PV204 Security technologies

Bitcoin basics I.

Petr Švenda ✉ ***svenda@fi.muni.cz*** 🐦 ***@rngsec***
Centre for Research on Cryptography and Security, Masaryk University

CR⊙CS

Centre for Research on
Cryptography and Security

WHY BITCOIN?

Especially if you are not interested in Bitcoin.

"Bitcoin fixes everything!"

fixes this

Bitcoin basics I.

https://twitter.com/DominicFrisby/status/1388448025970884609

Goals for the lecture

- Bitcoin does not fix everything, but is on a frontline
  - No safety net, no chargeback, attacker anonymous => security technique must really work, great for battle-testing security ideas, natural "bug bounty program"
- 6 main tech pieces we will cover (also usable outside Bitcoin world)
  1. How to backup key(s) (single seed, BIP39, Shamir)
  2. How to make always fresh keys (derivation via BIP32, also address privacy)
  3. How to protect signing key against malware
     - (multisig, hardware wallet, airgap pc + tx broadcast, mpc sig)
  4. How to introduce restricted signing policy (time, limit… lockscript/multisig)
  5. How to protect your financial privacy (CoinJoin, Tor)
  6. How to use hardware wallet with secure element

Motivation example – what are limitations of this example?

- I have PayPal/Visa/Venmo… why something else, slower, energy intensive…
  - Scenario: Bank ledger, user accounts after KYC (ID), transfers between users
  - Complicated
- Problems:
  - Central point of failure (maintenance, server compromise, transaction censorship, no ID, money freeze…)
  - Increasing number of "coins" in system (inflation)
  - Difficult auditability of complete current supply (Eurodollar system, dollar-denominated deposits at foreign banks, limited/no reserve requirements…)
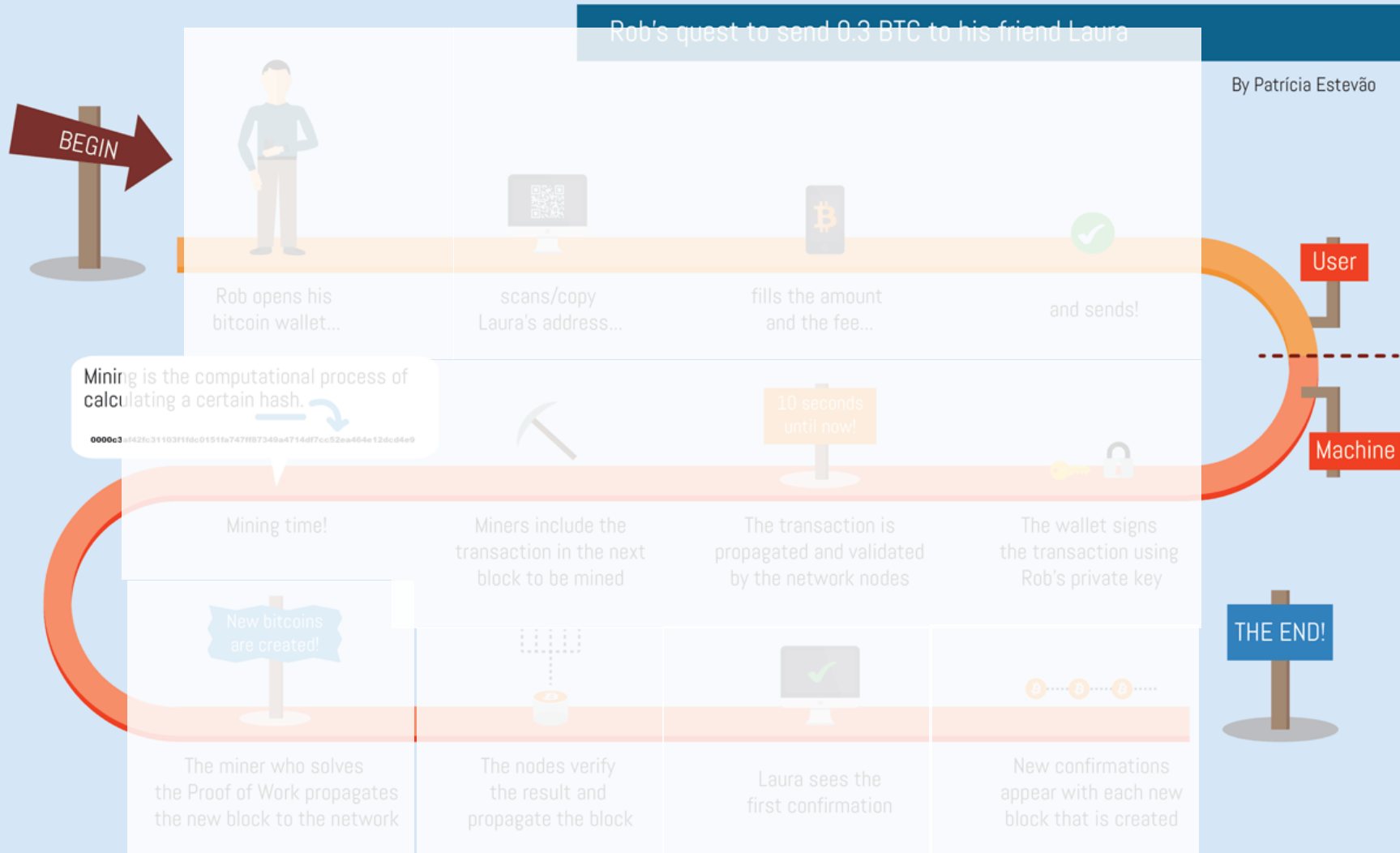
slido

What is your previous exposure to the cryptocurrencies? Please check all items which applies to you.

ⓘ Start presenting to display the poll results on this slide.

BASICS

THE BITCOIN TRANSACTION LIFE CYCLE

Rob's quest to send 0.3 BTC to his friend Laura

By Patrícia Estevão

BEGIN

User

Machine

THE END!

Rob opens his bitcoin wallet…

scans/copy Laura's address…

fills the amount and the fee…

and sends!

Mining is the computational process of calculating a certain hash.

0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9

10 seconds until now!

Mining time!

Miners include the transaction in the next block to be mined

The transaction is propagated and validated by the network nodes

The wallet signs the transaction using Rob's private key

New bitcoins are created!

The miner who solves the Proof of Work propagates the new block to the network

The nodes verify the result and propagate the block

Laura sees the first confirmation

New confirmations appear with each new block that is created

- Wallet
- Address
- Fee
- Transaction
- Signing
- Network nodes
- Block
- Mining
- Proof of Work
- Verification
- Block reward
- Tx confirmation
- And many more…

Main design goals of the Bitcoin

1. Decentralization
   - No central authority or intermediary (=> no single point of failure), possibility of self-custody
   - No limitation on network participants (no permission to join is required)
   - Applies to executing a transaction, but also development, infrastructure, mining…
2. Transparency
   - All transactions recorded on public ledger; validity of every "bitcoin" easy to verify
   - Total number of bitcoins in circulation easy to assess (monetary policy, fixed supply)
3. Security based on cryptography (mainly signature, hash functions)
   - Ownership of bitcoins proved only cryptographically (no "chargeback" based on human decision)
   - Protection of bitcoins reduced to protection of private key(s)
4. Pseudonymity of participants
   - bitcoins connected to public keys, not usernames (does not automatically mean anonymity!)

Problems to tackle

- How to prevent double spending?
- How to allow for permissionless participation?
- Who will store authoritative copy of public ledger?
- How to prevent modification of ledger history?
- Who will include next block in blockchain?
- How to maintain decentralization in distant future?
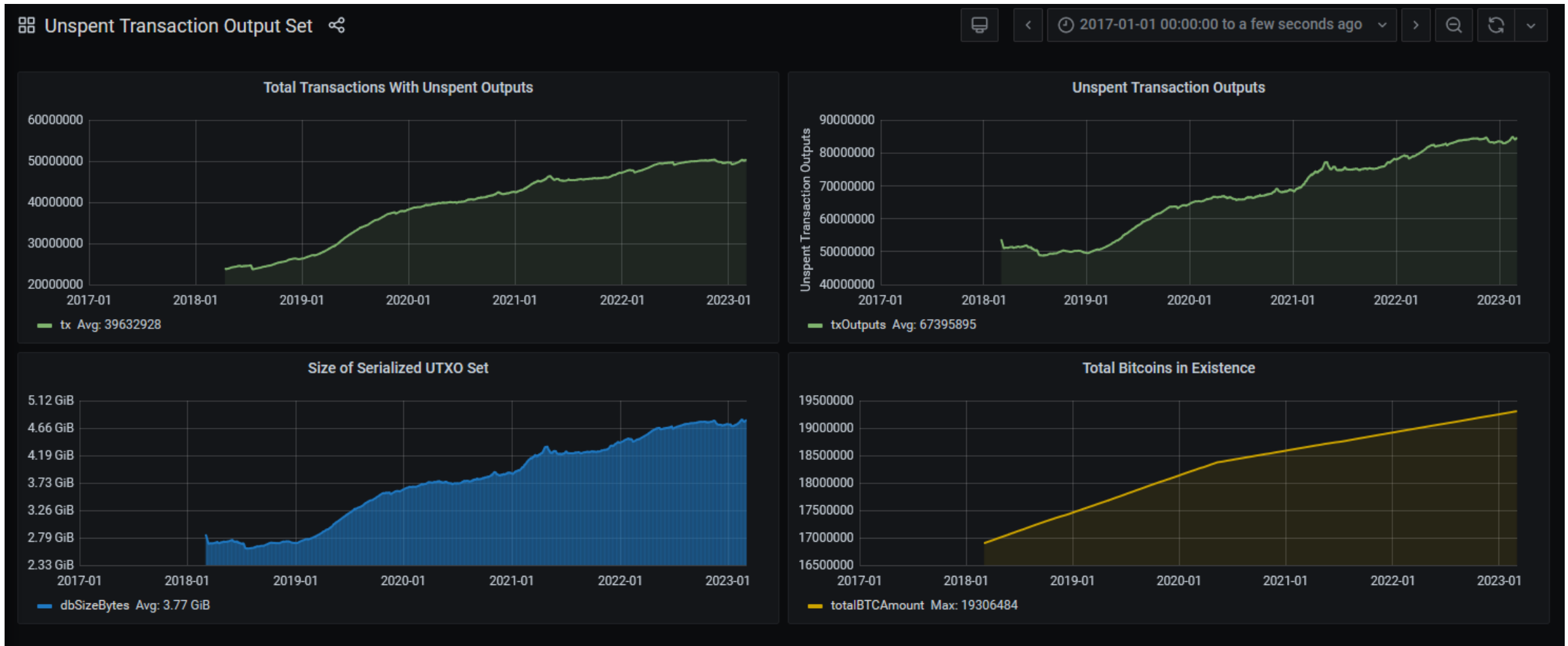- …

Double-spending problem and Bitcoin's solution

- Digital data are inherently easy to copy perfectly
  - If used as monetary coins, how to prevent double/triple… spending the same coin?
  - Previous proposals (eCash, B-money, Bit Gold..) required central party for prevention
- Digital coin X is "spent" by a transaction between users A and B
  - Double spend is another transaction from A to C using same coin X
- If all transactions are ordered strictly in time, double spend is not possible
  - Later transaction with same coin X is invalid
  - Decentralized ordering is costly as all participants need to agree on global state
- If ordered after every transaction => costly and slow
- Bitcoin orders in batches of transaction every 10 minutes on average
  - User needs to wait one (or more blocks) for ordering (longer => higher certainty)
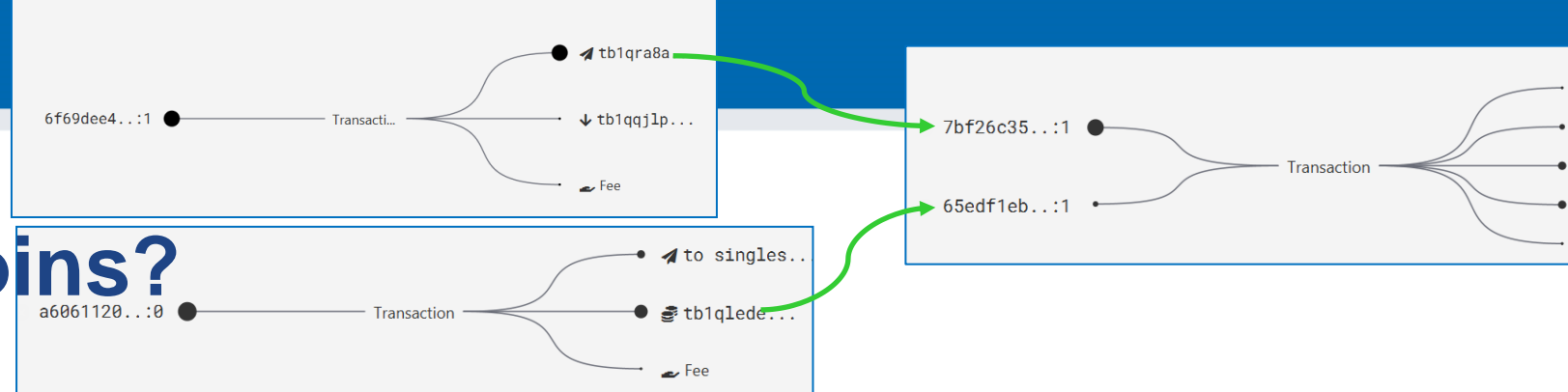
Where are my bitcoins?

- Public ledger of all transactions (blockchain)
  - Replicated between all Bitcoin fullnodes (P2P network)
- "Bitcoin holdings" - sum of values of not-yet-spent transactions control
  - Unspent Transaction Output (UTXO)
- "Bitcoin receive" operation – generate variable part of lockscript (public) and share with sender + monitor blockchain for my transaction
- "Bitcoin send" operation – take "your" UTXO and use it as input to new one
  - Specify recipient by script specifying what must be done int future send (lockscript)
  - Typical lockscript is "prove that you can sign with private key corresponding to THIS public key"
- Protection and handling of private keys is paramount
  - "Not your keys, not your bitcoin!"

UTXO set = all currently valid "bitcoins"

Heatmap distribution of UTXOs in time and value

Hash	000000000000000000000d4d0649c6fcdfafddccbb729a4ace783289e6ba065a52
Height	661045
Timestamp	2020-12-12 15:05:30 UTC
Size	1326415 B
Weight Units	
Number of Transactions	
Number of UTXO created	
Number of UTXO destroyed	5838
Difficulty	19157154724710.14
Merkle Root	9faae5978325ea3df8b033f3fd7cfb7fb023baa5a0dfc34838ca73a4d7738
Chainwork	0000000000000000000000000000000000000000016ca55a4151edf173d4ea738
Version	0x20000000
Bits	0x170eb156
Nonce	0x28dd04f9

https://utxo.live/
https://github.com/martinus/BitcoinUtxoVisualizer

Heatmap distribution of UTXOs in time and value

The Current UTXO Set

19 Mar 2023 block 781431

Number of unspent outputs

https://utxo.live/
https://github.com/martinus/BitcoinUtxoVisualizer

Problem: How to allow for permissionless participation?

BITCOIN NETWORK

Bitcoin P2P network

Blockchain

fullnode

fullnode

SW-only wallet

With hardware wallet
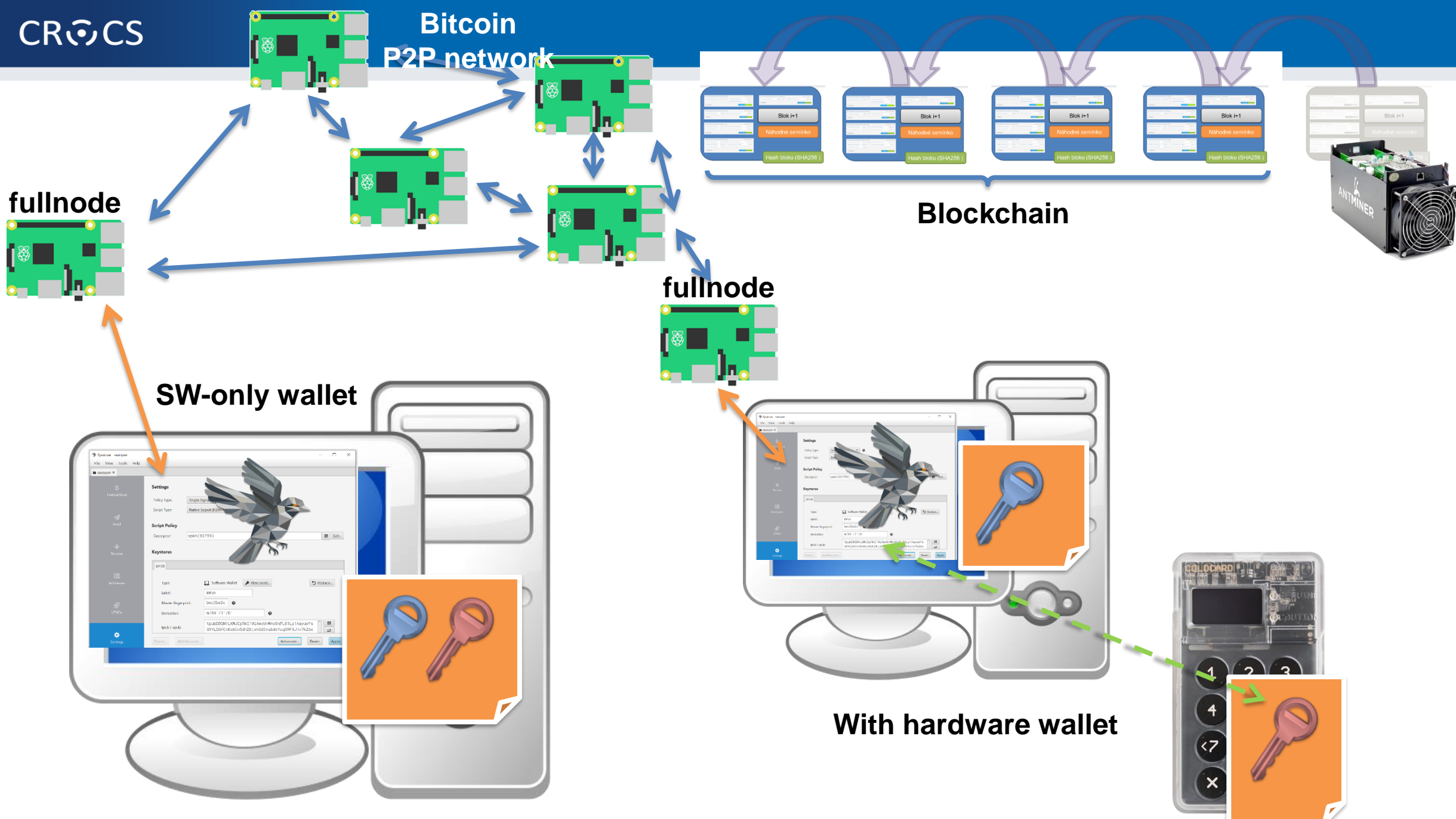
P2P Bitcoin network map https://bitnodes.io/

REACHABLE BITCOIN NODES

Updated: Sat Mar 18 10:21:17 2023 CET

16537 NODES   [CHARTS]

IPv4: +0.1% / IPv6: +0.6% / .onion: +21.8%

Top 10 countries with their respective number of reachable nodes are as follows.

RANK	COUNTRY	NODES
1	n/a	9992 (60.42%)
2	United States	1752 (10.59%)
3	Germany	1403 (8.48%)
4	France	448 (2.71%)
5	Netherlands	398 (2.41%)
6	Canada	273 (1.65%)
7	Finland	240 (1.45%)
8	United Kingdom	211 (1.28%)
9	Russian Federation	169 (1.02%)
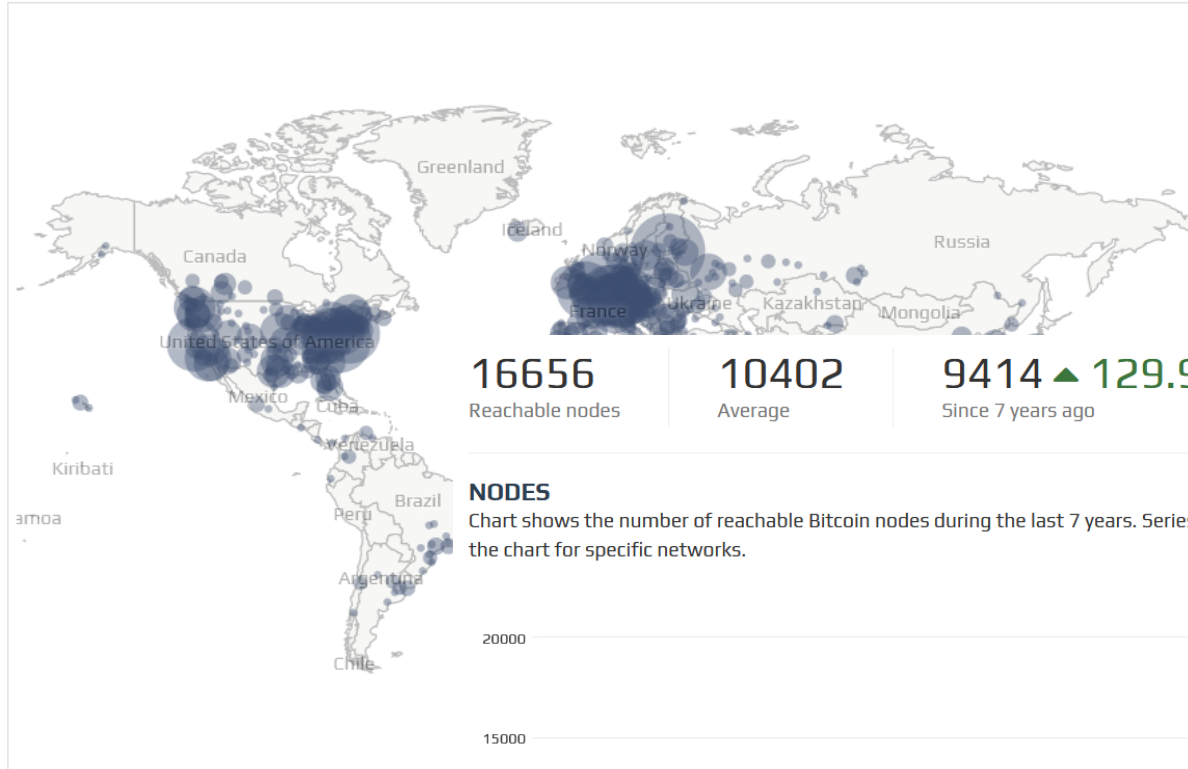
Map shows concentration of reachable

16656	10402	9414 ▲ 129.99%
Reachable nodes	Average	Since 7 years ago

NODES

Chart shows the number of reachable Bitcoin nodes during the last 7 years. Series can be enabled or disabled from the legend to view the chart for specific networks.

[24h 90d 1y 7y]

Lo 5135   Hi 16656   Avg 10402   Last 16656 nodes

IPv4   IPv6   .onion
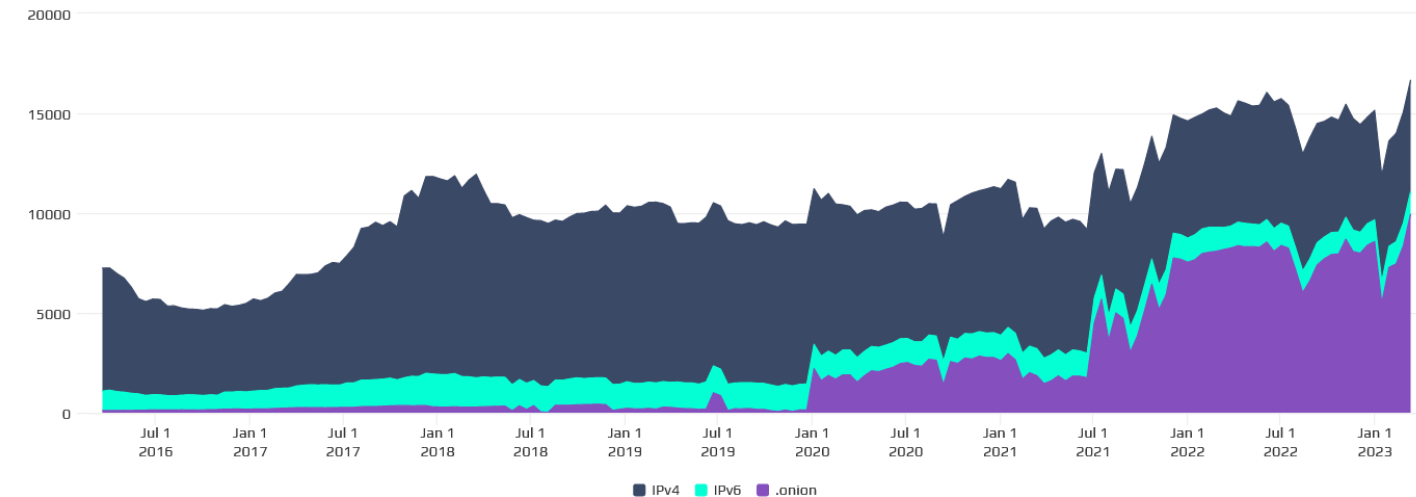
What is Bitcoin "fullnode"

- Software capable to connect and interact with P2P network
- Downloads whole blockchain, fully verifies all blocks (PoW) and transactions starting from the Genesis block (or trusted checkpoint)
  - Dynamically builds own UTXO set (unspent txs) and Mempool (unconfirmed txs)
- Propagates new incoming blocks and transactions
- No formal specification of Bitcoin consensus exists
  - Bitcoin Core software is defacto specification (https://github.com/bitcoin/bitcoin)
  - Other implementations also exists (but large majority of nodes are Bitcoin Core)
- Currently several days to fully synchronize (CPU/bandwith), ~465GB
- Can be run over Tor to protect user privacy
- Bitcoin wallet needs to connect to some fullnode (your = better privacy)

Networks in Bitcoin (Mainnet, Testnet, Regtest, Signet)

- Mainnet – main, global production network ("real" bitcoins)
- Testnet – testing network (global, some mining happens…)
  - Restarted from time to time, contains many different types and versions of TXs
- Regtest – local instance of Bitcoin network
  - Used for local testing (integration, regression,debugging)
  - Blockchain started from block 0, you are the only miner
  - (mined bitcoins unusable on Mainnet)
  - You can insert own transactions, decide on mining new blocks, debug…
- Signet – testing network like Testnet, but with features not yet active on Mainnet
  - Initially for testing Taproot, now for future possible softforks
- (Lighting – second layer network of payment channels atop of mainnet)
  - Practically instant and very low fees independently from mainnet

Mempool – unconfirmed transactions

- Every fullnode maintains own list of unconfirmed transactions (mempool)
  - No single global mempool! But local mempools tend to synchronize quickly
- Miners construct next block from transactions maximizing profit (mostly)
- What if tx is in a mempool, but with too low fee (not getting confirmed)?
  1. Child pays for parent –additional transaction spending output of previous (high fee)
  2. Replace By Fee (RBF) flag – new tx, but with higher fee, replaced by nodes
  3. (Wait for purge, pay miner out of band…)
- If too many unconfirmed txs present, some existing are purged (removed)
  - Default size of mempool (for Bitcoin Core) is ~300MB
  - Selection depends on configuration (low-fee tx, large tx, old tx)
  - If discarded, it can be re-inserted later from other nodes or resubmitted (by owner)

Popular mempool explorer – https://mempool.space
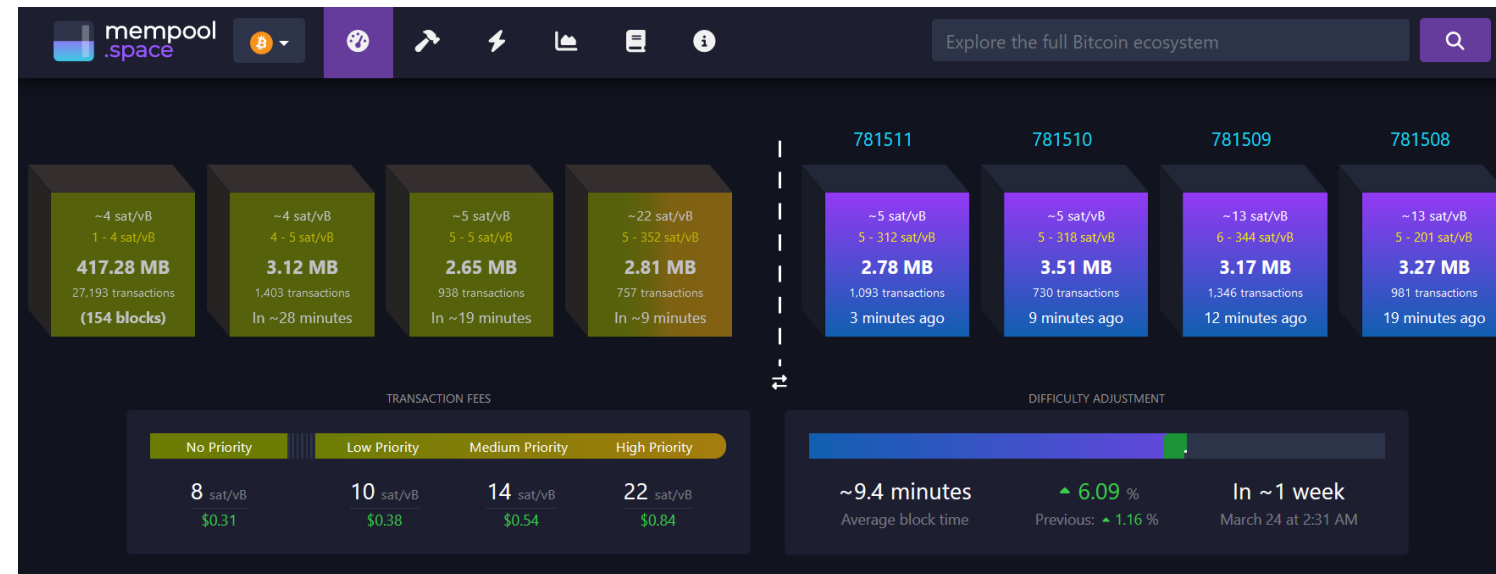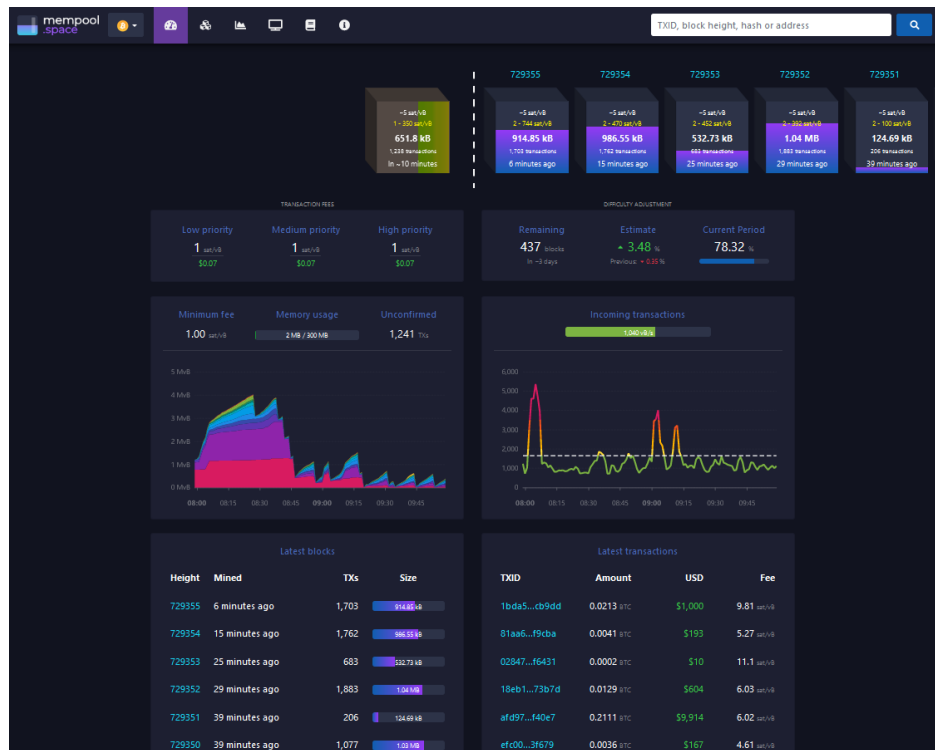
- Can be run on your own fullnode (privacy improvement)
- Testnet version https://mempool.space/testnet

Mempool size in time

Blocksize wars in 2017, blockchain "spamming", bullrun Dec 2017 (~$20k)

Bullrun towards May 2021 (~$69k)

Ordinals/Inscription Feb 2023
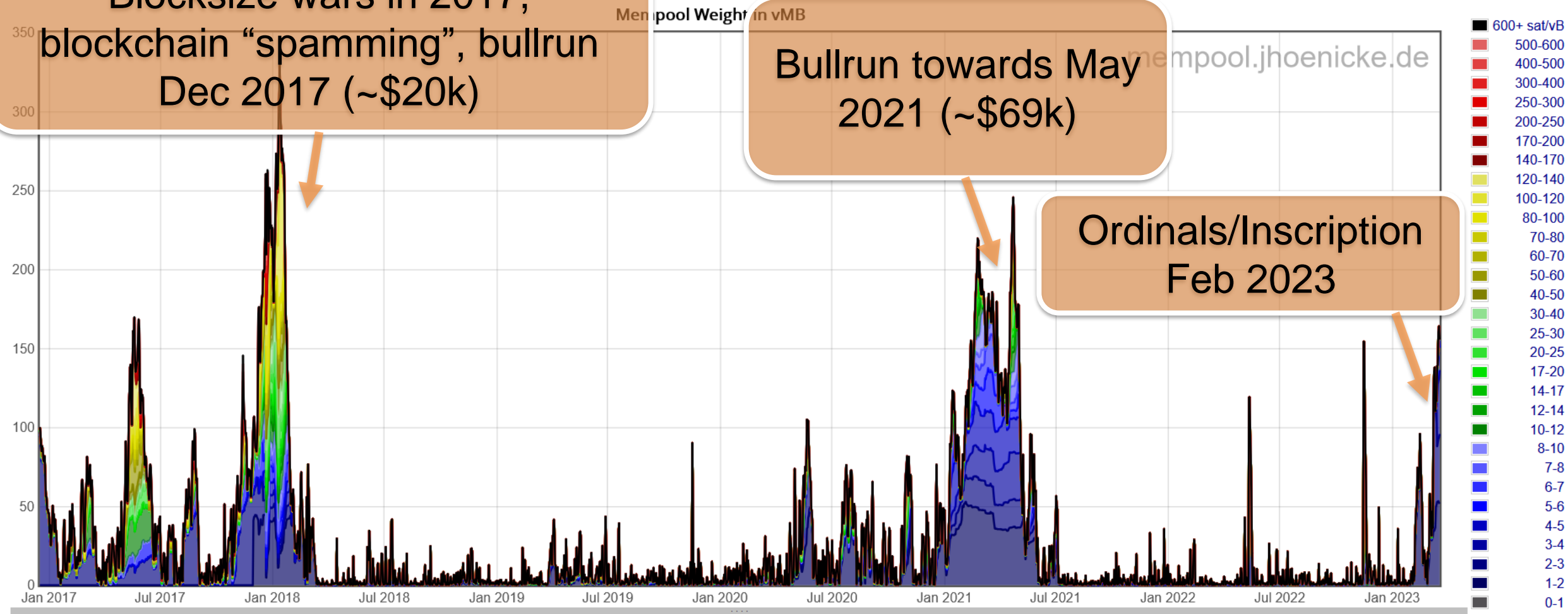
DEMO: LOOK AND COMMENT THE CURRENT MEMPOOL STATE

TRANSACTION

Transaction components

- Binary structure optimized for small size (further decreased over time)
  - Version
  - Inputs (bitcoins spent, points to some previous tx outpt + unlock script)
  - Outputs (bitcoins received, description of lock script)
  - Locktime (when starts to be valid, absolute or relative, time or block height)
- Can be created offline, broadcasted immediately or later (Lighting)
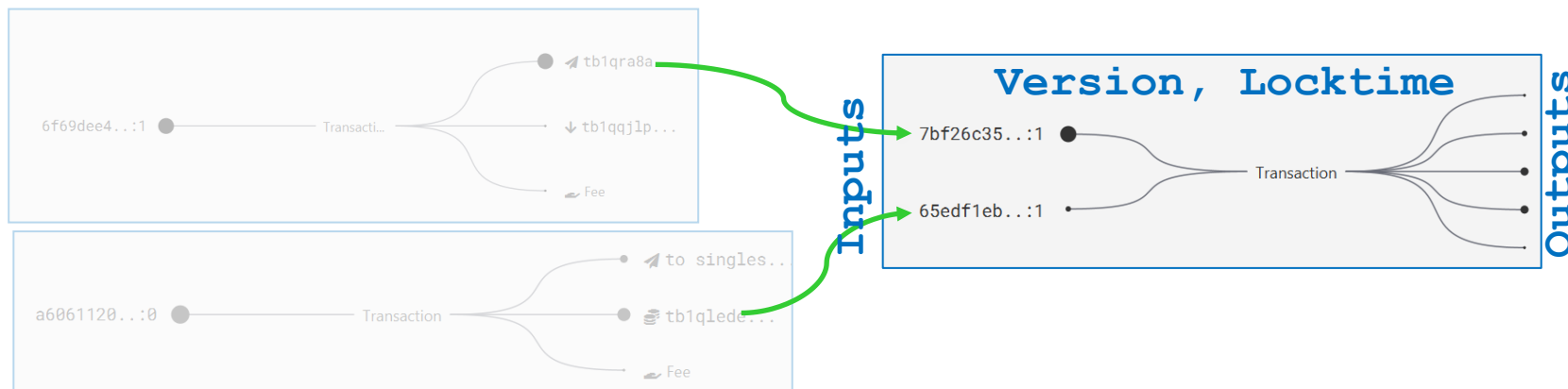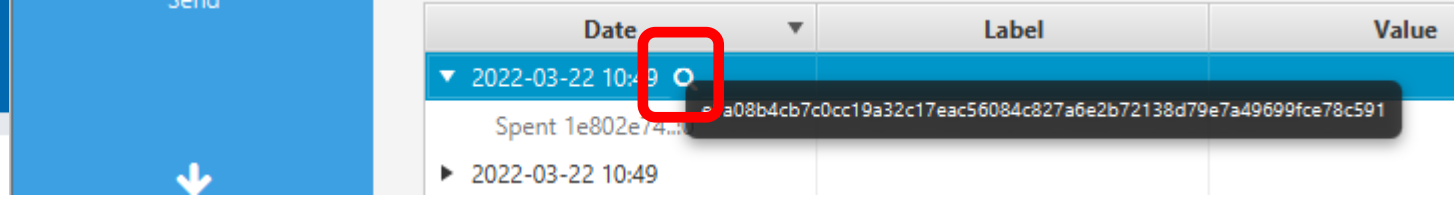
Various transactions can be created

One input, two outputs + fee (possibly classic pay)

Two inputs, one output + fee (possibly consolidation)
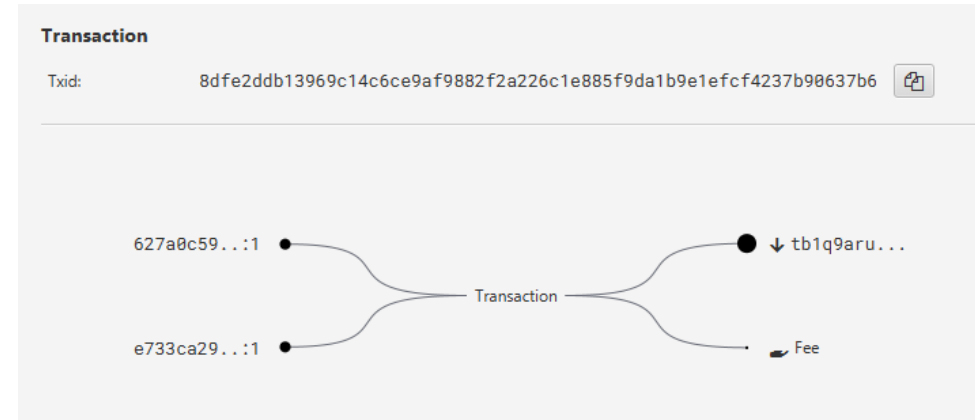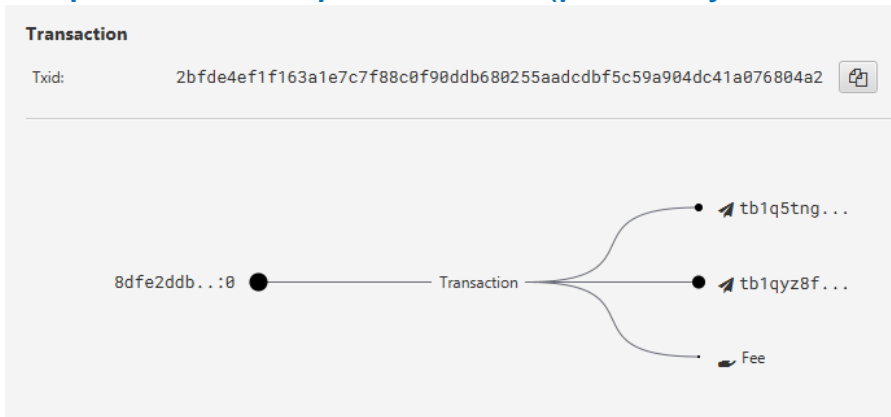
One input, many outputs, fee (CoinJoin premixing)

Many inputs, many outputs, fee…
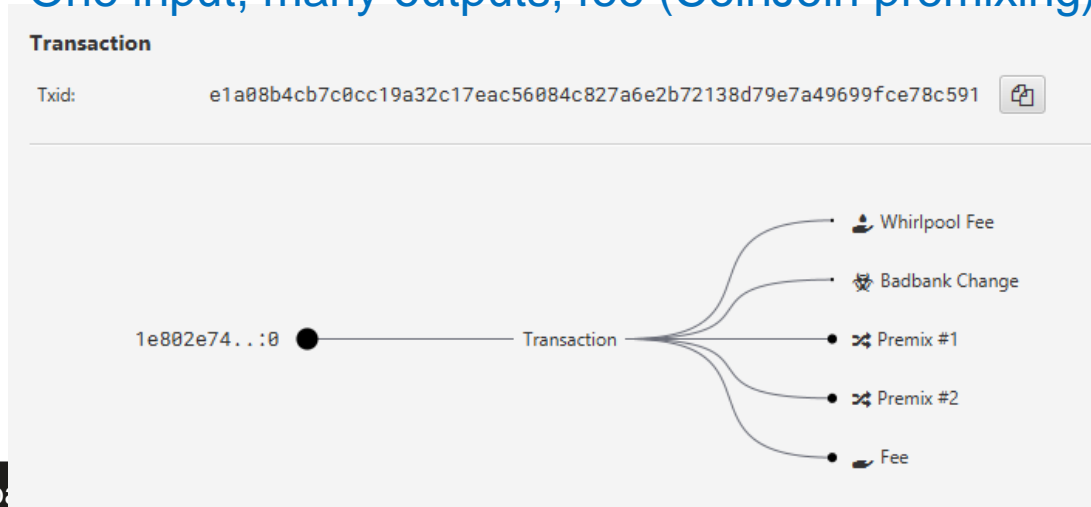
Standard pay (change likely bc1)
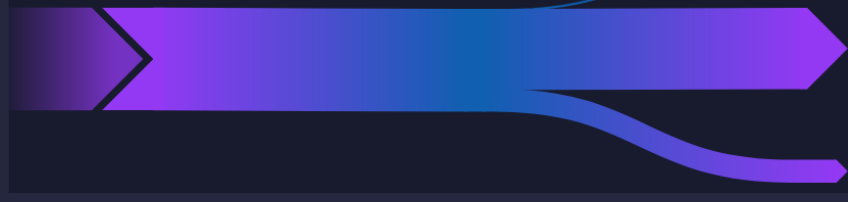
Inputs & Outputs — Details

bc1qcfrqk98y3f2yxfpwgm4…cd3l7jrh	0.01824993 BTC	16g2ZjLHG86GMtxJJtVXQcq…qBfWYAfK	0.01430567 BTC →
		bc1qcfrqk98y3f2yxfpwgm4…cd3l7jrh	0.00391114 BTC →
		Details	0.01821681 BTC

Multisig pay (3.87) from multiple inputs
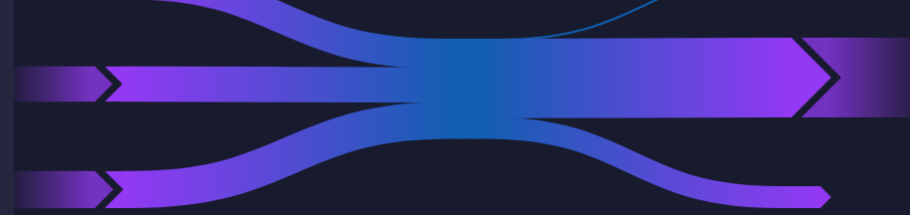
Inputs & Outputs — Details

bc1qc8dyets2cdckkuf29kd…aswdf19e Multisig 2 of 3	1.43051325 BTC	36Sn1rywJ6V6xGU3BMZMsfg…jy4AXp3J	3.87400001 BTC →
bc1qguwfqf0np430ygmmvrw…wsxghtjq Multisig 2 of 3	1.69673127 BTC	bc1qn8x5uzaucq7cu0hn4mk…nq9sc45p	1.03549133 BTC →
bc1qn8x5uzaucq7cu0hn4mk…nq9sc45p Multisig 2 of 3	1.78239593 BTC		
			4.90949134 BTC

Standard pay (0.04) with consolidation

Inputs & Outputs — Details

bc1q7e7j5u8rk6we50j2y4v…xsssws7q	0.00433811 BTC	bc1q9clhqd4mp36h4v4cfmd…773lru3d	0.04000000 BTC →
bc1q7e7j5u8rk6we50j2y4v…xsssws7q	72.63982966 BTC	bc1q7e7j5u8rk6we50j2y4v…xsssws7q	72.60409777 BTC →
			72.64409777 BTC

Consolidation (even annotated OP_RETURN)

Inputs & Outputs — Details

|			
bc1qpraml47gzdrznusqt6x…549jwzja	117.11561344 BTC	bc1qpraml47gzdrznusqt6x…549jwzja	117.14108466 BTC →
bc1qpraml47gzdrznusqt6x…549jwzja	0.00241705 BTC	OP_RETURN consolidate	0.00000000 BTC →
bc1qpraml47gzdrznusqt6x…549jwzja	0.00619855 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00010001 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00158000 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00591711 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00355191 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00158659 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00225320 BTC		
bc1qpraml47gzdrznusqt6x…549jwzja	0.00200000 BTC		
			117.14108466 BTC

DEMO: LOOK AT CURRENT MEMPOOL TRANSACTIONS (CONFIRMED, UNCORFIRMED)
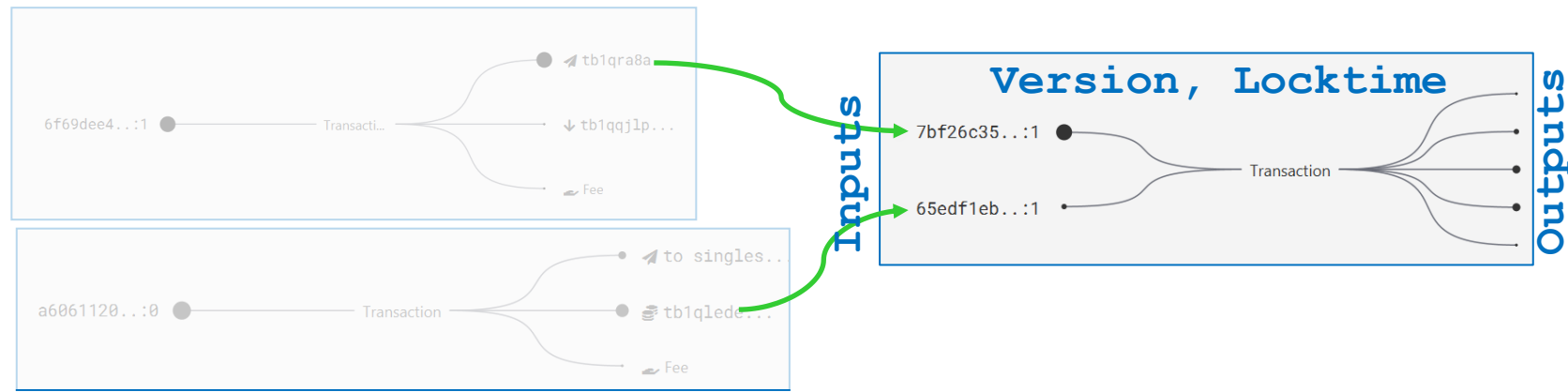
LOCK AND UNLOCK SCRIPTS

Types of receiving "addresses"

- There is no "address" defined in Bitcoin network
- Standard patterns how to construct lock script emerged over the time
  - e.g., unlock if signature is verifiable with the public key stored in lock script (P2PK)
  - "Address" is the variable part of the lock script differing between (different receivers / transactions)
- *Notation warning: scriptSig (script + signature), scriptPubKey (initial meaning script + public key == P2PK)*
- Well-known standard types of lock scripts
  - Pay-to-public-key (P2PK)
  - Pay-to-public-key-hash (P2PKH, starts with 1)
  - Pay-to-script-hash (P2SH, BIP16)
  - OP_RETURN (any data 40B)
  - Native Pay-to-witness-script-hash (P2WSH, starts with 3)
  - P2WSH-nested-in-P2SH
  - P2SH-P2WPKH, P2SH-P2WSH
  - Native P2WPK, P2WSH (Bech32, starts with bc1)
  - Pay-to-Taproot (P2TR, Schnorr signature, starts bc1p)
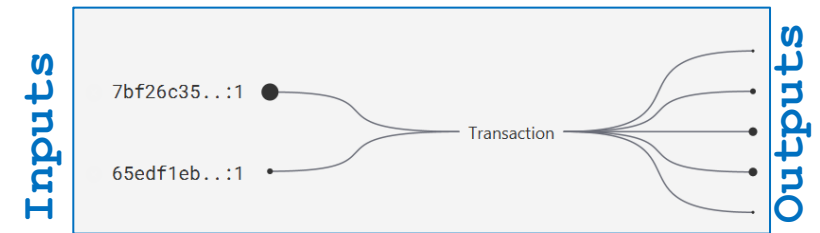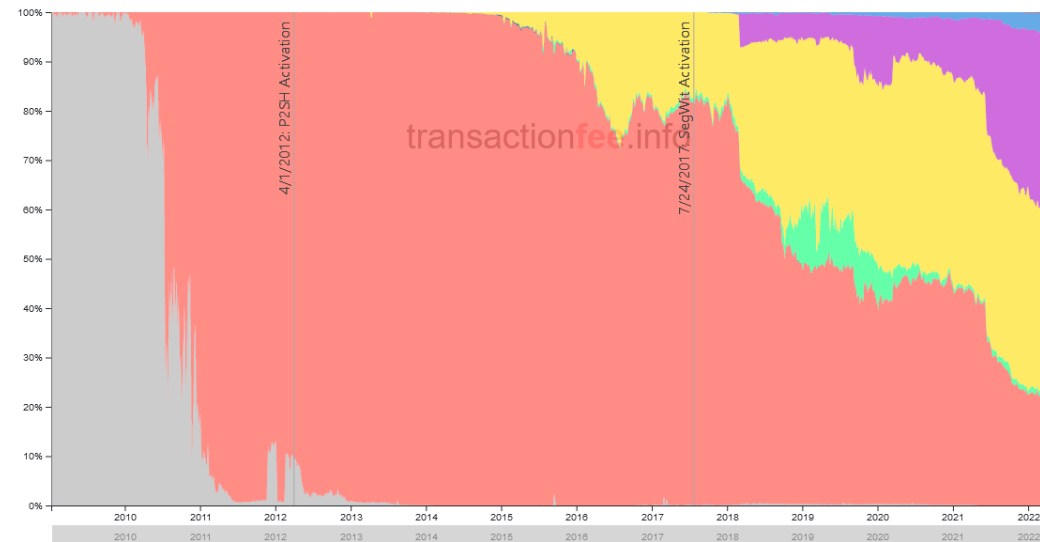
Pay-to-public-key (P2PK), Pay-to-public-key-hash (P2PKH)

- Pay-to-public-key (P2PK)
  - Lock script contains direct value of public key and instructions to push signature and verify with the public key
  - Used initially by Satoshi and others, now infrequent
  - Disadvantage: if practical dlog attack against secp256k1 is found, private key can be computed
- Pay-to-public-key-hash (P2PKH), starts with '1'
  - Lock script contains hash of public key later used for signature verification
  - Advantage: smaller lockscript, attacker does not know public key until spent

P2PKH - script execution (https://nioctib.tech/)

Executing **Script Sig** [1]

Stack

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d
OP_DUP OP_HASH160
0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d OP_EQUALVERIFY
OP_CHECKSIG

Executing **Script Sig** [2]

Stack

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

OP_DUP OP_HASH160
0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d OP_EQUALVERIFY
OP_CHECKSIG

Stack

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901
0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d
OP_DUP OP_HASH160
0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d OP_EQUALVERIFY
OP_CHECKSIG

Executing **Script PubKey** [3]

Stack

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

OP_DUP OP_HASH160
0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d OP_EQUALVERIFY
OP_CHECKSIG

Executing **Script PubKey** [4]

Stack

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

OP_HASH160 0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d
OP_EQUALVERIFY OP_CHECKSIG

Executing **Script PubKey** [5]

Stack

0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d OP_EQUALVERIFY
OP_CHECKSIG

Executing **Script PubKey** [6]

Stack

0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d

0x6f3f0b93b060ea9c0d76989c9747c9b6cfad617d

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe10
29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f3399de0c29b593d022048267261596dcdb8a49659f0a9c702058b56a8b90fb39e8ff901

Script

OP_EQUALVERIFY OP_CHECKSIG

Executing **Script PubKey** [7]

Stack

1

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

OP_VERIFY OP_CHECKSIG

Executing **Script PubKey** [8]

Stack

0x02b621afa86afdb74d874e876413cf199833f4a5f68e10335134876eebe29bbe6d

0x304402205c5876144bf491eb6aece2625cbc3049819f35094e8feaf808399de0c29b593d022048267261596dcdb8a49659f0a9c74f2a423d6c7bef02058b56a8b90fb39e8ff901

Script

OP_CHECKSIG

Executing **Script PubKey** [9]
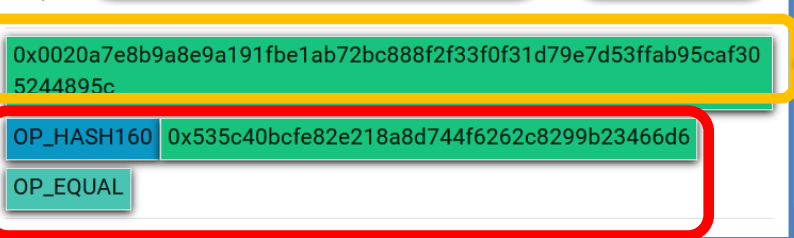
Stack
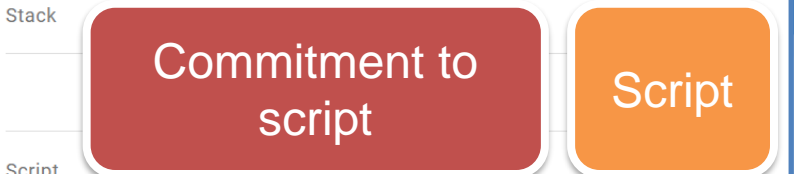
1

Script

Execution Succeeded

Stack

1

Script

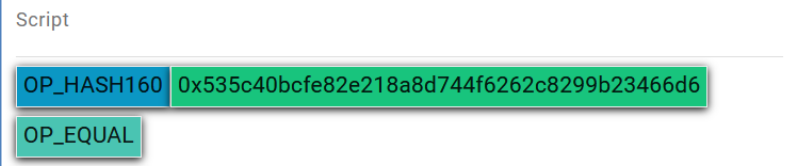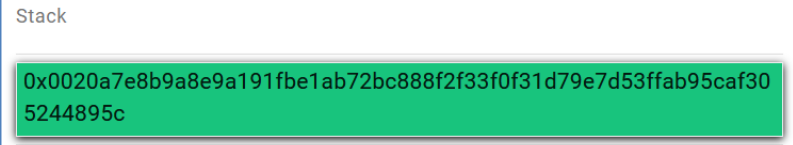# Pay to script hash (P2SH), BIP16, starts with '3'

- Lock script separated into two parts
  - 1) commitment to the script (hash value, checked later)
  - 2) actual lock script (hash value must match the commitment)
- Sending tx sets output's ScriptPub to the commitment
  - Shorter as only hash is posted, not whole lock script
  - Lock script is provided only later when spending (privacy, fee to be paid)
  - Lock script can have multiple spending paths (Merkle tree) and only the one used is posted (better for privacy)
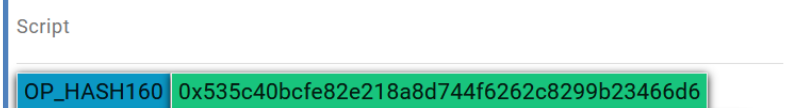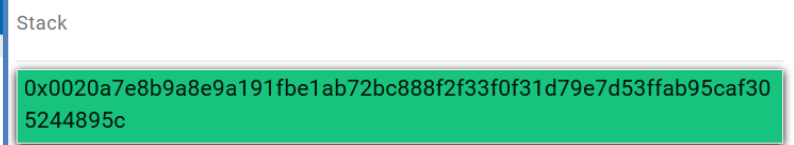- Redeeming tx provides actual lock script + unlock script

Executing Script Sig [0]

Stack

> Commitment to script
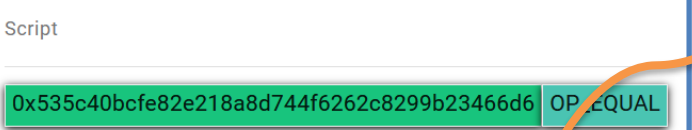
> Script

Script

`0x0020a7e8b9a8e9a191fbe1ab72bc888f2f33f0f31d79e7d53ffab95caf305244895c`

`OP_HASH160` `0x535c40bcfe82e218a8d744f6262c8299b23466d6`

`OP_EQUAL`

Executing Script Sig [1]

Stack

`0x0020a7e8b9a8e9a191fbe1ab72bc888f2f33f0f31d79e7d53ffab95caf305244895c`

Script

`OP_HASH160` `0x535c40bcfe82e218a8d744f6262c8299b23466d6`

`OP_EQUAL`

Executing Script PubKey [2]

Stack

`0x0020a7e8b9a8e9a191fbe1ab72bc888f2f33f0f31d79e7d53ffab95caf305244895c`

Script

`OP_HASH160` `0x535c40bcfe82e218a8d744f6262c8299b23466d6`
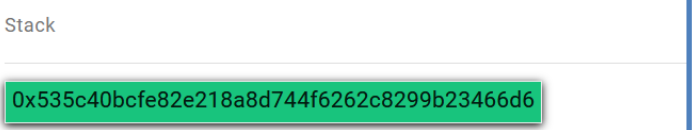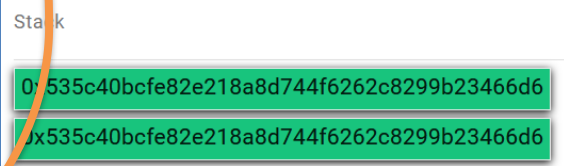
> If initial script structure was commitment and value on stack is true, special code branch of code is executed, using original witness script

Executing Script PubKey [3]
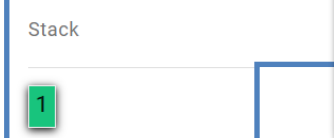
Stack

`0x535c40bcfe82e218a8d744f6262c8299b23466d6`
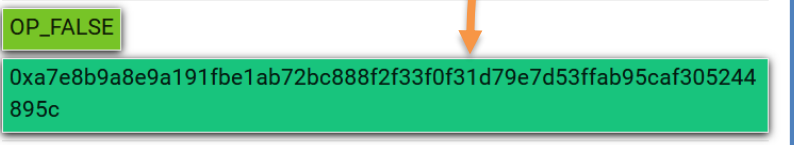
Script

`0x535c40bcfe82e218a8d744f6262c8299b23466d6` `OP_EQUAL`

Executing Script PubKey [4]

Stack

`0x535c40bcfe82e218a8d744f6262c8299b23466d6`

`0x535c40bcfe82e218a8d744f6262c8299b23466d6`

Script

`OP_EQUAL`

> Check script hash

Execu...

Stack

`1`

Stack

`0x3044022006eface088364596f612ddd158a4101c71ff770bc5a6bbe35930d9790376b87a022047e165e0fce43a446a7524b5530023e805a7460c1b564f96182f2c1efa5003c901`

`0x3044022053d0678e8994a4b10fa883047f584d372026cf6576a84bebdae0c131c04bb622022066a6b7f64dee7606a6db9d30b0b7dfd42e2c0250a98adb61d13`

`0`

Script

`OP_2`

`0x02263d...851fba58`

> Witness script is executed (here 2-of-3 multisig) OP_FALSE is used to push 0 on stack (multisig bug)

`0x02b700d35d7d372af7de1a4108032883dd1f38a6a4b0bb43bc9ec62a6641a7f5a4`

`0x0357d8e66ae3104c01b28037355b8e66baffc37e12d6fa346f9a4025754372196c`

`OP_3` `OP_CHECKMULTISIG`

Executing Script P2SH [6]

Stack

Script

`OP_FALSE`

`0xa7e8b9a8e9a191fbe1ab72bc888f2f33f0f31d79e7d53ffab95caf305244895c`

Executing Script P2SH [7]

Stack

`0`

Script

`0xa7e8b9a8e9a191fbe1ab72bc888f2f33f0f31d79e7d53ffab95caf305244895c`

...

Executing Script P2SH [8]

Stack

`0xa7e8b9a8e9a191fbe1ab72bc888f2f33f0f31d79e7d53ffab95caf305244895c`

`0`

Script

Interesting, non-standard scripts

- SHA1 collision bounty
  - Bitcoins locked to script requiring two different inputs hashed to same SHA1 hash
  - Redeemed shortly after Google published SHA1 collision blocks
    - https://blockstream.info/tx/8d31992805518fd62daa3bdd2a5c4fd2cd3054c9b3dca1d78055e9528cff6adc
    - https://nioctib.tech/#/transaction/f2f398dace996dab12e0cfb02fb0b59de0ef0398be393d90ebc8ab397550370b
  - More details: https://bitcoinjs-guide.bitcoin-studio.com/bitcoinjs-guide/v5/part-three-pay-to-script-hash/puzzles/computational_puzzle_sha1_collision_p2sh.html
  - Similar bounties for

OP_RETURN

# of OP_RETURN outputs
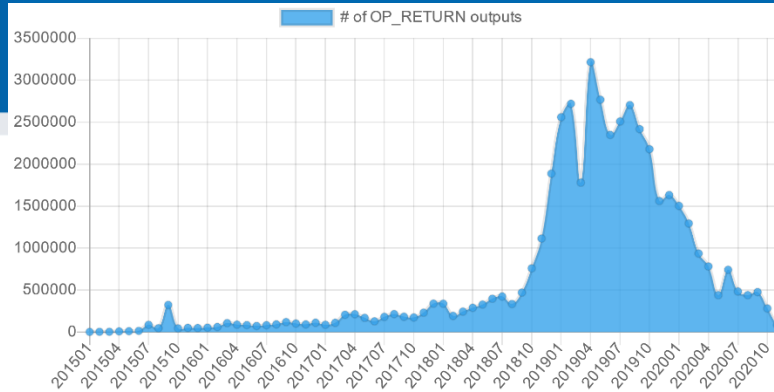
- If OP_RETURN is encountered during execution of unlock+lock script, it is FALSE
  - Such output is provably unspendable
- Somewhat controversial instruction
  - Some feels, that blockchain shall not be used for non-financial data (USDT was initially on Bitcoin via OP_RETURN)
  - But there were already ways how to store arbitrary data into blockchain anyway (e.g., bytes of value, invalid address)
- Analysis of OP_RETURN data
  - https://www.blockchainresearchlab.org/2020/03/13/how-do-op-return-transactions-impact-bitcoin/
  - https://opreturn.org/

Paying from

🏠 *1HnhWpkMHMjgt167kvgcPyurMmsCQ2WPgg*

₿ 0.0022 BTC - Transaction output 1

🔒 ScriptSig - P2PKH

0x30450220446df4e6b875af246800c8c976de7cd6d7d95016c4a8f7bcdb
ba81679cbda242022100c1ccfacfeb5e83087894aa8d9e37b11f5c054a75
d030d5bfd94d17c5bc953d4a01

0x045901f6367ea950a5665335065342b952c5d5d60607b3cdc6c69a03d
f1a6b915aa02eb5e07095a2548a98dcdd84d875c6a3e130bafadfd45e694
a3474e71405a4

🐞 Interpret or debug

To

🏠 *No address*

₿ 0 BTC - not spent yet

🔒 ScriptPubKey - NULL DATA

`charley loves heidi`

OP_RETURN 0x636861726c6579206c6f76657320686569646469

🏠 *1HnhWpkMHMjgt167kvgcPyurMmsCQ2WPgg*

₿ 0.002 BTC - Transaction

🔒 ScriptPubKey - P2PKH

OP_DUP OP_HASH160

0xb8268ce4d481413c4e848ff353cd16104291c45b OP_EQUALVERIFY

OP_CHECKSIG

https://nioctib.tech/#/transaction/f2f398dace996dab12e0cf
b02fb0b59de0ef0398be393d90ebc8ab397550370b

Miniscript (A. Poelstra, P. Wuille, S. Kanjalkar, 2019)

- Language for easier and error-prone creation of Bitcoin scripts
  - Subset of Bitcoin script language
  - Human-readable, easy to express complex locking conditions
  - https://bitcoin.sipa.be/miniscript/
- Simple building blocks (policies)
  - Single-key, Multi-key,
  - Time-locks, Check-sequence,
  - Hash-lock…
- Compiler creates optimal script
  - And cost analysis

Supported policies:
- pk(*NAME*): Require public key named *NAME* to sign. *NAME* can be any string up to 16 characters.
- after(*NUM*), older(*NUM*): Require that the nLockTime/nSequence value is at least *NUM*. *NUM* cannot be 0.
- sha256(*HEX*), hash256(*HEX*): Require that the preimage of 64-character *HEX* is revealed. The special value H can be used as *HEX*.
- ripemd160(*HEX*), hash160(*HEX*): Require that the preimage of 40-character *HEX* is revealed. The special value H can be used as *HEX*.
- and(*POL*,*POL*): Require that both subpolicies are satisfied.
- or([*N@*]*POL*,[*N@*]*POL*): Require that one of the subpolicies is satisfied. The numbers N indicate the relative probability of each of the subexpressions (so 9@ is 9 times more likely than the default).
- thresh(*NUM*,*POL*,*POL*,...): Require that NUM out of the following subpolicies are met (all combinations are assumed to be equally likely).

Miniscript examples

A single key

Policy

pk(key_1)

Miniscript output:

pk(key_1)

Spending cost analysis

- Script: 35 WU
- Input: 73.000000 WU
- Total: 108.000000 WU

Resulting script structure

```
<key_1> OP_CHECKSIG
```

A 3-of-3 that turns into a 2-of-3 after 90 days

Policy

thresh(3,pk(key_1),pk(key_2),pk(key_3),older(12960))

Miniscript output:

thresh(3,pk(key_1),s:pk(key_2),s:pk(key_3),sln:older(12960))

Spending cost analysis

- Script: 122 WU
- Input: 166.250000 WU
- Total: 288.250000 WU

Resulting script structure

```
<key_1> OP_CHECKSIG OP_SWAP <key_2> OP_CHECKSIG OP_ADD OP_SWAP <key_3>
OP_CHECKSIG OP_ADD OP_SWAP OP_IF
  0
OP_ELSE
  <a032> OP_CHECKSEQUENCEVERIFY OP_0NOTEQUAL
OP_ENDIF
OP_ADD 3 OP_EQUAL
```

BLOCKS AND MINING

Problem: Who will include next block in blockchain?

- Transactions (state updates) has to be included somehow into block to be "permanently" valid
- Entity including new block has special position and power
  - Can decide which transactions (state updates) will be included
    - May lead to censorship of certain transactions
    - May lead to transactions reordering impacting the financial value (e.g. MEV)
  - Can decide where new block is appended
    - Shall be last previous block, but can cause malicious forks abandoning part of previously extended blockchain (e.g., 51% attack to rewrite history)
  - Typically receive some reward (motivation for participation)
    - May cause long-term centralized accumulation of underlying token
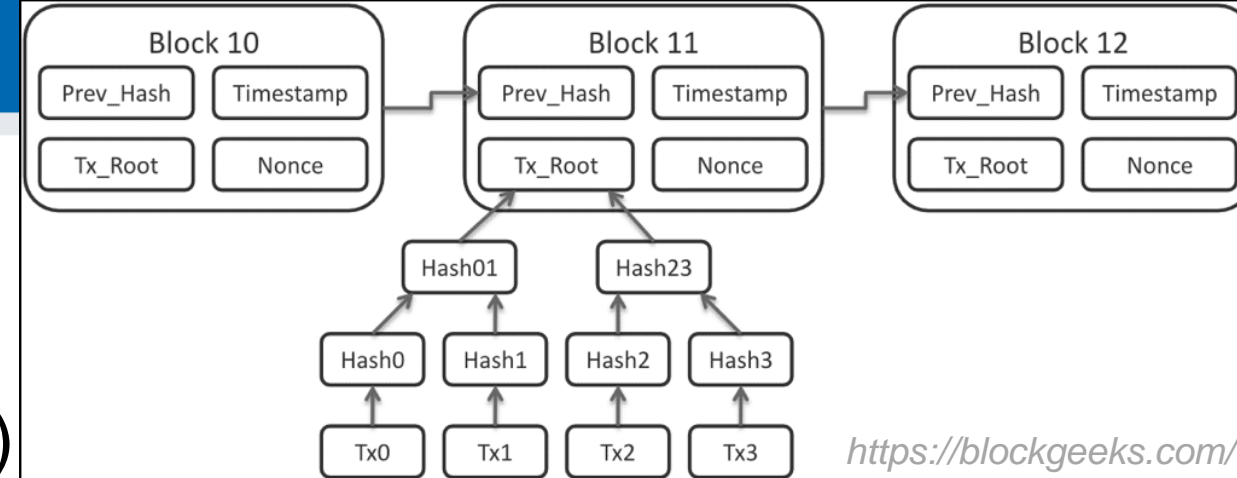
Who can include next block to blockchain?

> We will focus mainly on Proof Of Work used in Bitcoin

- Proof of Work (PoW, Bitcoin, Ethereum 1.0, Zcash…)
  - Solver of computationally hard puzzle can include new block
- Proof of Stake (PoS, Zcoin, Cardano, BNB, Ethereum 2.0…)
  - More coins you own, higher the probability you will be selected to include next block
  - Various variants, Stake pools…
- Merged Mining (Namecoin…)
  - Hash of block from the chain is included in coinbase tx of other chain (typically Bitcoin)
  - The chain is not performing own mining, Bitcoin miners are getting reward for inclusion of other chains
- Proof of Proof (PoP)
  - Hash of block from other chain is included in Bitcoin transaction (typically OP_RETURN)
  - Security of other chain is improved by security of Bitcoin blockchain
- Proof of Authority (PoA)
  - Small number of trusted actors create new blocks

Bitcoin block

https://blockgeeks.com/
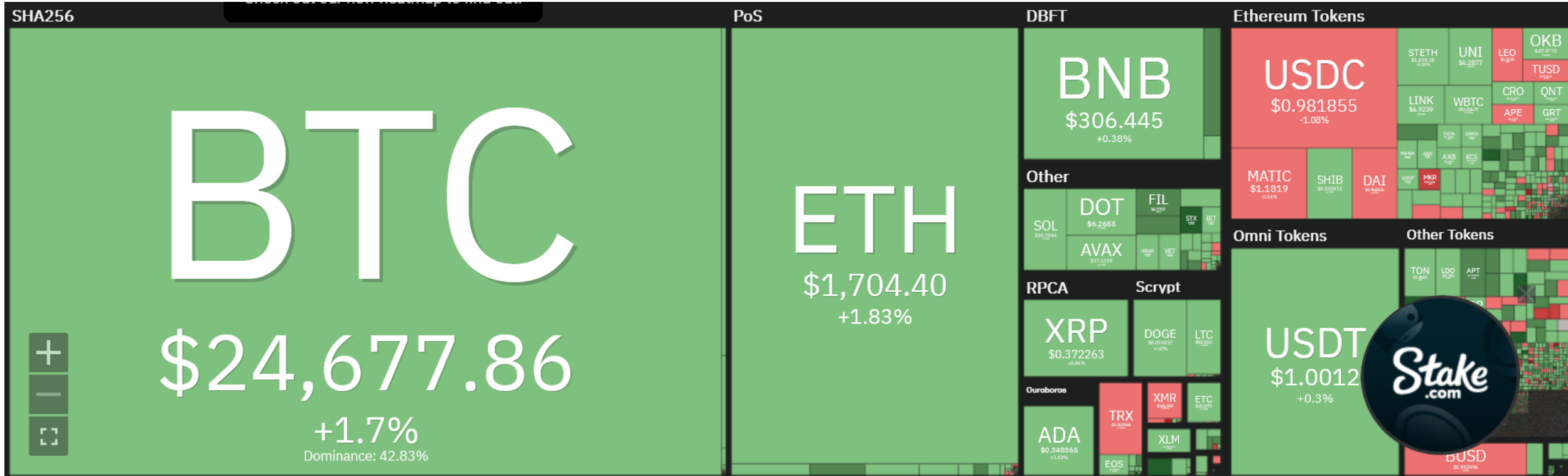
- Header (80 B) + data (up to ~4MB)
  - Version
  - Previous block hash (linking to past blockchain)
  - Merkle root of all included transactions (Coinbase tx + others)
  - Timestamp (unix time)
  - Bits (specification of required mining difficulty)
  - Nonce (variable part to mine , now insufficient)

- Coinbase transaction (reward for miners, emission of new bitcoins)
  - First transaction in every block (only one)
  - Only one input, previous TX ID = 0x0000..00, prev. TX index = 0xffffffff
  - (Typically) equal to block reward + all fees from included transactions

Coin mining algorithm https://coin360.com/

Mining in Proof of Work chains

- Crucial for security of blockchain (no rewrite of history)
- Initially on CPU (Satoshi: everyone can participate 1 CPU 1 vote)
- CPU→GPU →FPGA →ASIC
- Initially solo mining, later collaborative mining (too little chance alone)
- First mining pool: SlushPool in Prague (now Braiins Pool)
  - Miners join their hashrate, fraction of reward based on number of partial solutions
- Cambridge university centre for alternative finance (CBECI)
  - Where are the miners? https://cbeci.org/mining_map/
  - More mining details: https://cbeci.org/cbeci/methodology

Miner reward – coinbase output: block + fees

Shows the average coinbase transaction output amount.
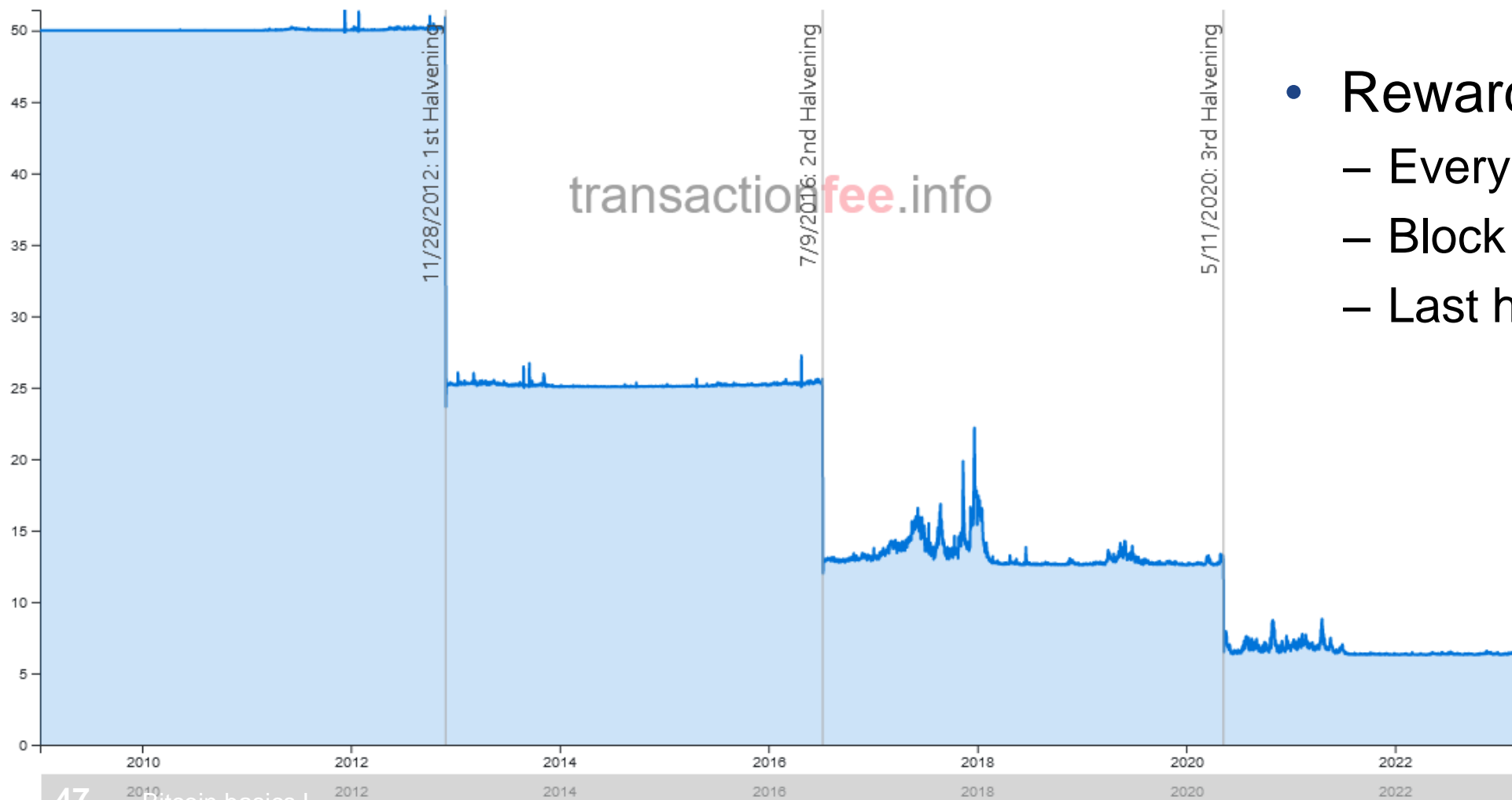
- Reward halving
  - Every ~4 years
  - Block reward drops to ½
  - Last halving in year 2140

Hashrate in time (~320EH/s = 3.2*10^{20} hash/sec = 2^{66}/sec)

https://mempool.space/graphs/mining/hashrate-difficulty#all

Blockchain forks

- Occasional natural forks happen
  - (not to be confused with softforks)
- Quickly resolved
  - usually, next block
- Sometimes temporary double-spent can occur
  - Same input used in different txs
- https://forkmonitor.info/nodes/btc

CR⊙CS

DEMO: SHOW DIFFICULTY ADJUSTMENT

Difficulty adjustment

- Bitcoin shall have one block every ten minutes (on average)
- Block must have overall hash with specific number of leading zeroes (March 2023 ~75 binary 0)
  - Miners change part of block header to try different hashes until required found
- How to specify the number of leading zeroes for decades in future?
  - Speed of new blocks found depends on the overall speed of hashing
  - Overall speed of hashing depends on technology advancements (single chip) and number of chips deployed
  - Impossible to predict technology and interest into distant future
  - If # zeroes is too low => blocks are found too fast (and vice versa)
- Idea of difficulty adjustment (part of consensus protocol), https://en.bitcoin.it/wiki/Difficulty
  - Check number of actually mined blocks every 2016 blocks (shall be ~14 days)
    - Increase/decrease difficulty for next period based on actual number of mined blocks
  - Every full node can deterministically compute expected difficulty (lower # zeroes rejected)
- Block hash must be below the "Target" number (computed to avg keep 1 block / ~10 min)
  - "Target" is transformed to "Bits" (condensed 4 bytes number – coefficient (3B) + exponent (1B))
  - Current difficulty is relative number of current Target with respect to Target of Genesis block

China mining dominance (09/2019→04/2020: 75.6%->65%)

09/2019

China	75.62%
Russia	5.93%
United States	4.06%
Malaysia	3.25%
Iran	1.74%
Kazakhstan	1.42%
Libya	1.16%
Canada	1.06%
Norway	0.95%
Germany	0.90%

Average monthly sha

04/2020

China	65.08%
United States	7.24%
Russia	6.90%
Kazakhstan	6.17%
Malaysia	4.33%
Iran	3.82%
Canada	0.82%
Germany	0.56%
Norway	0.48%
Venezuela	0.42%

Average monthly share of total hashrate

CR⊙CS
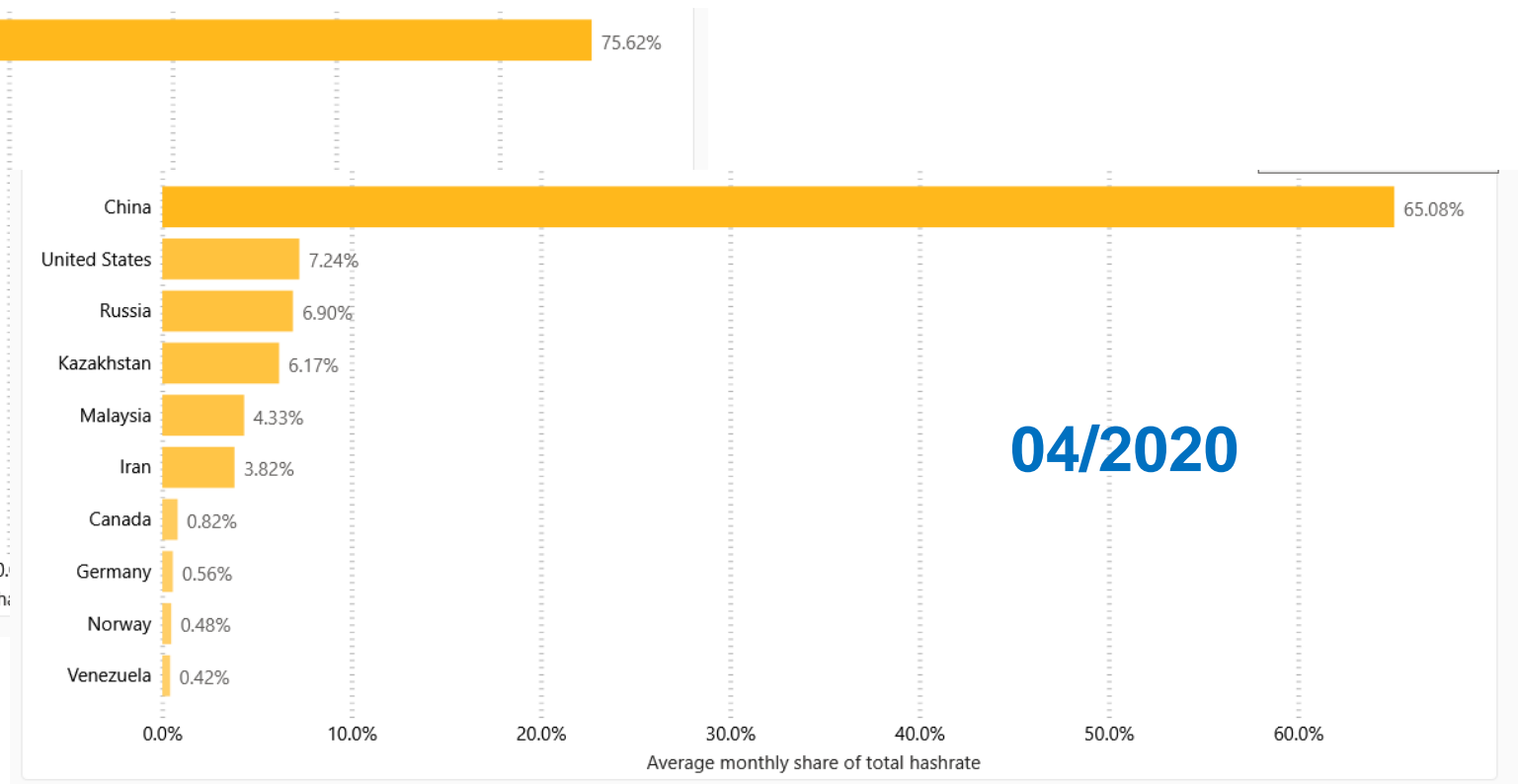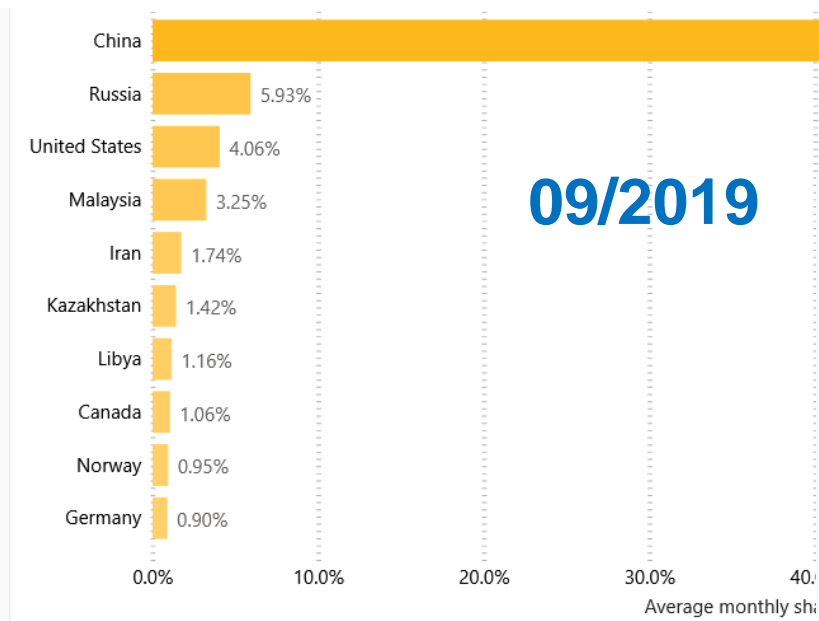
Bitcoin mining map (January 2022)

Evolution of network hashrate

Estimated absolute hashrate (monthly average) Exahashes per second (Eh/s)

- Mainland China
- Malaysia
- Germany *
- Other
- Iran, Islamic Rep.
- Ireland *
- Russian Federation
- Kazakhstan
- United States
- Canada
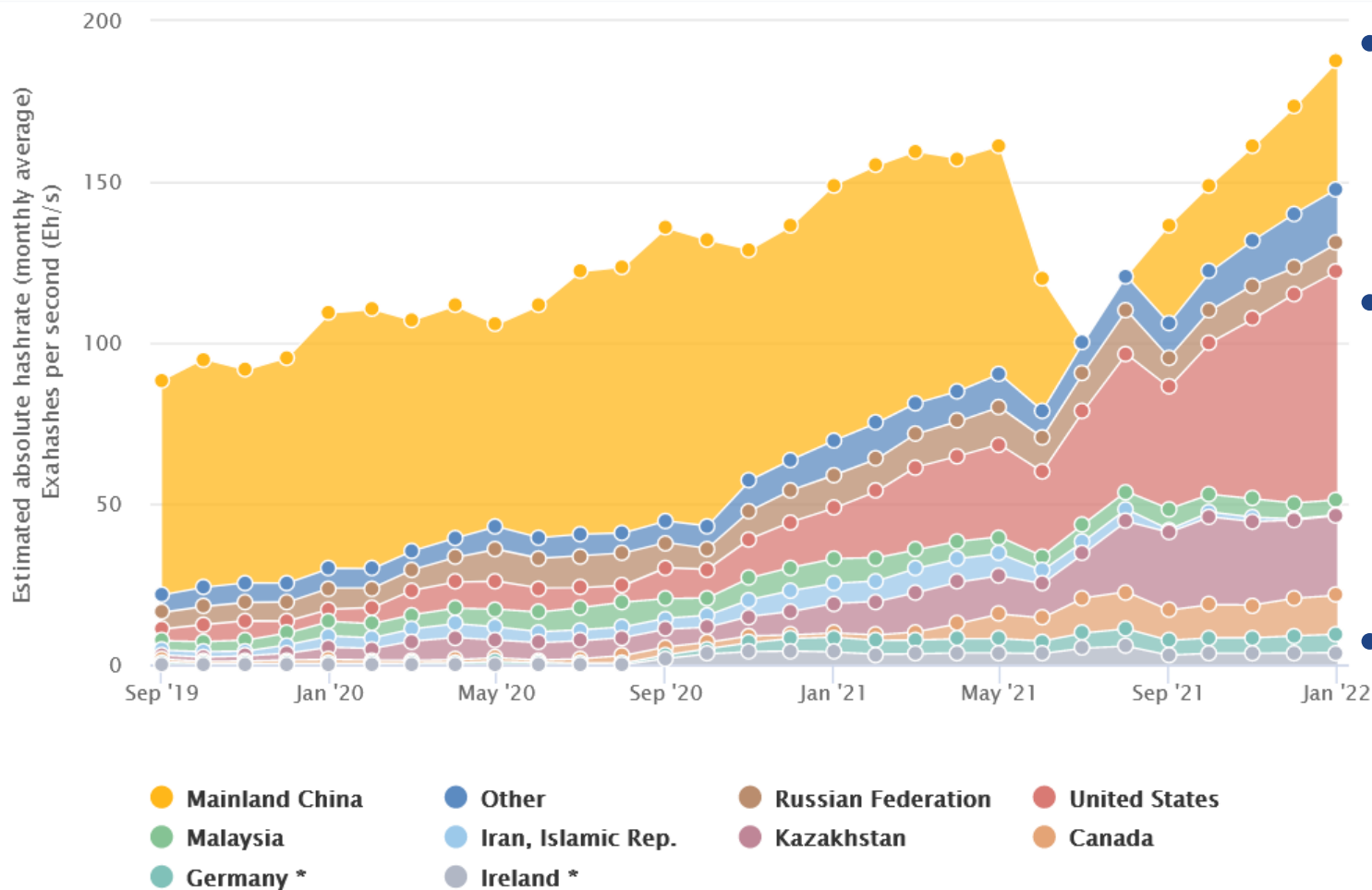
- China used to be largest
  – >80% (till 2018, slow decrease)
  – Mining ASICS made in China
- China evicted "all" miners in May 2021
  – Officially 0% (unofficially still active)
  – Now coming back 21.11%
- Resulted in strong increase in:
  – US 37.84%, Kazakhstan 13.22%
  – Canada 6.48%, other 9% …

Is Bitcoin mining wasteful?

- Heavily discussed topic ("Bitcoin will boil the oceans by 2020")
- Some questions to ask (**Do your own research!**)
  - What value you are getting for the energy expended? (neutral decentralized monetary system)
  - Miners want the cheapest energy available to maximize profits => energy nobody wants => waste energy
  - What is the source of the energy used? (btc mining 60-70% "green" energy due to its low cost)
  - Can mining help to stabilize electrical grid with intermittent (solar, wind) sources? (instant turn on/off of mining ASICs, consumption of only cheap (= not demanded) energy)
  - How long is mining hardware profitable before dismantling? (depends on energy price, 5+ years)
  - Can miners finance construction of energy sources (hydro…) at places otherwise not viable financially (stranded energy)?
  - Can miners incentivize higher portion of intermittent (solar, wind) sources? (bigger source even when low sun/wind?)

Interesting stats about mined transactions

- https://forkmonitor.info/nodes/btc
- https://transactionfee.info/
- https://cryptobriefing.com/unpacking-bitcoins-recent-double-spend-event

CR🔘CS

Warning: Why not put "blockchain" everywhere?

- ~~"Blockchain not Bitcoin", "Blockchainize everything"~~… claims
- Permissionless distributed consensus on global state is very expensive
  – Confirmation time, storage space, energy expenditure (PoW)…
  – Most applications does not need it!
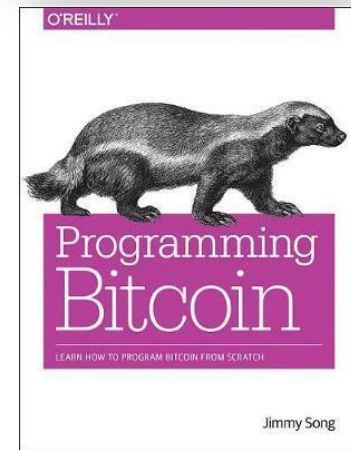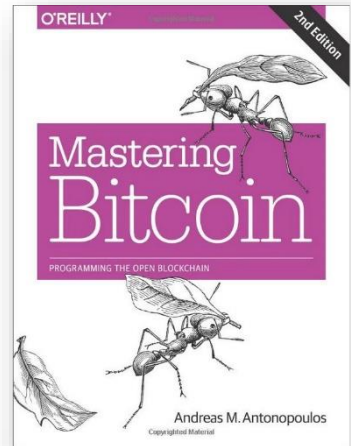 – Especially when other components of application are centralized (development, governance decisions, data storage…)

BLOCKCHAIN?

THAT JUST SOUNDS LIKE A LINKED LIST WITH EXTRA STEPS

makeameme.org

Study materials

- Mandatory reading
  - Bitcoin's academic pedigree (Arvind Narayanan, Jeremy Clark)
    - https://dl.acm.org/doi/10.1145/3132259 (copy in IS)
    - Explanation of roots of Bitcoin key components
- If you were not familiar with basics of Bitcoin before
  - Watch 'But how does bitcoin actually work?' by 3Blue1Brown (26min)
    - https://www.youtube.com/watch?v=bBC-nXj3Ng4
  - Read slides Hello Bitcoin (including notes under every slide)
    - From https://www.hellobitco.in/, copy of slides in IS

Further reading

- Mastering Bitcoin (Andreas M. Antonopoulos and others)
  - https://github.com/bitcoinbook/bitcoinbook
- Programming Bitcoin (Jimmy Song)
  - https://github.com/jimmysong/programmingbitcoin
- List of interesting resources
  - https://blockonomi.com/bitcoin-educational-resources/
  - https://learnmeabitcoin.com/, https://learnmeabitcoin.com/technical/

THANK YOU FOR COMING, SEE YOU NEXT WEEK