witness\_bytes})}$$

- A SegWit transaction's signature weighs a quarter of regular bytes as it is counted as witness bytes
- A non-SegWit transaction has **no witness bytes**; thus, its weight is equal to 4x its size

Taproot

- Introduced a new signature scheme, the Schnorr signatures (replaces ECDSA), which enables batch verification of aggregated signatures (e.g., in multi-sig wallets), increasing transaction processing speed
- Improves upon privacy as multi-sig transactions and single-sig transactions cannot be distinguished anymore

Outline

1. Evolution of the Bitcoin Network
   - Protocol Update
   - Design
   - Signaling
   - Potential Results
   - SegWit and Taproot

2. Attacks on the Bitcoin Network
   - Double Spending Attack
   - Replay Attack
   - 51% Attack
   - Selfish-mining Attack

3. Limitations & Challenges of Bitcoin
   - Transaction Throughput
   - Energy Consumption

Attacking the Consensus Mechanism

Is it possible to steal Bitcoins?

No: Since UTXOs are secured with the hash of the public key of a user[1], the attacker cannot generate a valid transaction spending these UTXOs.

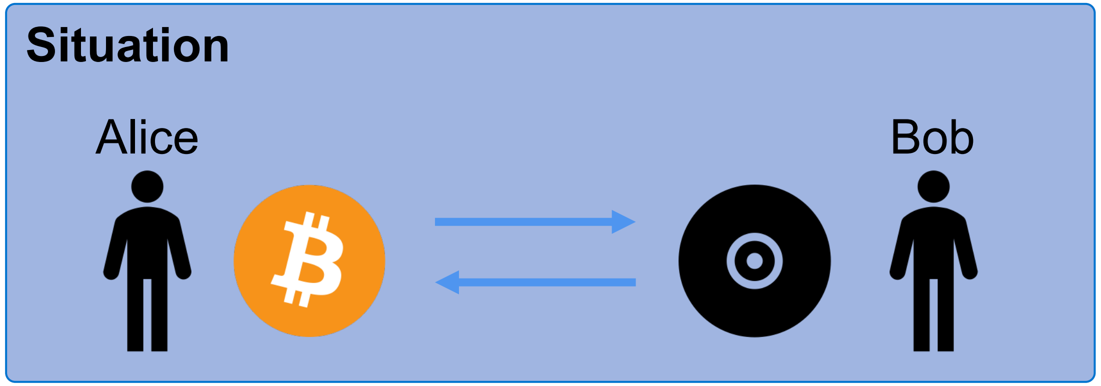
Is blocking a participant (Wallet Owner) in the blockchain network possible?

Assume that a malicious node wants to block all transactions by Bob and it won the mining puzzle. The node can choose to not include any transactions from Bob in its block. However, the next random node (if it is honest) will include Bob's transactions. Therefore, it is almost impossible to completely censor out someone.

[1] The script of the UTXO can also define other requirements for spending.

Double Spending

The idea of digital cash did evolve around the idea that we need to prevent a double spend. Two transactions spending the same Txout is somehow hard, but not impossible. We will go into the details of this attack.



Alice wants to buy a music file from Bob's online shop with Bitcoin. She creates a transaction which sends the Bitcoins to Bob. An honest node sees Alice's transaction and includes it in its block. Bob sees the new block and sees his transaction included in it. He sends the file to Alice, in good faith to have his money.

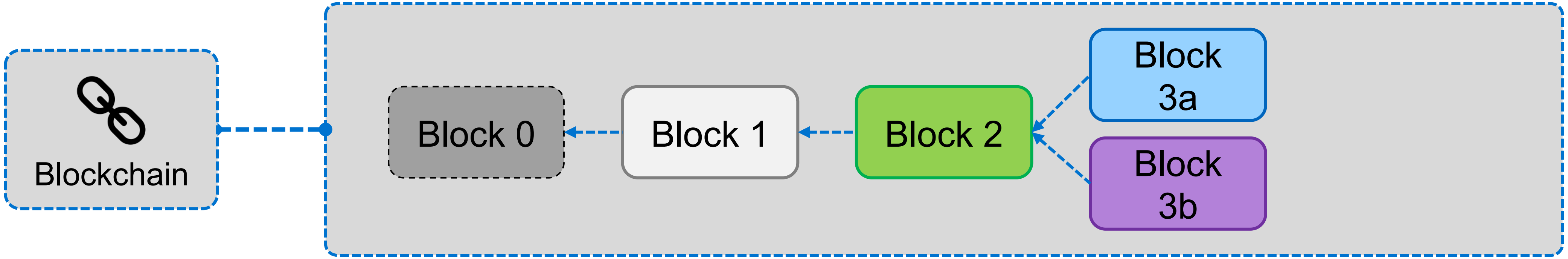
So far so good: What are the options for Alice to "double spend" the Bitcoins she sent to Bob?

Double Spending (cont.)

Take a look at the underlying blockchain:

- The blue block (block 3a) contains Alice valid transaction to Bob.
- After an honest node proposed this block, Alice was selected to propose the new block.
- What can she do?
  - Option 1: Build on top of block 3a, she accepts the fact that the transaction has happened. This is not what she wants, she wants to double spend!
  - Option 2: Build on top of block 2 a new block 3b (purple), not containing the transaction she sent to Bob, but a transaction spending the same coins (she would have sent to Bob) to herself.[1] This is described as forking.

➔ Standard double-spending pattern



[1]She has to create a transaction spending the UTXO, as if not, the transaction would simply be included in a later block.

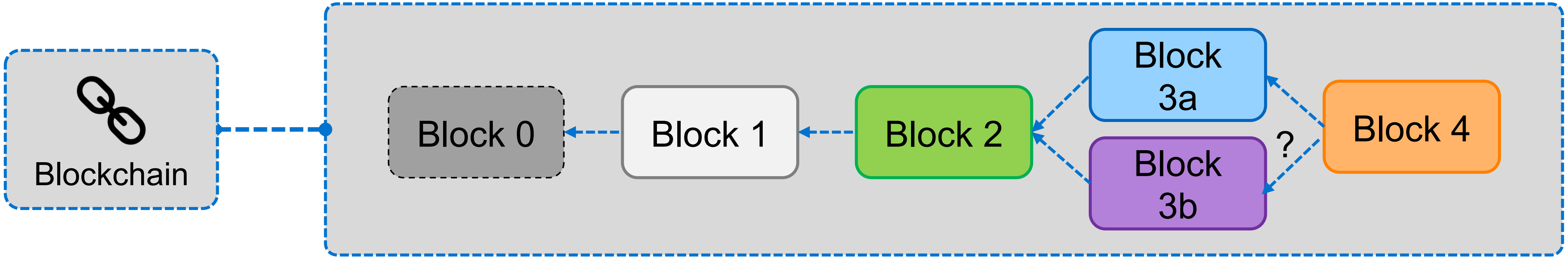
Double Spending (cont.)

Does this mean double spending is possible?

No. It is impossible to create a valid block or blockchain with two transactions consuming the same UTXO. What happens is that two “realities” are created. Block 3a (blue) declares a reality where Bob is paid and block 3b (purple) declares a reality in which Alice sends the money to herself.

How is this conflict resolved?

The next node that get selected proposing a new block resolves the issue. It has to select the block on which it wants to create its new block. As all nodes adopt the longest chain, one reality (one of the blocks 3) is “orphaned”, meaning this block does not have any relevance to the network anymore.



Double Spending (cont.)

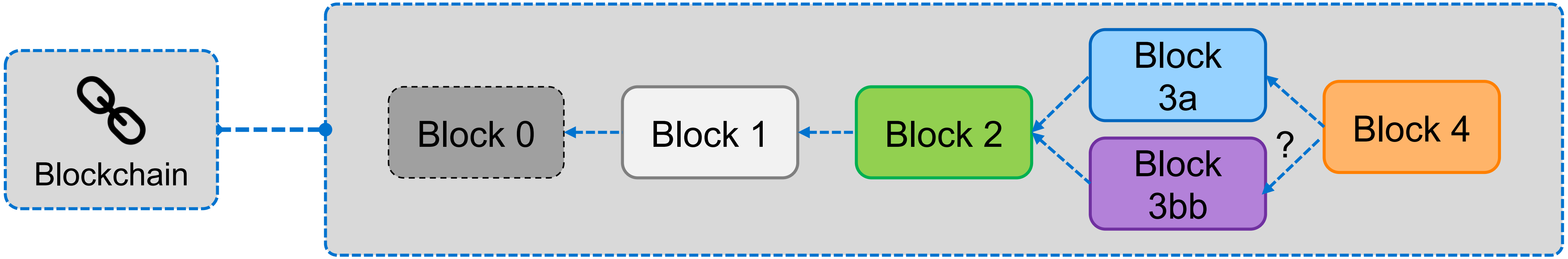
When is the attack successful?

The attack is successful, if Alice convinces the network that her block (purple block) is the valid block that should be included in the longest blockchain.

From our story, we know that the blue block is the “valid” block. From the perspective of an individual node, both blocks are equally valid.

What should Bob do to prevent such an attack?

Bob should wait until it is clear that the payment to him is actually included in the longest blockchain, ideally with several confirmations (blocks on top of the block containing his transaction) before sending the file.



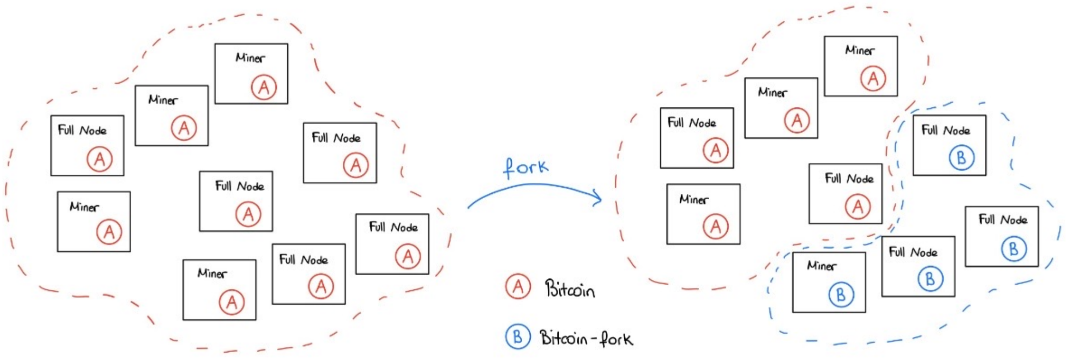
Replay Attack

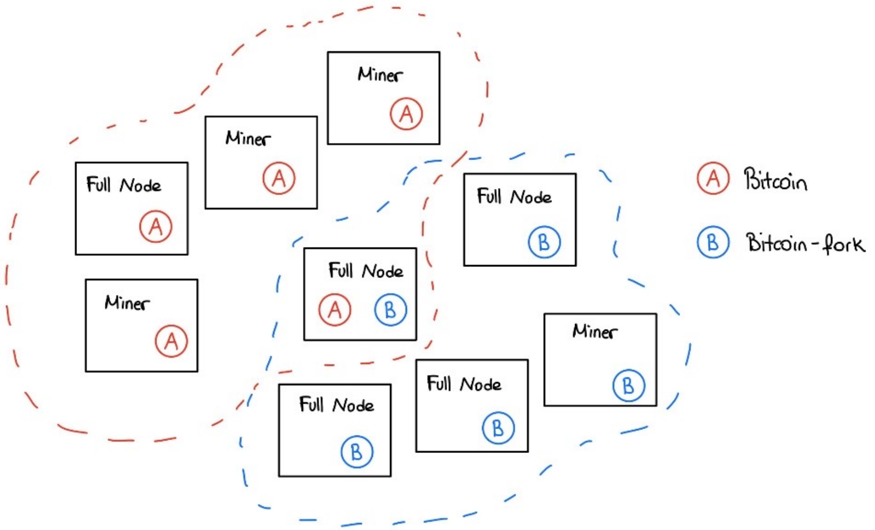
A replay attack occurs when an attacker re-submits the same transaction to the network multiple times and successfully executes it each time.

Example Scenario

1. A **hard fork** happened on Bitcoin and now Alice owns same amount Bitcoins on both networks.
2. After the split, Alice sends 5 Bitcoins to Bob on the legacy Bitcoin network to settle a debt.
3. Bob realizes he can receive additional coins by replicating Alice's transaction on the new chain because addresses remain the same.
4. Miners on the new blockchain validate the replicated transaction, enabling Bob to successfully execute a replay attack.

- *Someone who joins the network after the hard fork is **not vulnerable** to the replay attack as their address has no transaction history in either of the chains.*





51% Attack

A 51% attack is the **worst possible scenario** in a blockchain. It means that more than 50 % of the hash power belongs to one entity and this entity uses this power maliciously.

The attack enables:

- **History rewriting**: The attacker can build a blockchain with the highest accumulated value, defining all contents:
  - Blocking / DoS-ing addresses / users
  - Collecting all mining rewards
  - Creating successful double-spending patterns (orphaning many blocks)

- However:
  - Cannot invent money, cannot propose invalid blocks or transactions, as they would simply be rejected.
  - As of the high hash power, the attacker is highly invested.
  - Entities highly invested have no interest in destroying the network, as they profit the most from it.

➔ **A successful executed 51 % attack would destroy the trust in the system and with that, the value of the currency in the system.**

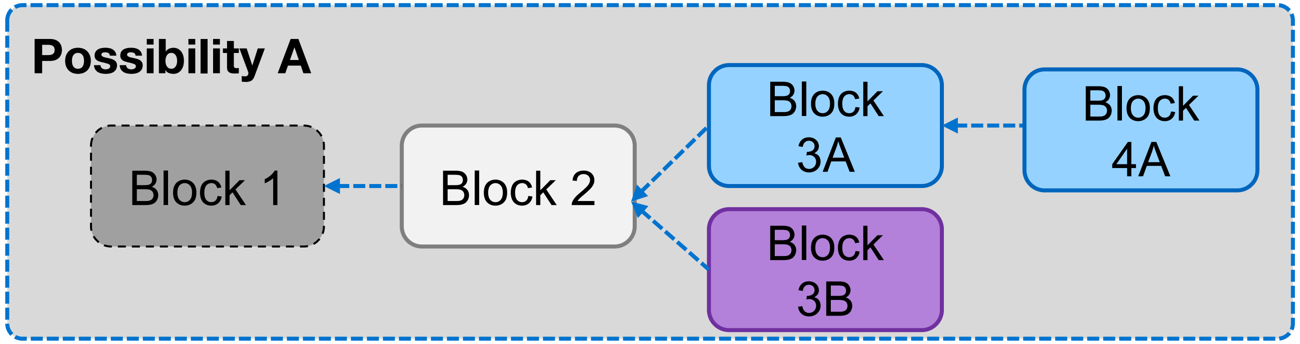
Selfish-mining Attack

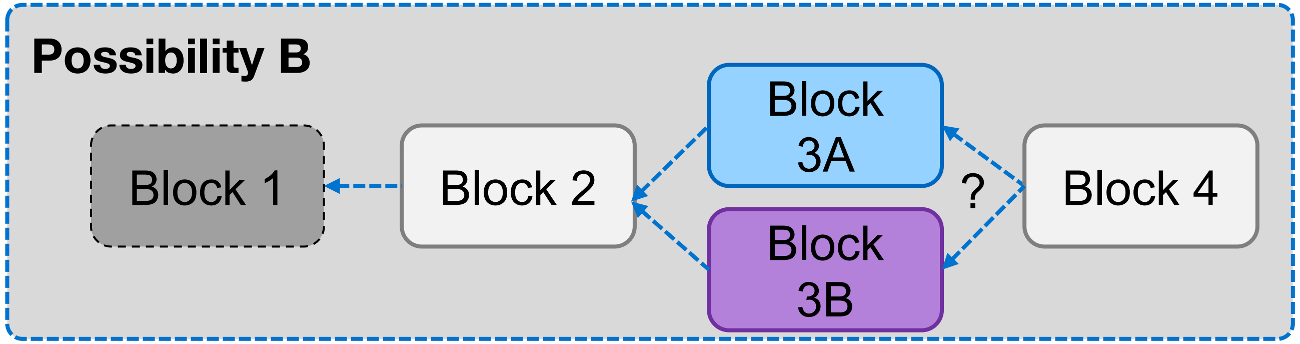
A selfish-mining attack exploits the probability of being able to propose two blocks one after another.

It works as follows:

- The attacking node finds a new block 3A, but does not propose it to the network
  - **Possibility A**: The node finds a second block 4A building on its block 3A. The network is still at block 2. When the network finds another block 3B, the attacker publishes both blocks 3A and 4A, making the new 3B an orphan block. The network has worked on an old chain, practically wasting its power.
  - **Possibility B**: When the network proposes block 3B before the attacker finds block 4A, the attacker publishes 3A, hoping the network will select block 3A with probability α.

➔ Attack is possible for hashing power minimum of 25% with α = 50% and 33% with α = 0%.





Can only be used to increase profits. Has not been observed in practice.

If $\alpha = 100\ \%$, what is the minimum hash rate needed to execute this attack?

Eyal, Ittay & Sirer, Emin. (2013). Majority Is Not Enough: Bitcoin Mining Is Vulnerable. 8437. 10.1007/978-3-662-45472-5_28.

Outline

1. Evolution of the Bitcoin Network
- Protocol Update
- Design
- Signaling
- Potential Results
- SegWit and Taproot

2. Attacks on the Bitcoin Network
- Double Spending Attack
- Replay Attack
- 51% Attack
- Selfish-mining Attack

3. Limitations & Challenges of Bitcoin
- Transaction Throughput
- Energy Consumption

Transaction Throughput

The current technology and protocol introduce a **theoretical maximum transaction throughput** determined by three factors:

block time **10 minutes** *and* max block size/weight **1 MB = 4 MWU**

and transaction size **250 bytes** → $576{,}000\,\frac{\text{transactions}}{\text{day}}$

or

and transaction size **500 bytes** → $288{,}000\,\frac{\text{transactions}}{\text{day}}$

- With SegWit, Bitcoin block size can **exceed 1MB**, as blocks are limited by weight units (WU) rather than raw byte size.
- SegWit transactions save WU by moving signature data to the witness component, reducing the overall weight of transactions.
- Non-SegWit transactions still occupy the same amount of space as before.

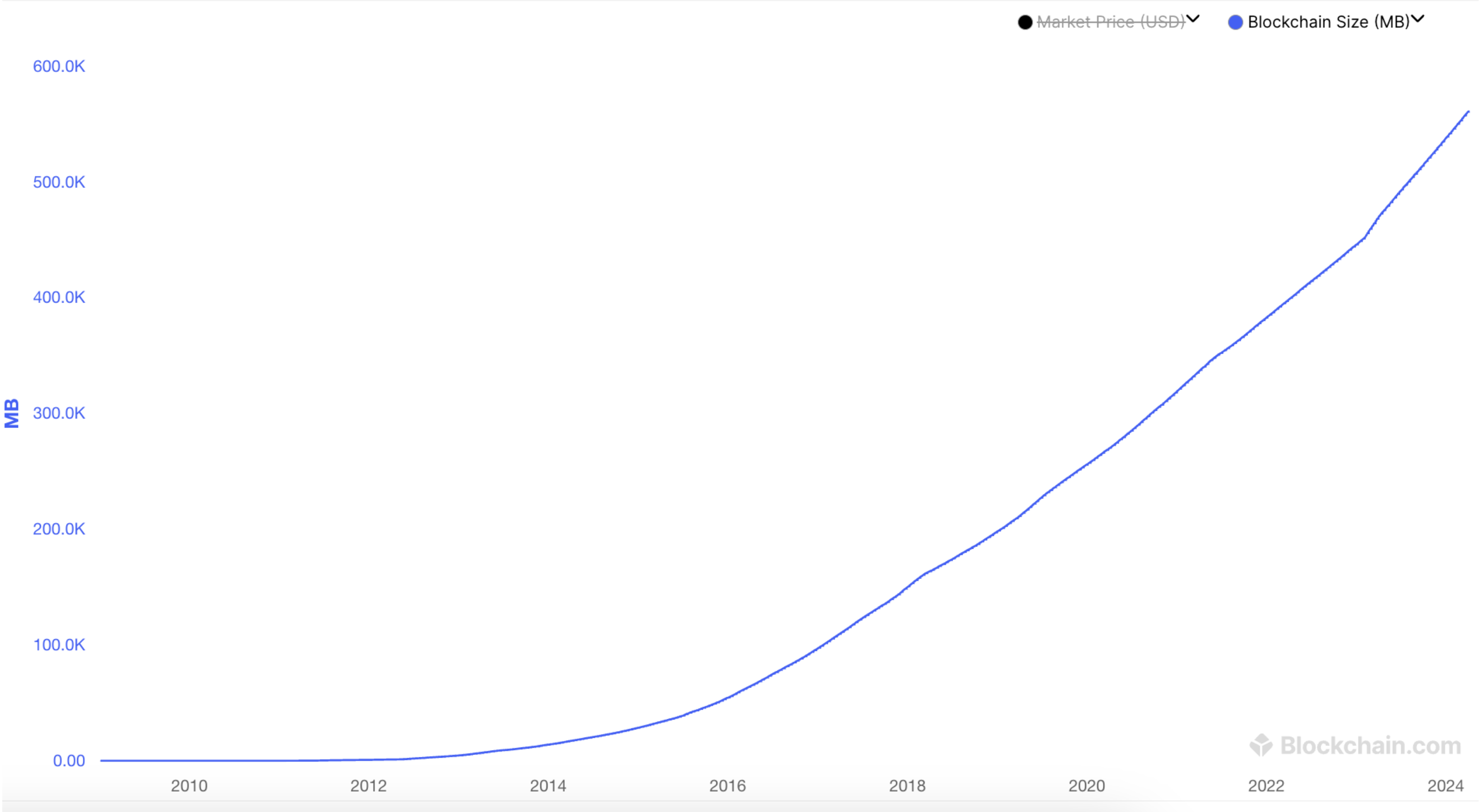
Screenshot taken from: https://mempool.space/

The recent throughput (April 7th, 2024) is 348,637 transactions per day. Source: https://ycharts.com/indicators/bitcoin_transactions_per_day

Croman, K. et al. (2016). On Scaling Decentralized Blockchains. In International Conference on Financial Cryptography and Data Security (pp. 106-125). Springer Berlin Heidelberg.

Bitcoin Blockchain Size

Bitcoin blockchain size increases approx. 1.7MB every 10 minutes.



If we would assume that the transaction throughput of Visa (150.000.000 per day), the Bitcoin blockchain would increase daily by:

$$150.000.000 \ * 250 \ bytes = \mathbf{37,5 \ GB}$$

The chart of the Bitcoin blockchain size: https://www.blockchain.com/charts/blocks-size

Energy Consumption of Bitcoin's Proof-of-Work

Difficulties:

- Miners do not disclose their energy consumption and the hardware used by them.
- We calculate with the hash rate and the most efficient mining hardware.

Speed of the network (April 2024):
62.01×10^{19} [hashes/s], i.e. 620.1 Exahashes (EH/s)

Average time for a block generation:
10 [minutes] = 600 [s]

Average hash tries per block:
$600 * 62.01 \times 10^{19} = 37.2 \times 10^{22}$ [hashes]

Power consumption of latest Antminer S21:
3500 [W] for 200×10^{12} [hashes/s]

Number of Antminer S21s to provide network speed:
$62.01 \times 10^{19} / 200 \times 10^{12} = 3{,}100{,}500$

Power consumed by these Antminers:
3,100,500 * 3500 [W] = 10.85 [GW]

More information https://m.bitmain.com/product/detail?pid=00020240311180613891frupBW6406B2, picture taken from the website
Bitcoin network hash rate: https://www.coinwarz.com/mining/bitcoin/hashrate-chart

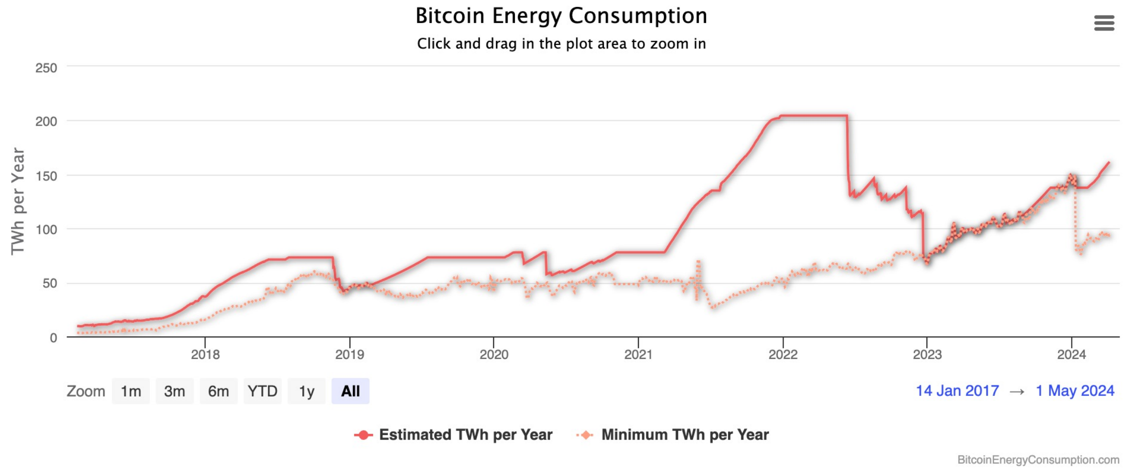
Energy Consumption of Bitcoin's Proof-of-Work (cont.)

- Energy consumption is one of the main issues of Bitcoin. Critics call it an energy guzzler, while supporters praise it for being less energy-intensive than the present global economy.

- The amount of energy that is consumed with Proof-of-Work cannot be underestimated, however, it is also not a reason to demonize the system when there is **no better solution with the same properties**.

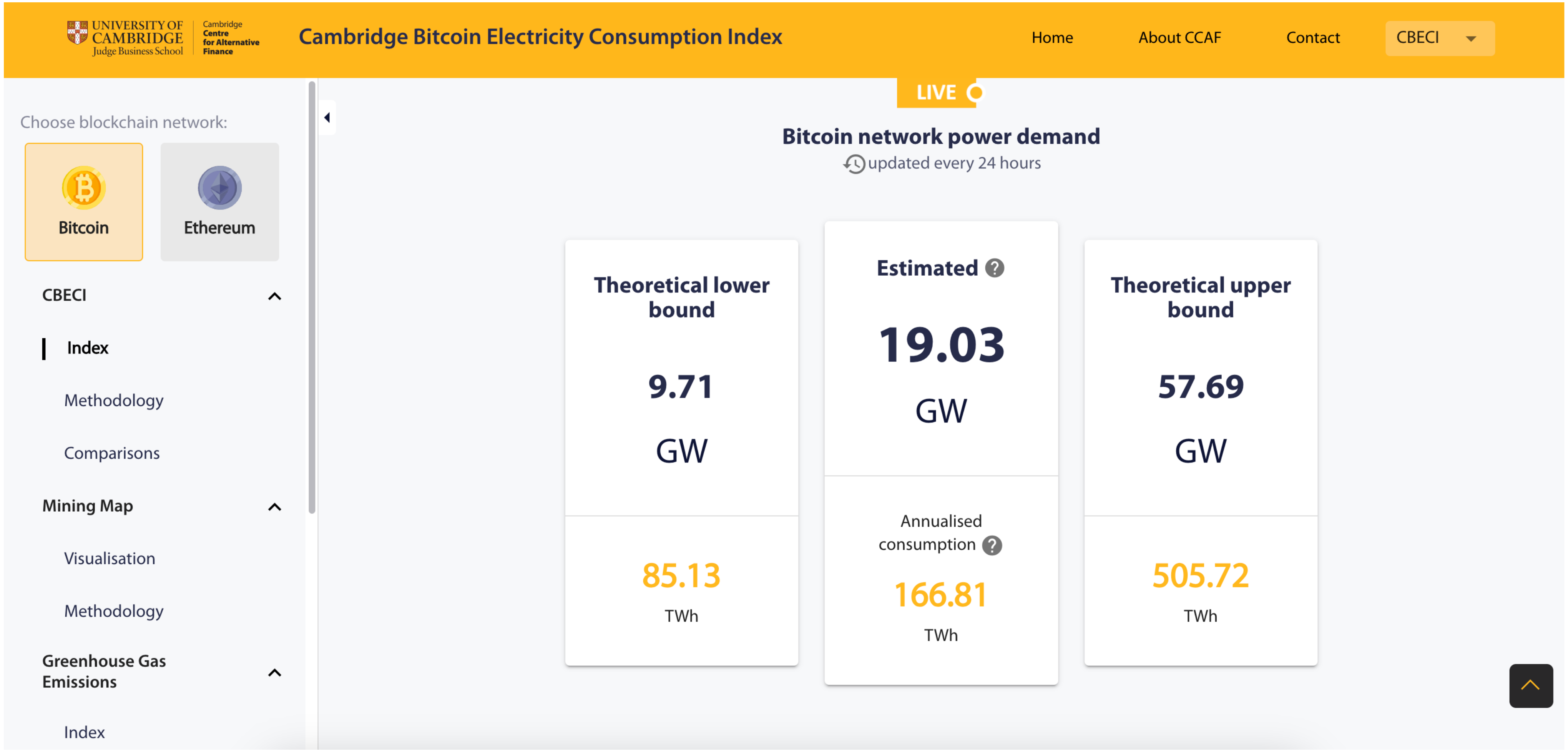
Annualized Total Bitcoin Footprints

Carbon footprint	Electrical energy	Electronic waste
90.17 Megaton (Mt) CO_2	161.67 TWh	29.30 kt
Comparable to the carbon footprint of Chile.	Comparable to the power consumption of Egypt.	Comparable to the small IT equipment waste of the Netherlands.



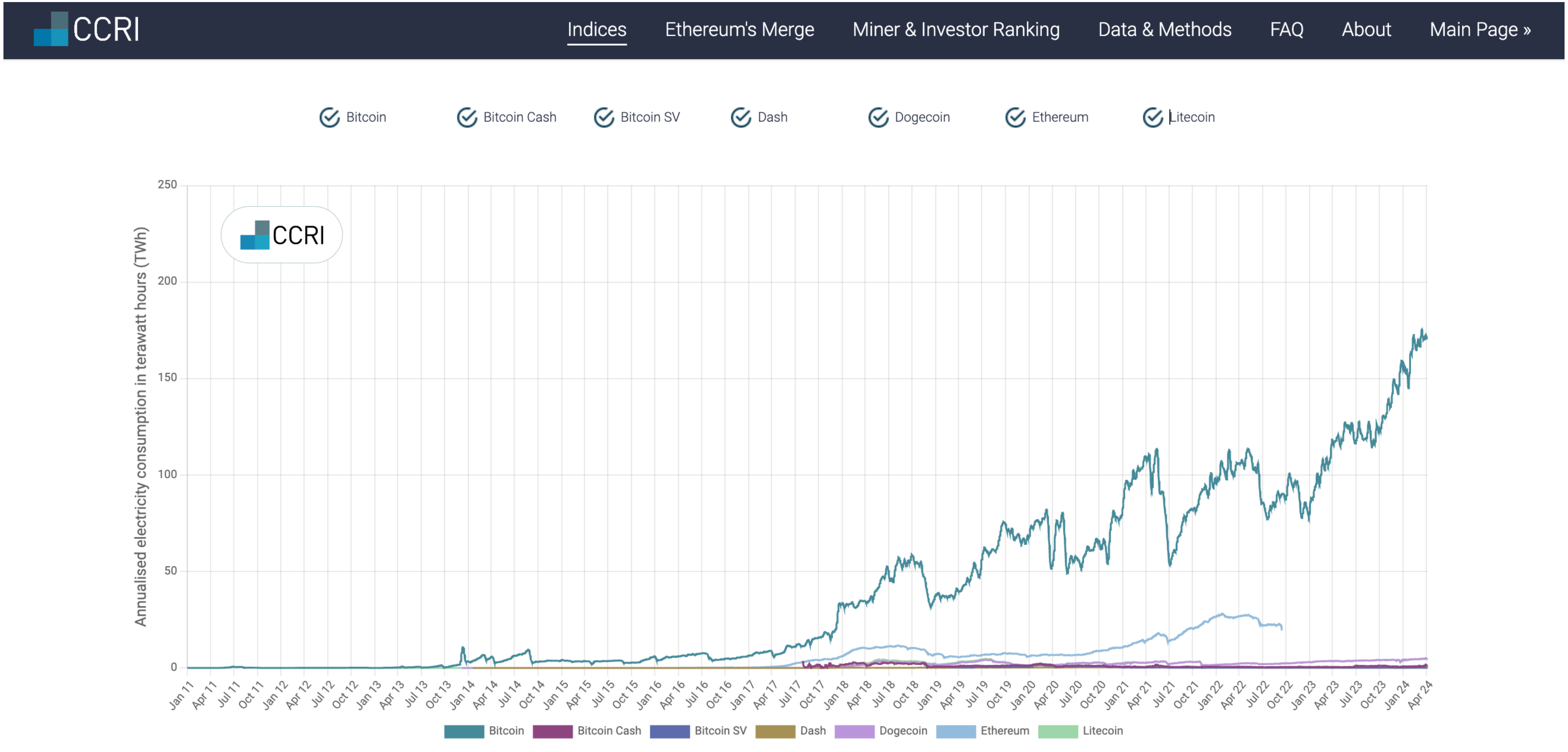
Source: https://digiconomist.net/bitcoin-energy-consumption

Cambridge Bitcoin Electricity Consumption Index



See here: https://ccaf.io/cbnsi/cbeci

Crypto Carbon Ratings Institute



See here: https://indices.carbon-ratings.com/

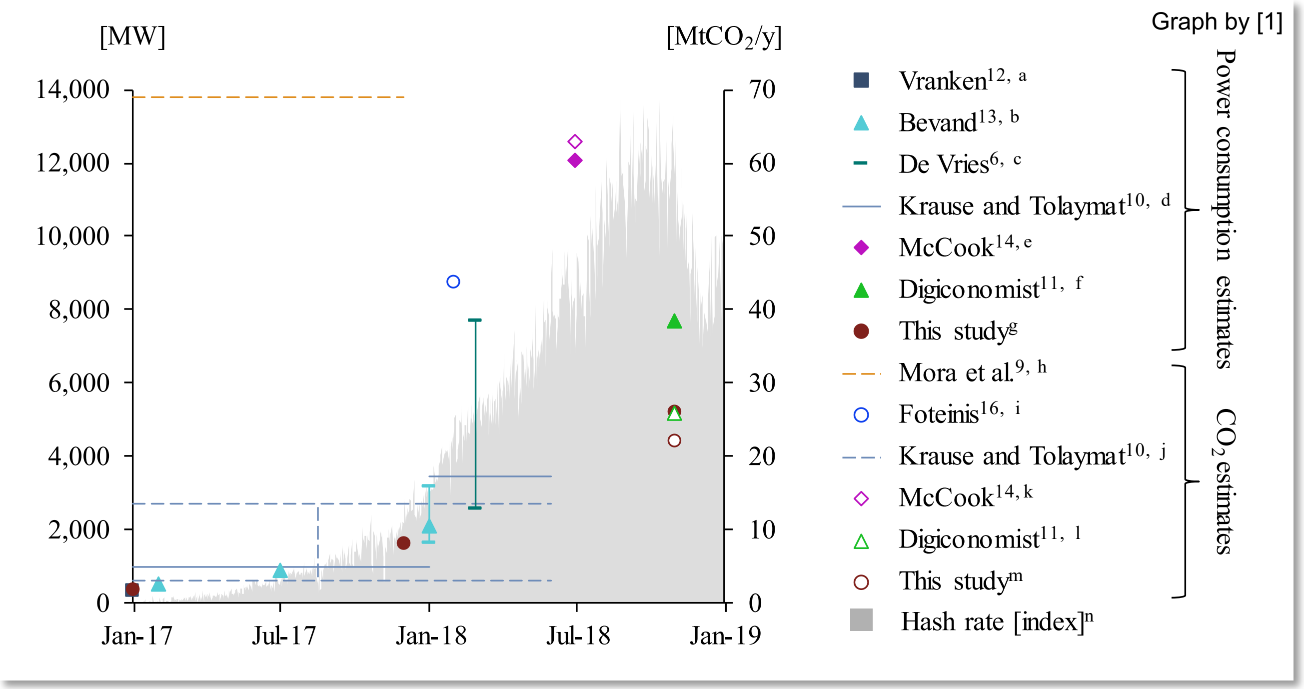
Estimated Bitcoin Energy Consumption

Description	Value 04/2020	Value 04/2019	2018
Bitcoin's current estimated annual electricity consumption* (TWh)	75.83	54.27	66.81
Bitcoin's current minimum annual electricity consumption** (TWh)	46.63	41.98	--
Annualized global mining revenues	$4,946,988,794	$3,855,324,810	$7,013,774,044
Annualized estimated global mining costs	$3,791,577,894	$2,713,725,581	$3,340,362,318
Current cost percentage	76.64%	70.39%	47.63%
Country closest to Bitcoin in terms of electricity consumption	Chile	Bangladesh	Czech Republic
Estimated electricity used over the previous day (KWh)	207,757,693	148,697,292	183,033,552
Implied Watts per GH/s	0.077	0.117	0.235
Total Network Hashrate in PH/s (1,000,000 GH/s)	112,236	52,757	32,502
Energy footprint per transaction (KWh)	719	413	895
Number of U.S. households that could be powered by Bitcoin	7,021,441	5,025,418	6,185,856
Number of U.S. households powered for 1 day by the electricity consumed for a single transaction	24.28	13.95	30.24
Bitcoin's electricity consumption as a percentage of the world's electricity consumption	0.34%	0.24%	0.30%
Annual carbon footprint (kt of CO2)	36,020	25,780	32,736
Carbon footprint per transaction (kg of CO2)	341.31	196.04	438,48

As of April 2020

Source: https://digiconomist.net/bitcoin-energy-consumption

Comparison of Different Approaches for November 2018



Approach	Estimate (11/2018)
Stoll et al.[1]	45,8 TWh
BBSE[2]	42,7 TWh
CBECI[3]	43,3 TWh
Digiconomist[4]	73,2 TWh

Major takeaways:

- Discussion is still ongoing
- Many different approaches: Similar results ➔ Bitcoin energy consumption is a problem
- Energy consumption alone not conclusive: CO_2-emissions are relevant for climate change
- Other cryptocurrencies (besides Bitcoin) with Proof-of-Work contribute further to the problem

[1] Stoll, Christian, Lena Klaaßen, and Ulrich Gallersdörfer. "The carbon footprint of bitcoin." *Joule* 3.7 (2019): 1647-1661
[2] "Back of the envelope-calculation", assuming 51.8 Mio. TH/s as of November 2018 and Antminer S9
[3] Cambridge Bitcoin Electricity Consumption Index, available at https://www.cbeci.org/
[4] Digiconomist: Bitcoin Energy Consumption Index. Available at https://digiconomist.net/bitcoin-energy-consumption

Bitcoin's Challenges

- Bitcoin Script is **limited in its expressive power**!
  - **Ethereum** and other solutions provide a Turing complete[1] Smart Contract language!
- Bitcoin does **not scale** / is too slow!
  - Layer-2 solutions like **Lightning Network**[2] enable higher transaction throughput with lower fees.
- Bitcoin is **too volatile!**
  - Stable coins either pegged by fiat currencies (**Tether**) or by collateral[3] (**Dai**) provide more stable prices.
- **Regulations!**
  - On one hand: over-regulations of cryptocurrencies in United States.
  - On the other hand: little or no regulation in many other countries.

[1]Turing complete means that any system can be replicated in the respective programming language.
[2]Lightning Network Technical paper: https://lightning.network/lightning-network-paper.pdf
[3]Collateral are other assets in blockchains, e.g., tokens or cryptocurrencies. Dai is backed by Ether, the cryptocurrency in Ethereum.