Cryptocurrencies

Lecture 4: Bitcoin

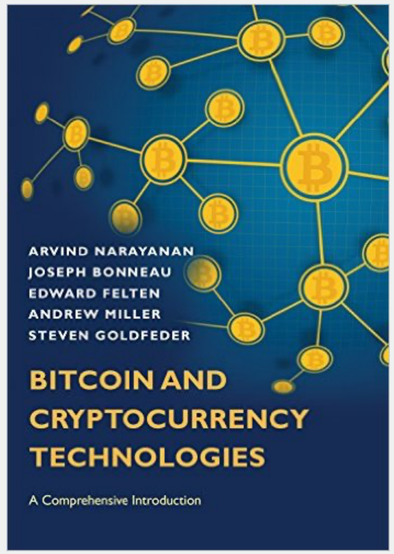
Matteo Maffei
matteo.maffei@tuwien.ac.at

November 12, 2025

Overview of today's lecture

Bitcoin

- Nakamoto consensus
- Network
- Transactions
- Data structures & blocks
- Mining



https://bitcoinbook.cs.princeton.edu/

Bitcoin

Nakamoto’s idea

- The impossibility results of Lecture 2 tell you more about what you cannot do in a model, rather than what you cannot do in reality 😉

- Nakamoto’s idea is two-fold
  - Make consensus probabilistic, thereby bypassing the impossibility result
  - Use “work” as an unpredictable source of randomness
  - Leverage the financial nature of the system to incentivize nodes to behave honestly

Nakamoto’s goals

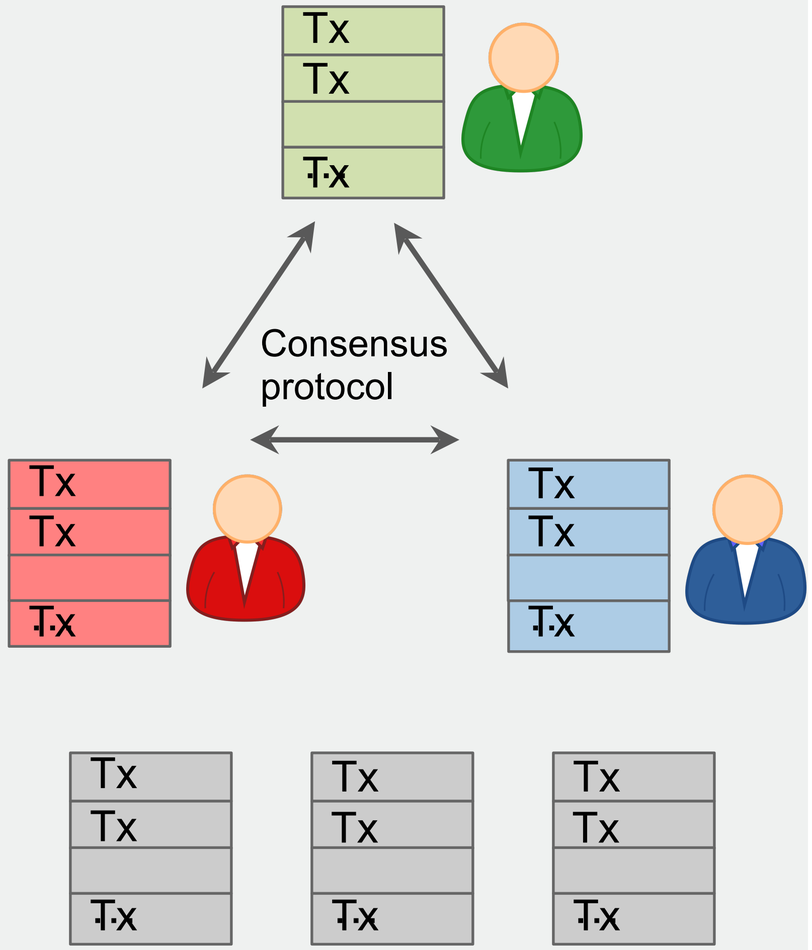
- Consensus on the order book
  - *all* transactions performed by users will be eventually included (liveness)

- Permissionless
  - anyone can participate in the consensus
  - possibly millions of processes (*scalability*)
  - an attacker can have multiple identities (*sybil attack*)

- Anonymity
  - no certified identity, so that for each transaction sender and receiver’s anonymity is preserved

Key idea: implicit consensus

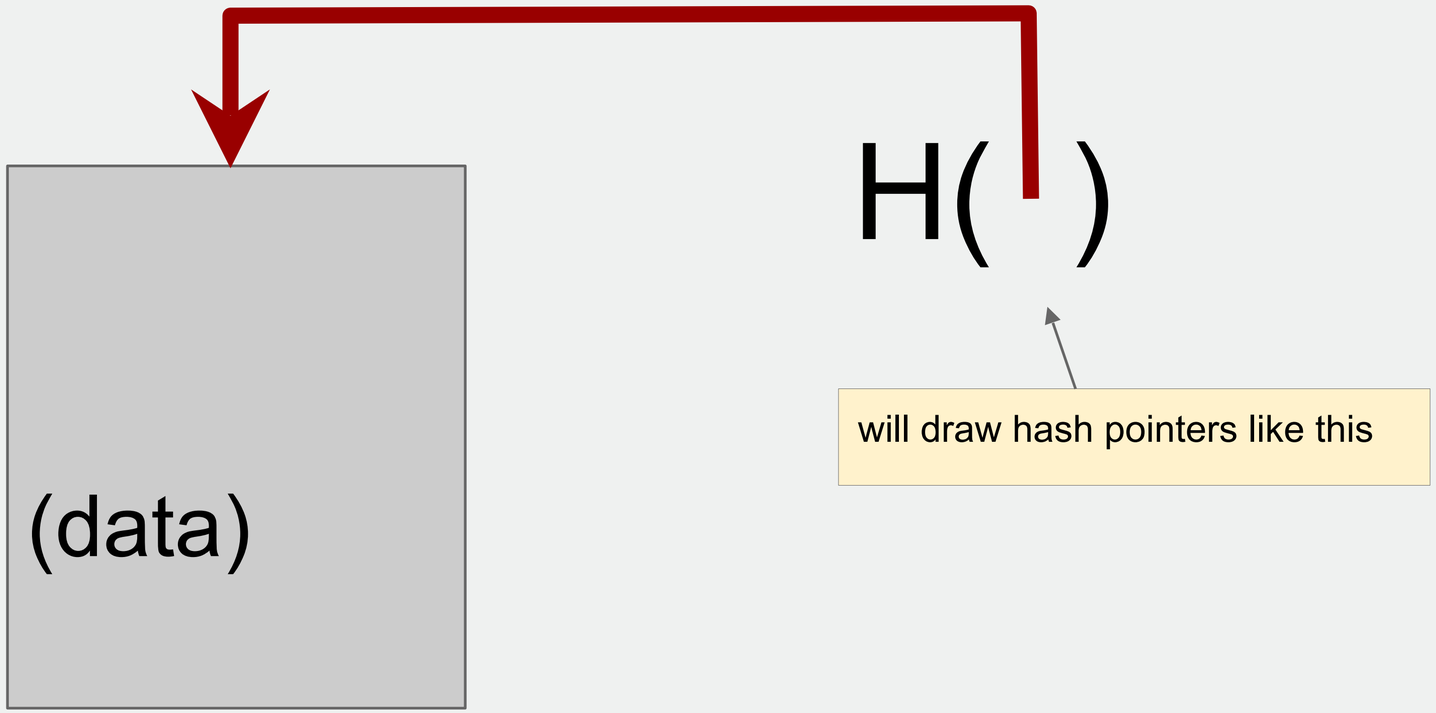
- In each round, a random node is picked (we will see how…)
- This node proposes the next block in the chain
- Other nodes implicitly accept/reject this block
  - by either extending the chain from that block (we will see how…)
  - or ignoring it and extending chain from earlier block



Detour:
Hash Pointers and Data Structures

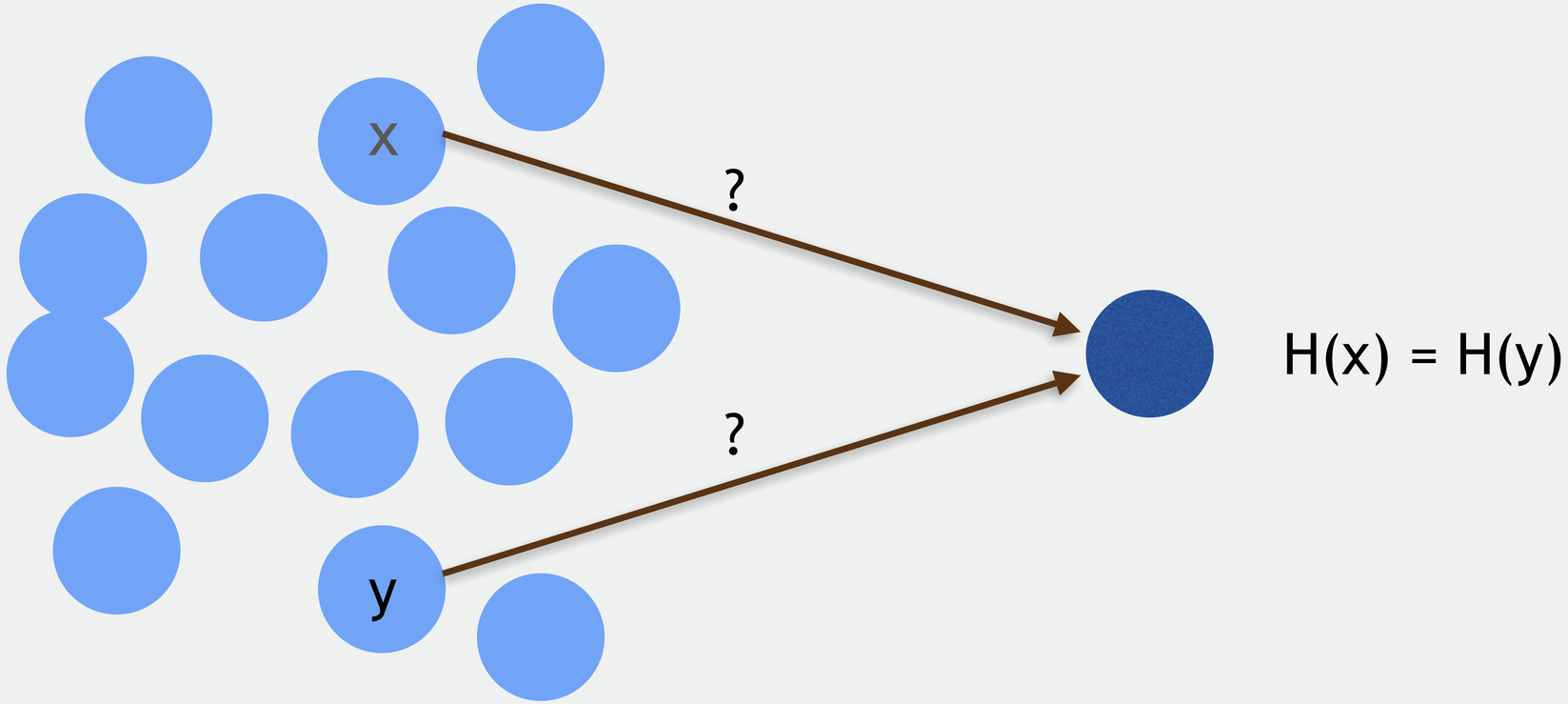
TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Hash Pointers



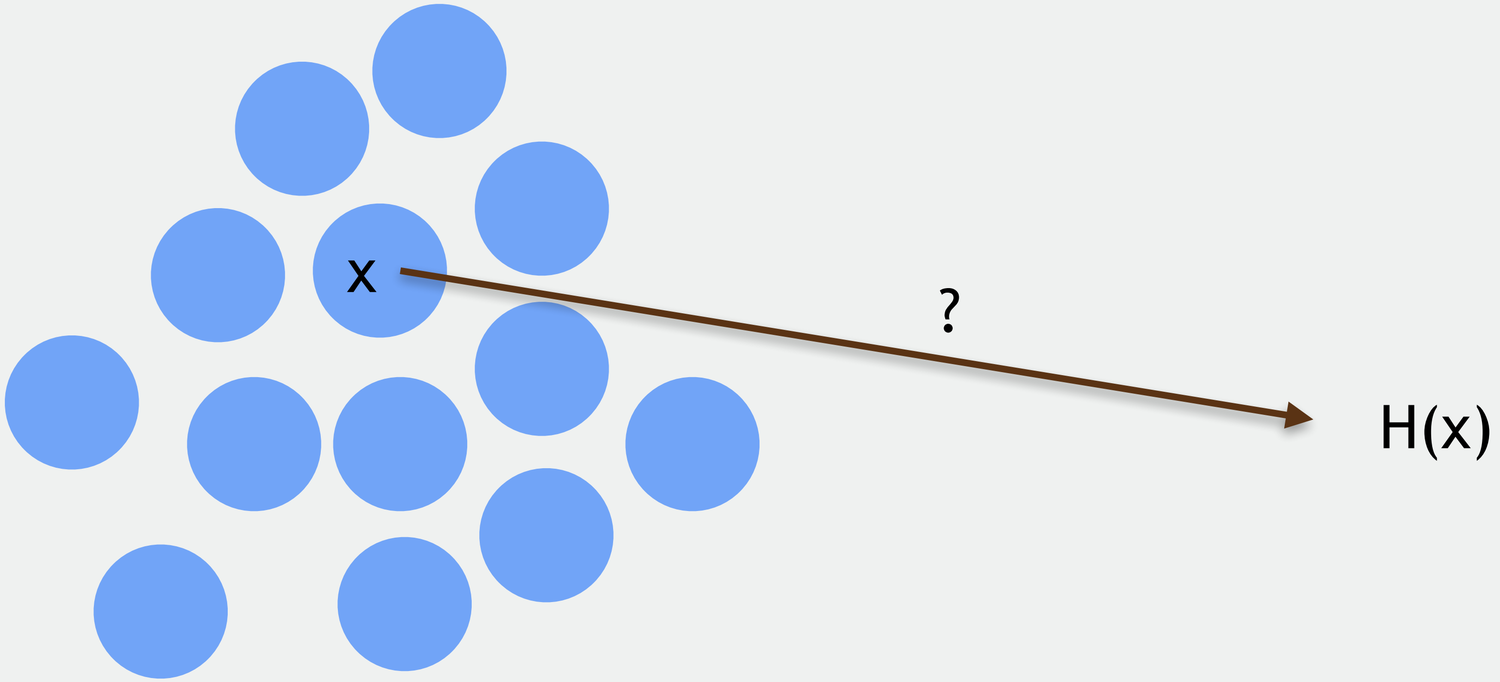
Hash property 1: Collision-free

- Nobody can find x and y such that x != y and H(x) = H(y)



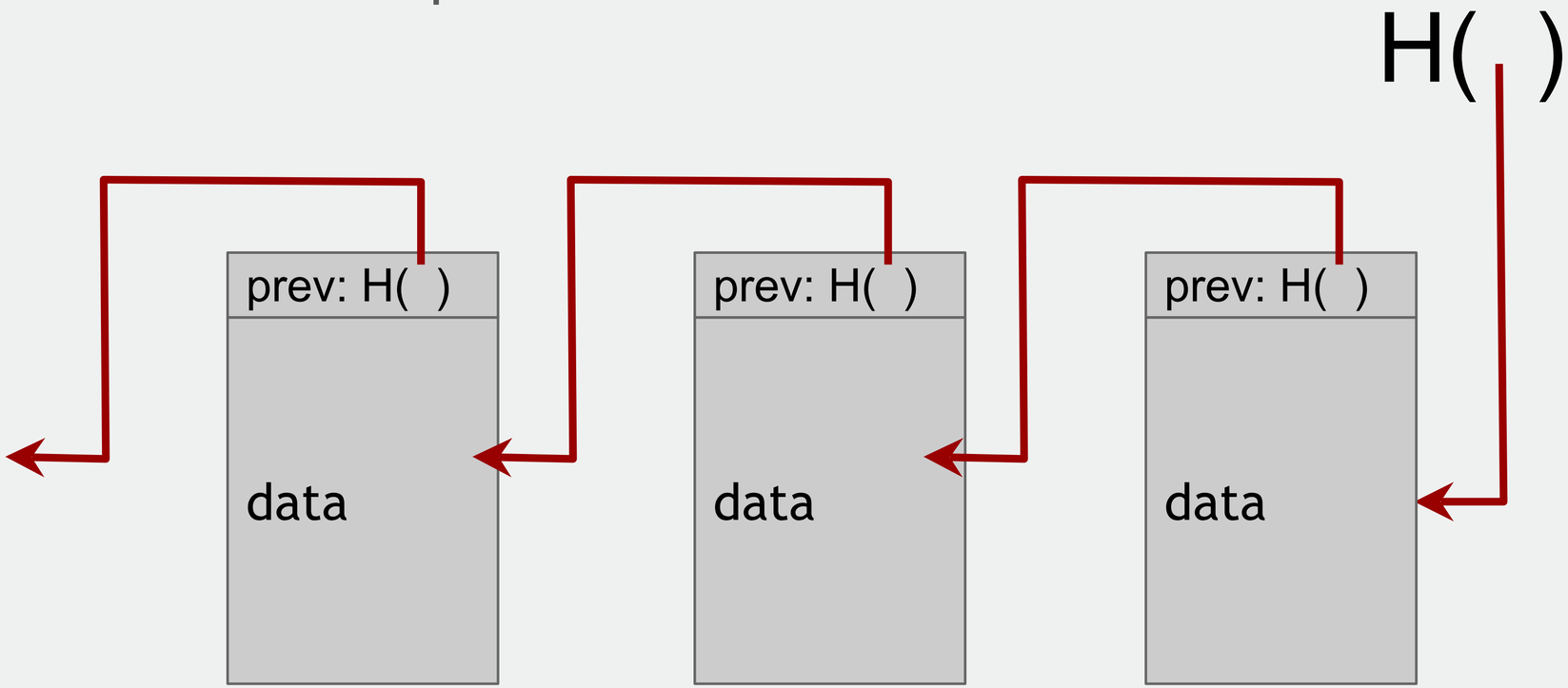
Hash property 2: Hiding

- Given H(x), it is infeasible to find x.



Block chains

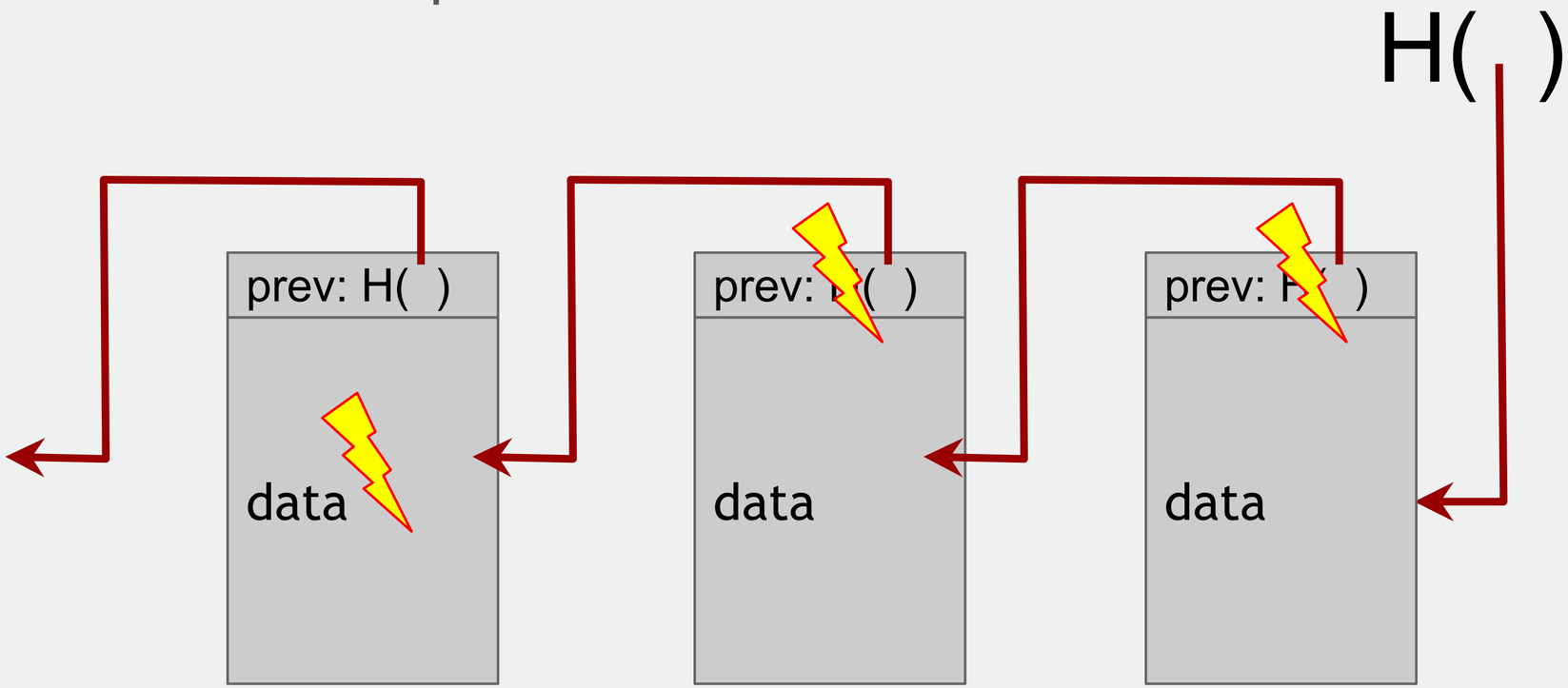
Linked list with hash pointers = “block chain”



use case: tamper-evident log

Block chains

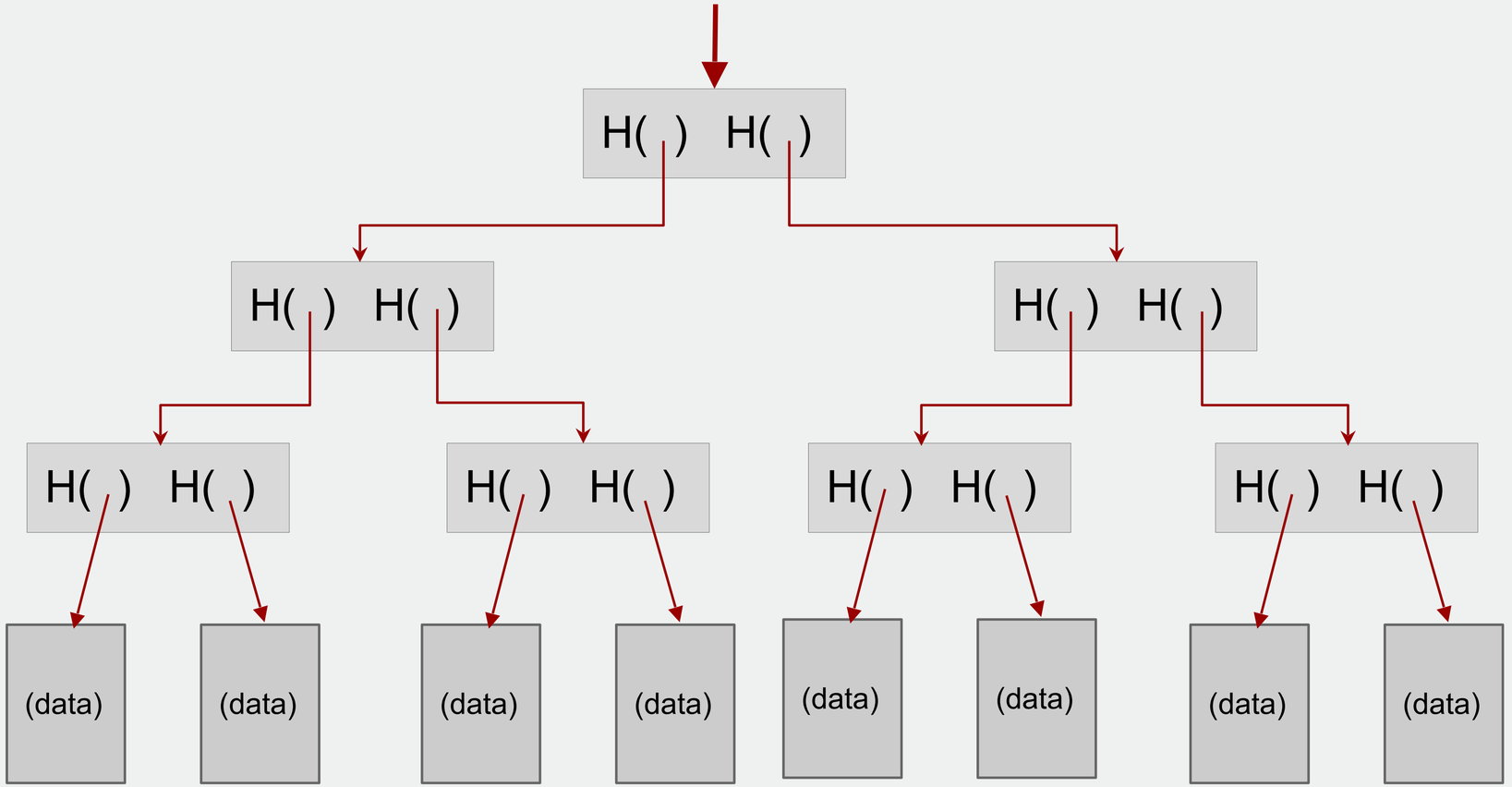
Linked list with hash pointers = “block chain”



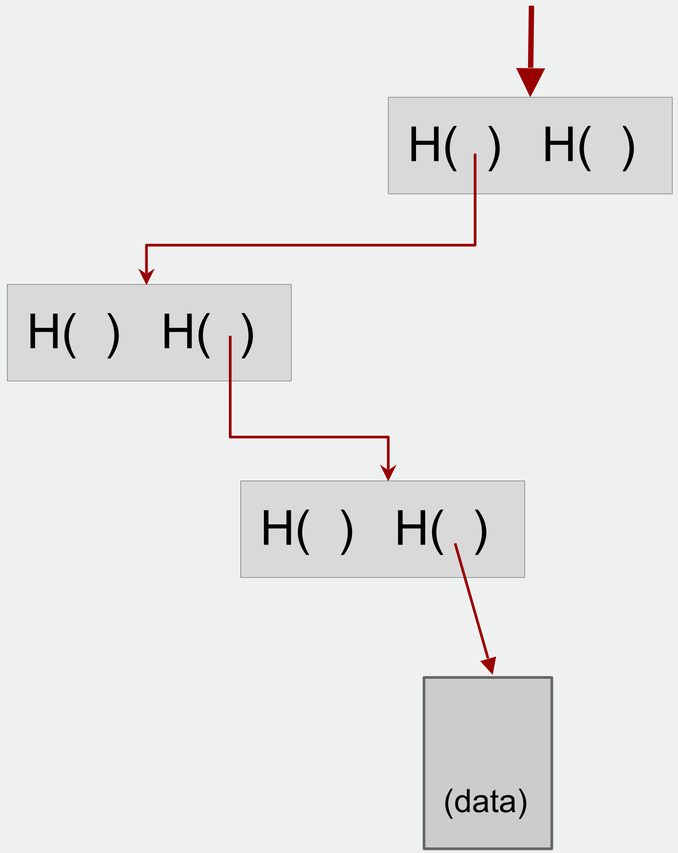
use case: tamper-evident log

Merkle trees

binary tree with hash pointers = “Merkle tree”



Proving membership



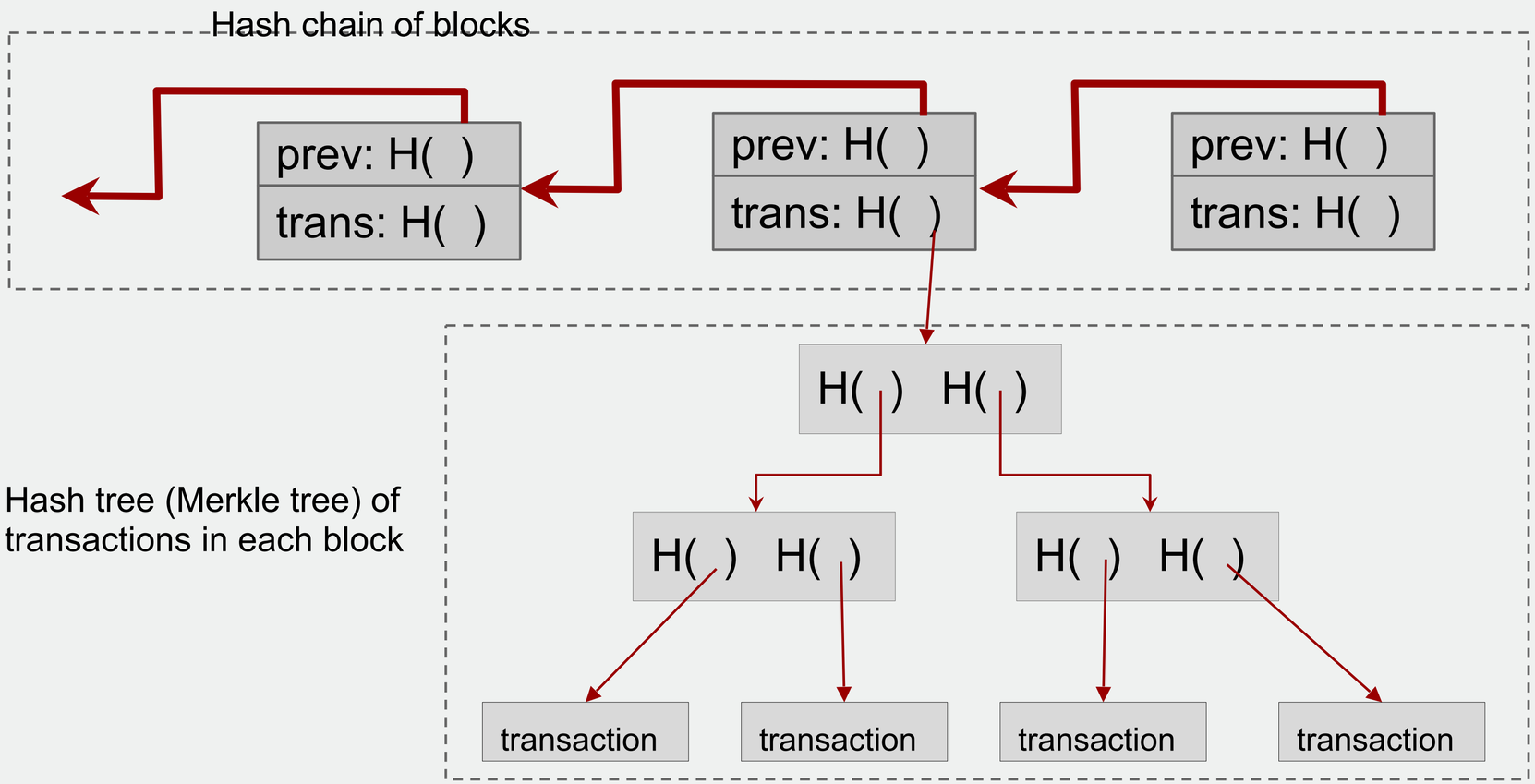
show O(log n) items

Advantages of Merkle trees

- Tree holds many items,
  but just need to remember the root hash
- Can verify membership in O(log n) time/space

- Variant: sorted Merkle tree
  - can verify non-membership in O(log n)
    (show items before, after the missing one)

Bitcoin block structure

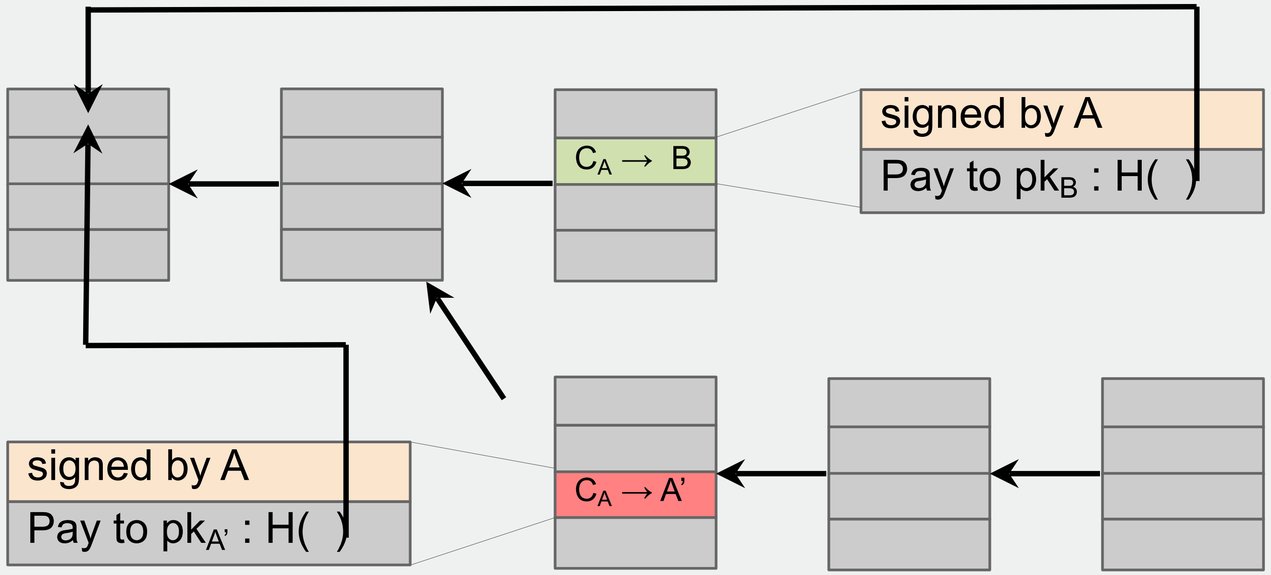


Hash tree (Merkle tree) of transactions in each block

Nakamoto consensus (simplified)

1. New transactions are gossiped to all nodes
2. Each node collects new transactions into a block
3. In each round a random node gets to gossip its block
4. Other nodes accept the block only if its transactions are all valid (unspent, valid signatures) and extend the longest chain
5. Nodes express their *acceptance of the block by including its hash in the next block* they create (extending the longest chain)

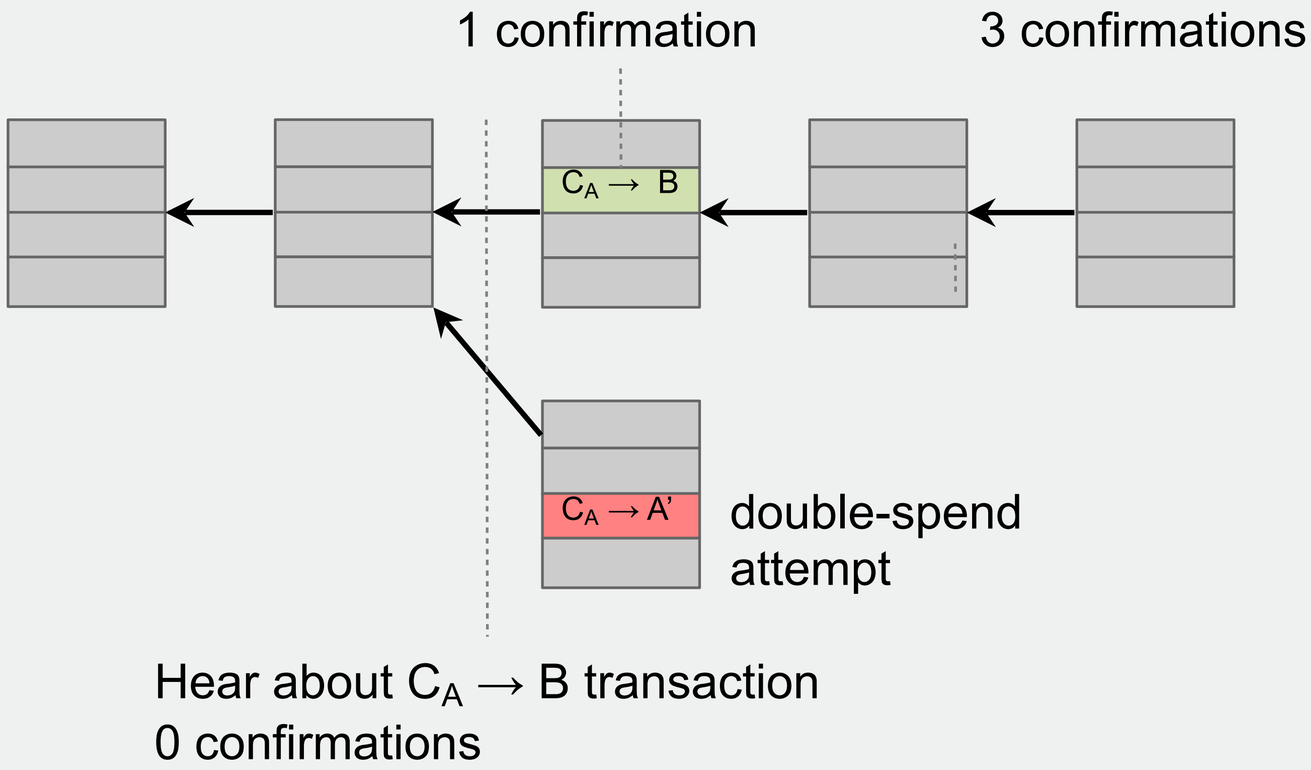
Longest chain consensus



Double-spend attack

- Which of the two transactions should be accepted? Either or them, but not both
- Nodes will extend the longest valid chain

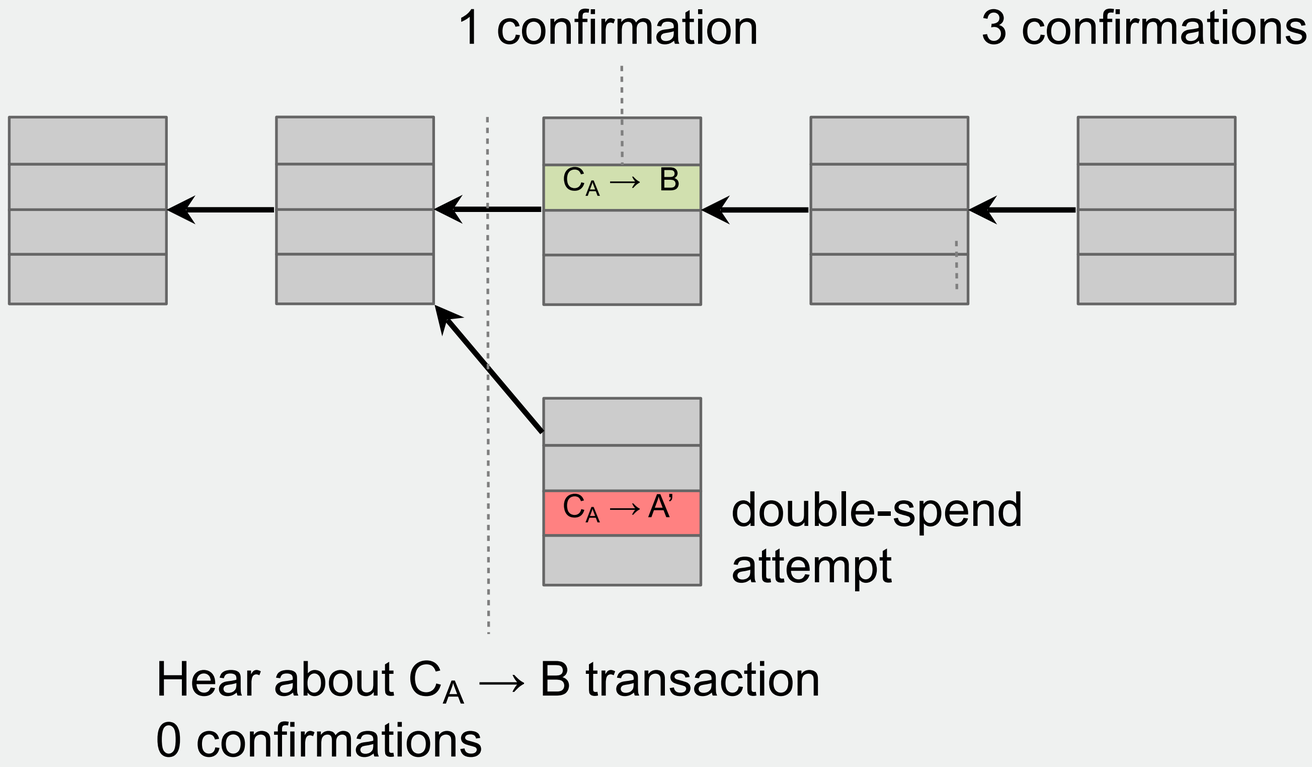
Example



Double-spend probability decreases exponentially with no. of confirmations.
Most common heuristic: 6 confirmations.

To wrap up…

- Protection against invalid transactions is cryptographic, but enforced by consensus
- Protection against double-spending is purely by consensus
- You're never 100% sure a transaction is in consensus branch. Guarantee is probabilistic



Remaining problems

Random leader selection

To approximate selecting a random node:

- select nodes in proportion to a resource that no one can monopolize (we hope)
  - In proportion to computing power:
    Proof-of-Work
  - In proportion to stake ownership:
    Proof-of-Stake
  - In proportion to the allocated disk space:
    Proof-of-Space

Expanded in Lectures 6, 7, 8

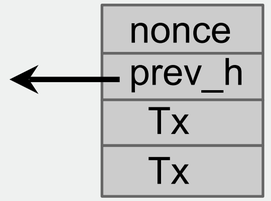
Proof of Work

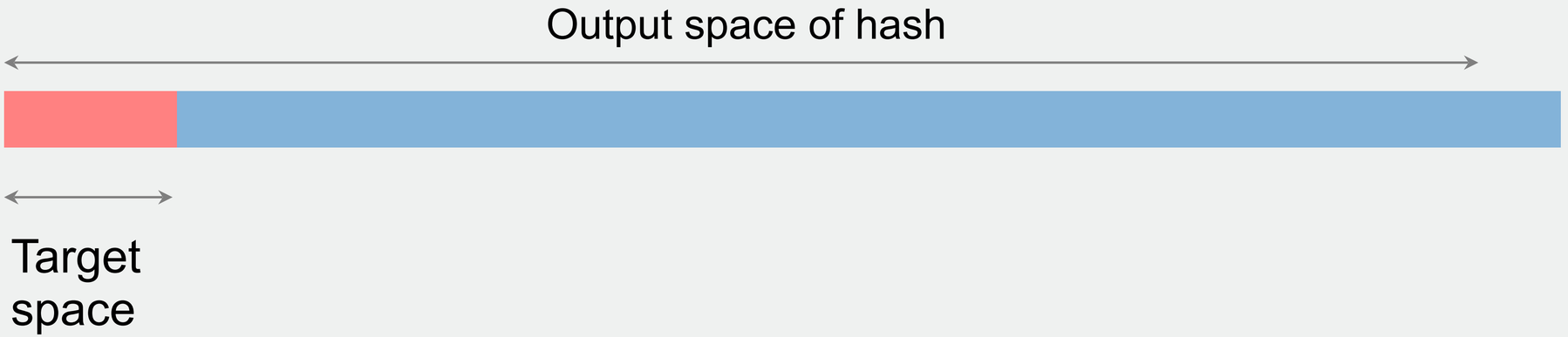
TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Hash puzzles

To create block, find nonce s.t.

H(nonce ‖ prev_hash ‖ tx ‖ … ‖ tx) is very small



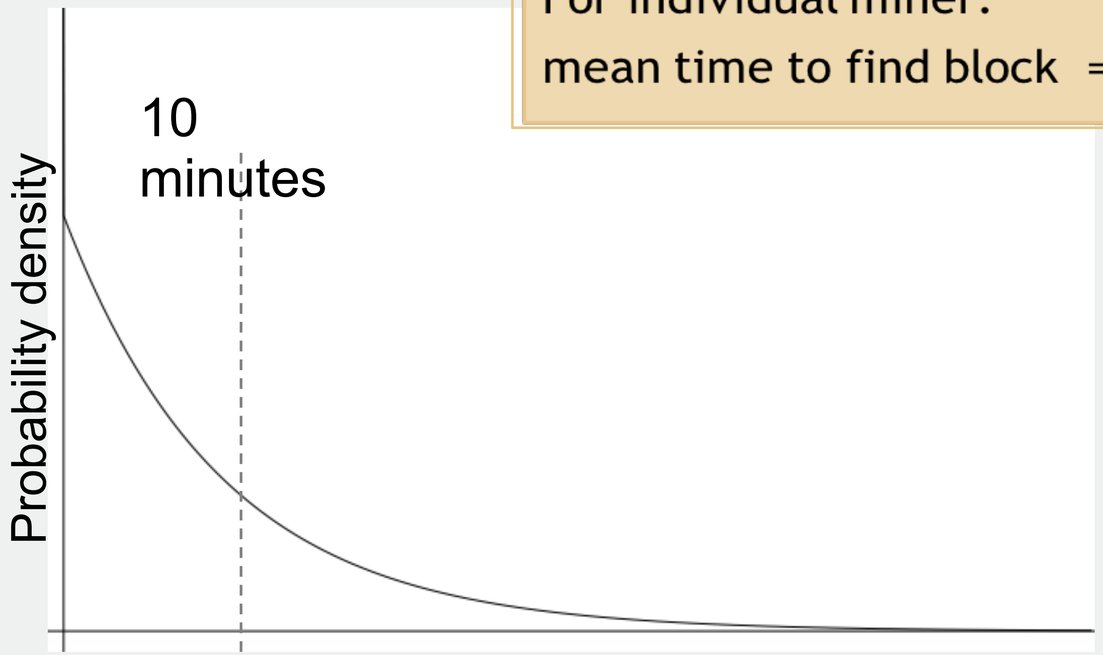


If the hash function is secure:
the only way to succeed is to try nonces until you get lucky
(brute force)

Solving hash puzzles is probabilistic

For individual miner:

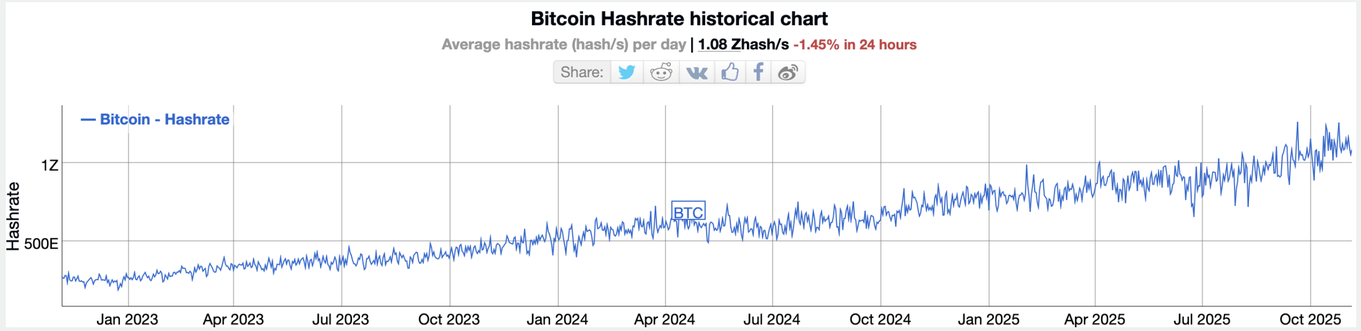
mean time to find block $= \frac{10 \text{ minutes}}{\text{fraction of hash power}}$

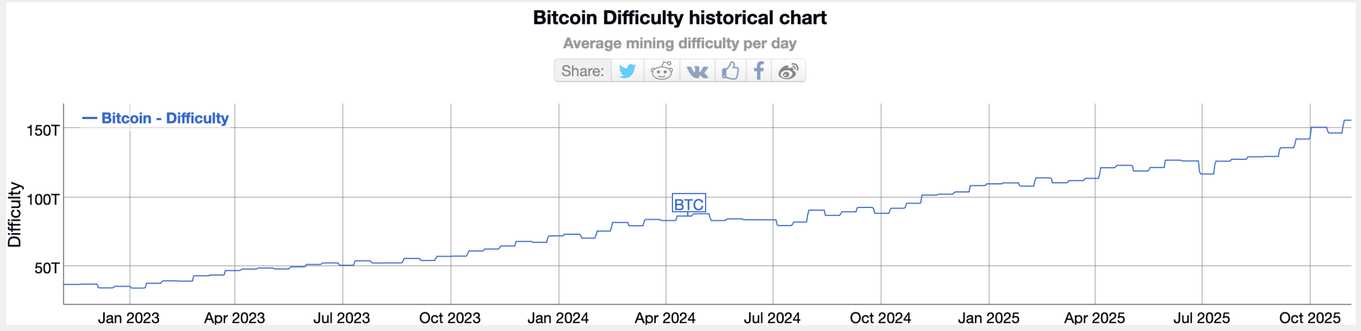


Time to next block (entire network)

PoW property 1: difficult to compute

- Nov 2025, BTC hashrate: ~ $1.08 \cdot 2^{21}$ hash/second
- Nov 2025, BTC difficulty: ~ 156 T
- Expected block time: difficulty x 2^{32} / hashrate





TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

PoW property 2: parameterizable cost

- Nodes automatically re-calculate the target every two weeks
- Goal: average time between blocks = 10 minutes

Prob (Alice wins next block) =
fraction of global hash power she controls

PoW property 3: trivial to verify

- Nonce must be published as part of block
- Other miners simply verify that
  $H(\text{nonce} \| \text{prev\_hash} \| \text{tx} \| \ldots \| \text{tx}) < \text{target}$

Bitcoin protocol

Security assumptions and guarantees

Blockchain = append-only data structure

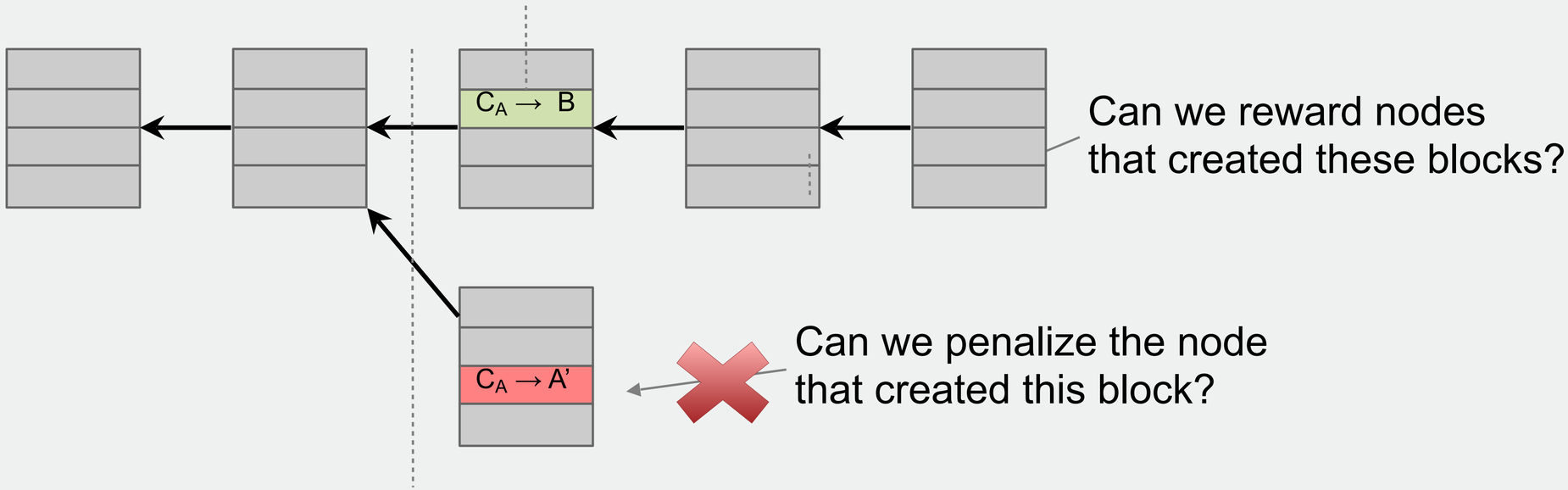
If the majority of miners weighted by hash power
follow the protocol, the blockchain:

(i) is safe: all nodes agree on the same blockchain (order of blocks/ transactions)
(ii) is live: the blockchain grows with honestly-created blocks

For proofs, see Lecture 5

Assumption of honesty is problematic

Can we give nodes <u>incentives</u> for behaving honestly?



Everything so far is just a distributed consensus protocol

But now we utilize the fact that the currency has value

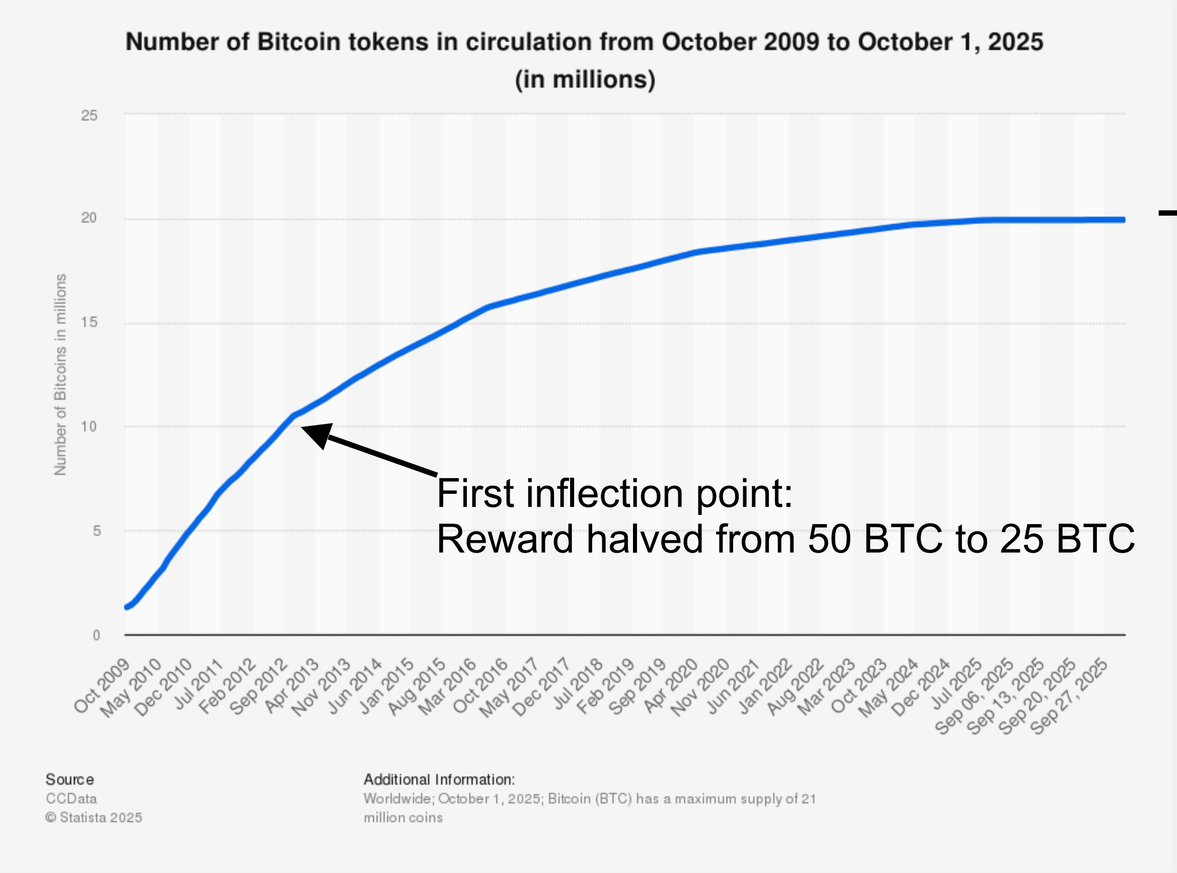
Incentive 1: block reward

- Creator of block gets to
  - include special coin-creation transaction in the block
  - choose recipient address of this transaction (included in the hash puzzle)

- Value is fixed: currently 3.125 BTC, halves every 4 years

- Block creator gets to “collect” the reward only if the block ends up on long-term consensus branch!

There’s a finite supply of bitcoins



→ Total supply: 21 million

Block reward is how new bitcoins are created

Runs out in 2140. No new bitcoins unless rules change

Incentive 2: transaction fees

- Creator of transaction can choose to make output value less than input value
- Remainder is a transaction fee and goes to block creator
- Purely voluntary, like a tip (in the USA!)
- Yet…transaction fees determine your “wait time” in the transaction queue (mempool)!

Mining Bitcoin in 6 easy steps

1. Join the network, listen for transactions
   a. Validate all proposed transactions

2. Listen for new blocks, maintain block chain
   a. When a new block is proposed, validate it
3. Assemble a new valid block
4. Find the nonce to make your block valid
5. Hope everybody accepts your new block
6. Profit!

Mining economics

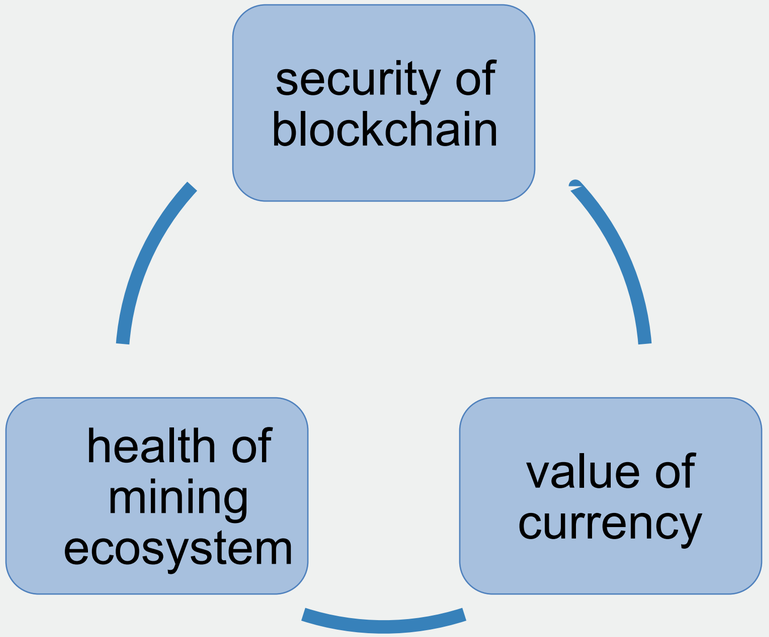
If mining reward (block reward + Tx fees)	>	hardware + electricity cost	→	Profit

Expanded in Lecture 7

Complications:

- fixed vs. variable costs
- reward depends on global hash rate

Bitcoin Security-Economics

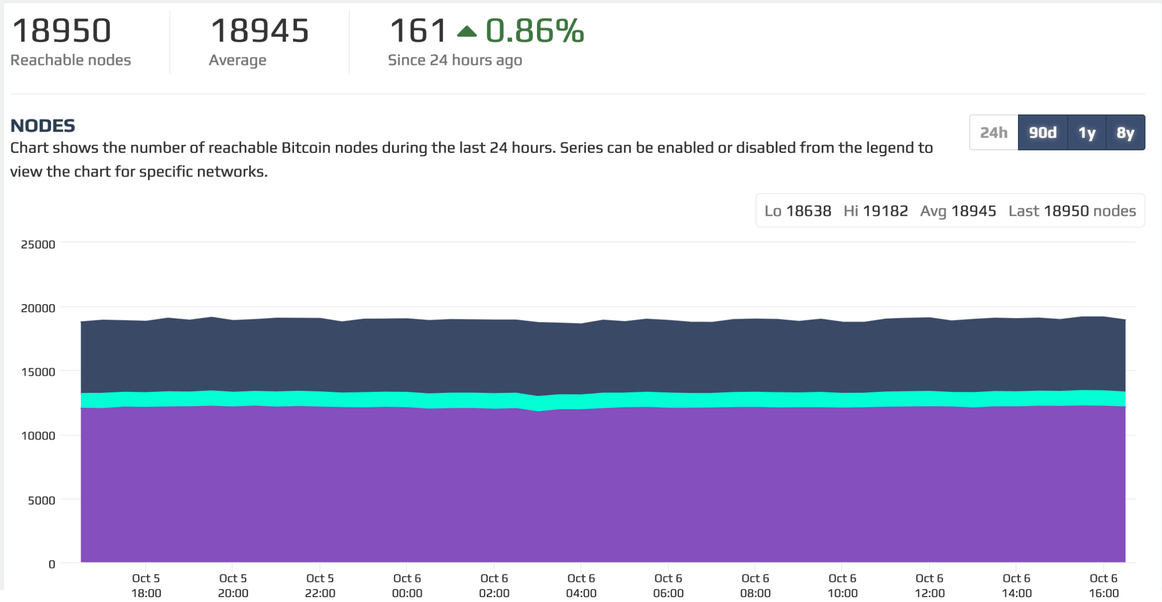


What can a “51% attacker” do?

- Steal coins from existing address? ✗
- Suppress some transactions? ✓
  - From the block chain ✗
  - From the P2P network
- Change the block reward? ✗
- Destroy confidence in Bitcoin? ✓✓

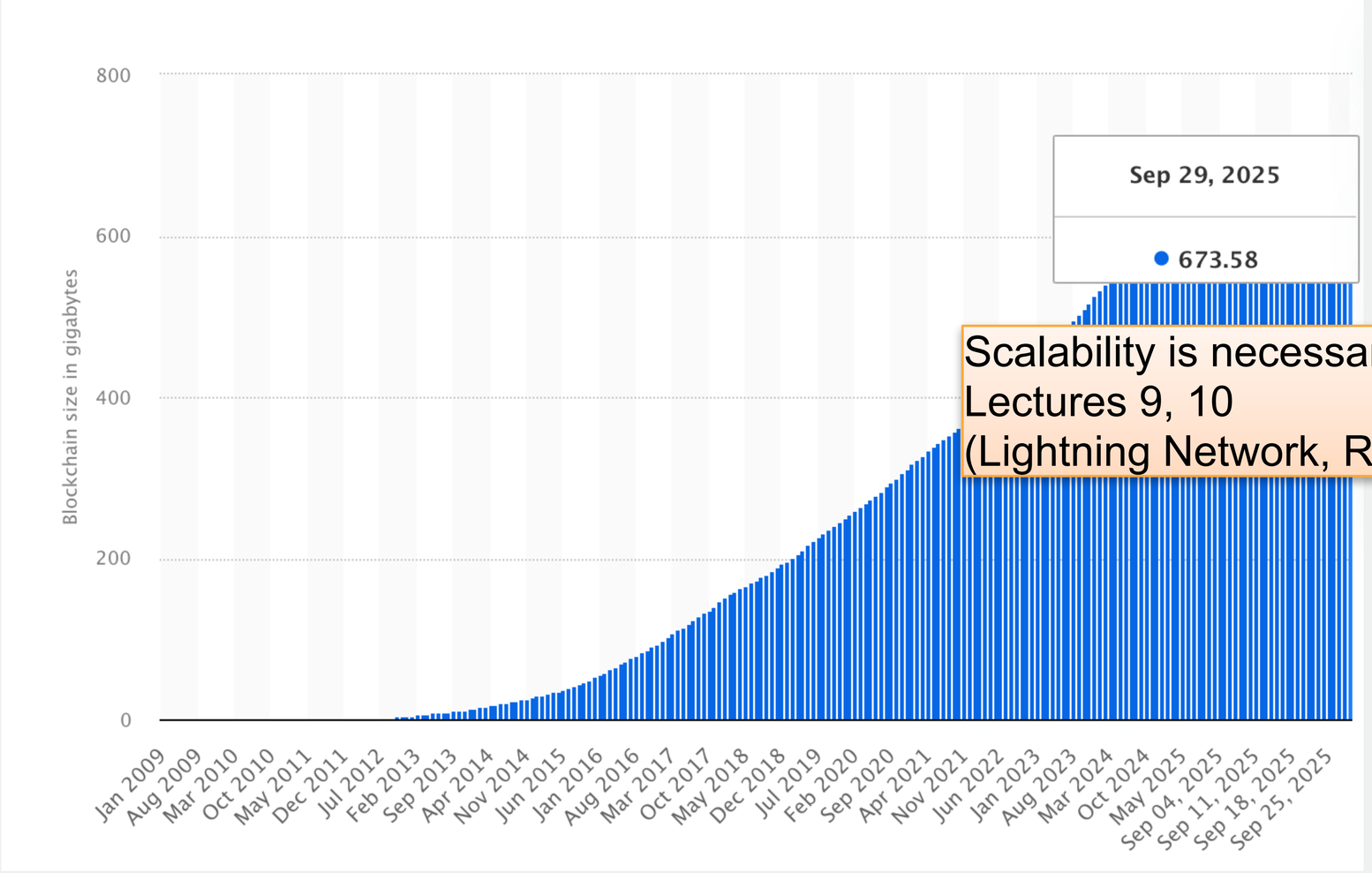
How big is the network?

- Impossible to measure exactly
- Estimates-up to 1M IP addresses/month
- Only about 18k “full nodes”
  - Permanently connected
  - Fully-validate, hear and store every transaction



Storage costs

September 2025: >600 GB



Scalability is necessary: see Lectures 9, 10 (Lightning Network, Rollups, etc.)

Key takeaways

- Consensus in permissionless setting
  - longest chain rule
  - no PKI needed, PoW is enough
  - incentives for honest behaviour
  - consensus over long time scales (about 1 hour) and probabilistic
  - weak form of anonymity, other blockchains provide stronger privacy guarantees

Expanded in Lecture 8

Bitcoin…under the hood!

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

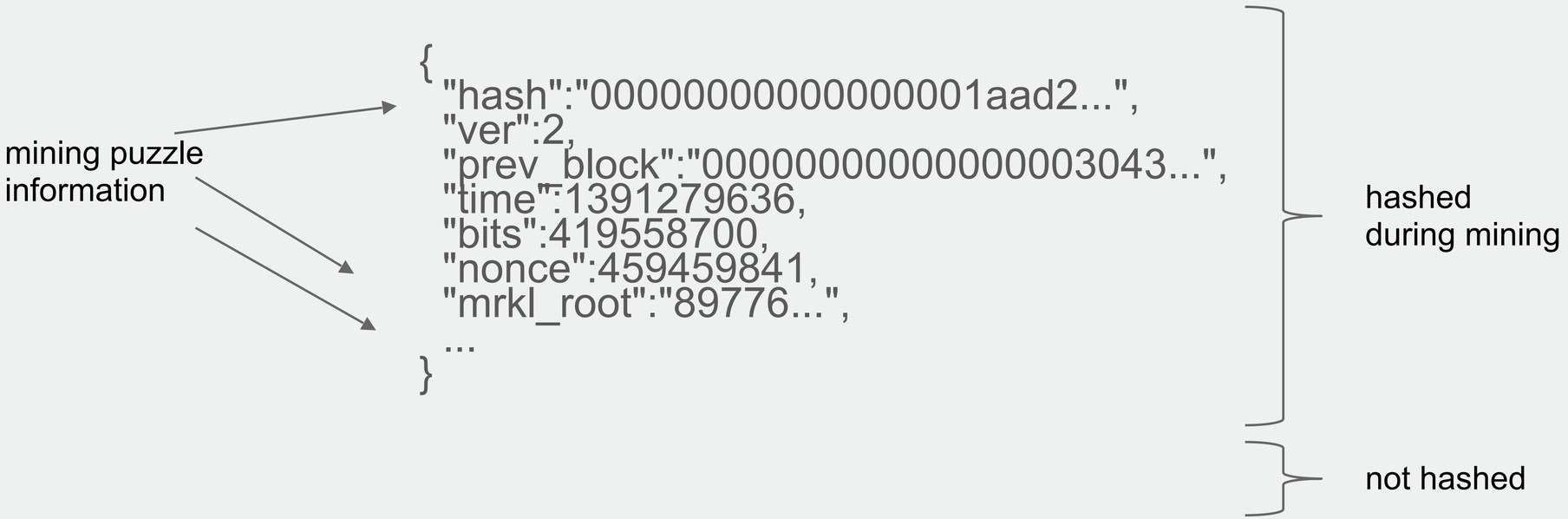
The real deal: a Bitcoin block

block header

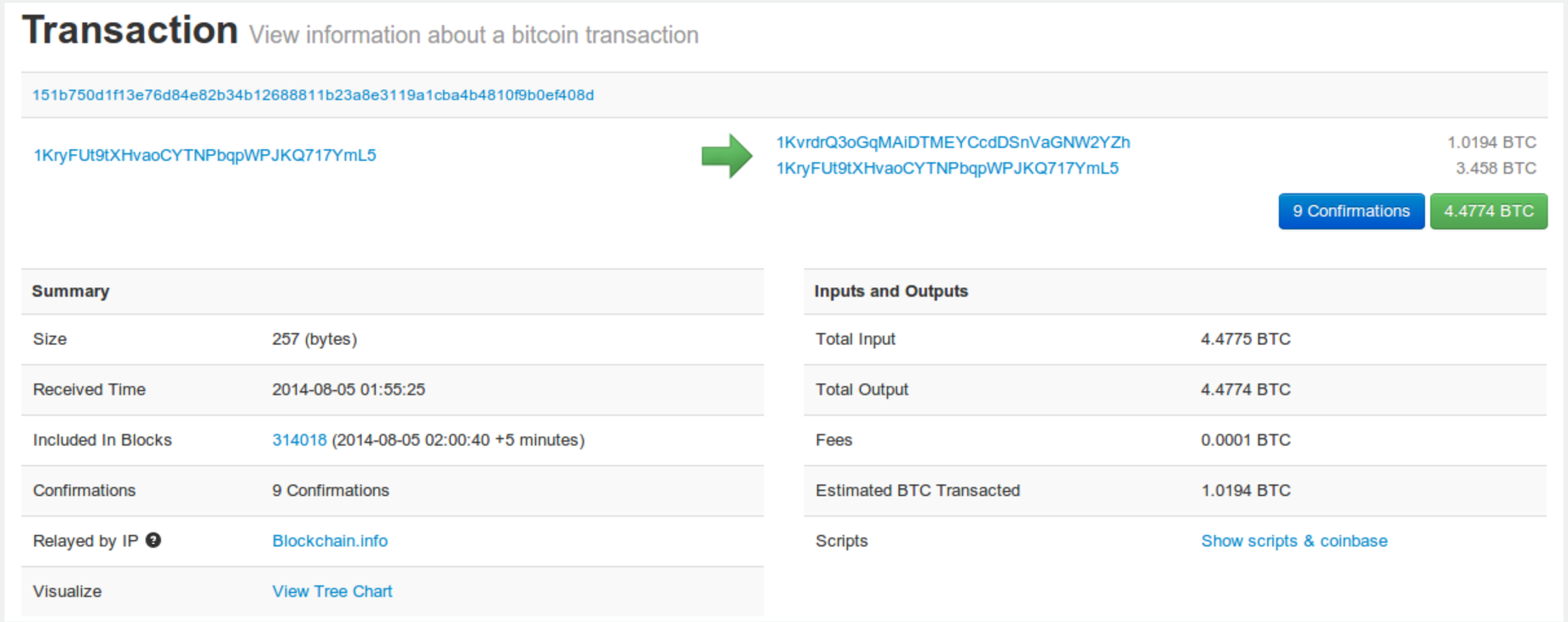
transaction data


```
{
 "hash":"00000000000000001aad2...",
 "ver":2,
 "prev_block":"00000000000000003043...",
 "time":1391279636,
 "bits":419558700,
 "nonce":459459841,
 "mrkl_root":"89776...",
 "n_tx":354,
 "size":181520,
 "tx":[
  ...
 ],
 "mrkl_tree":[
  "6bd5eb25...",
  ...
  "89776cdb..."
 ]
}
```

The real deal: a Bitcoin block header



See for yourself!



Transaction View information about a bitcoin transaction

151b750d1f13e76d84e82b34b12688811b23a8e3119a1cba4b4810f9b0ef408d

1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5	1KvrdrQ3oGqMAiDTMEYCcdDSnVaGNW2YZh	1.0194 BTC
	1KryFUt9tXHvaoCYTNPbqpWPJKQ717YmL5	3.458 BTC

9 Confirmations | 4.4774 BTC

Summary	
Size	257 (bytes)
Received Time	2014-08-05 01:55:25
Included In Blocks	314018 (2014-08-05 02:00:40 +5 minutes)
Confirmations	9 Confirmations
Relayed by IP	Blockchain.info
Visualize	View Tree Chart

Inputs and Outputs	
Total Input	4.4775 BTC
Total Output	4.4774 BTC
Fees	0.0001 BTC
Estimated BTC Transacted	1.0194 BTC
Scripts	Show scripts & coinbase

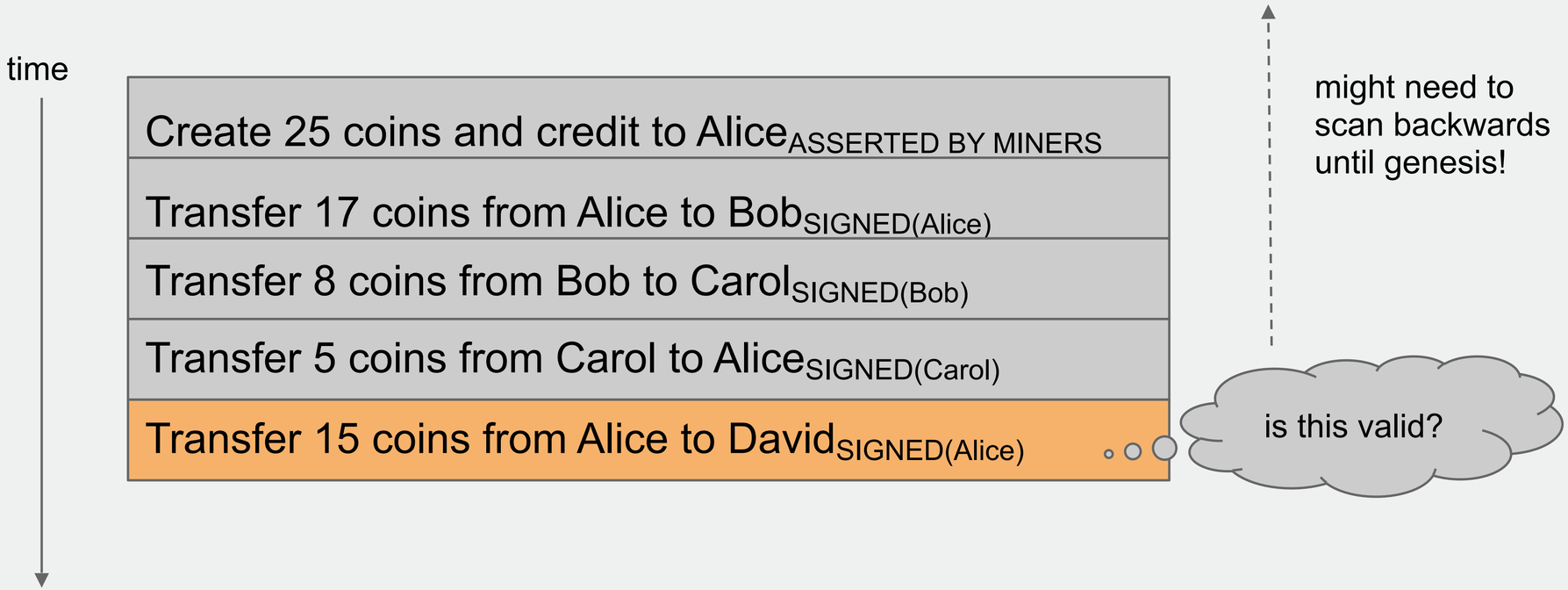
blockchain.info (and many other sites)

Addresses

- How to make a new identity:

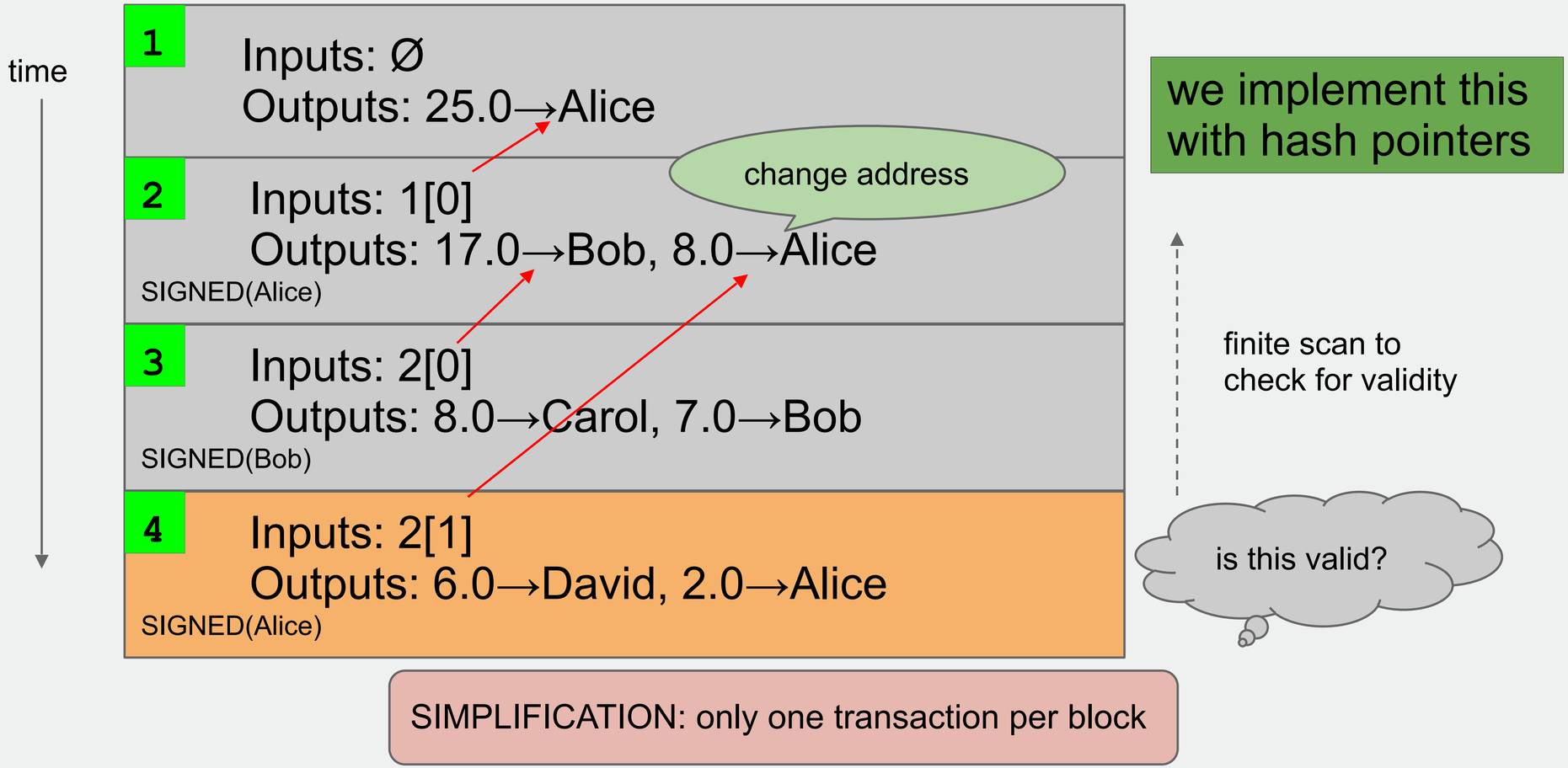
- create a new, random key-pair (sk, pk)
  - pk is the public “name” you can use
    [usually better to use Hash(pk)]
  - sk lets you sign messages on behalf of pk
- you control the identity, because only you know sk
- if pk “looks random”, nobody needs to know who you are

An account-based ledger (not Bitcoin)

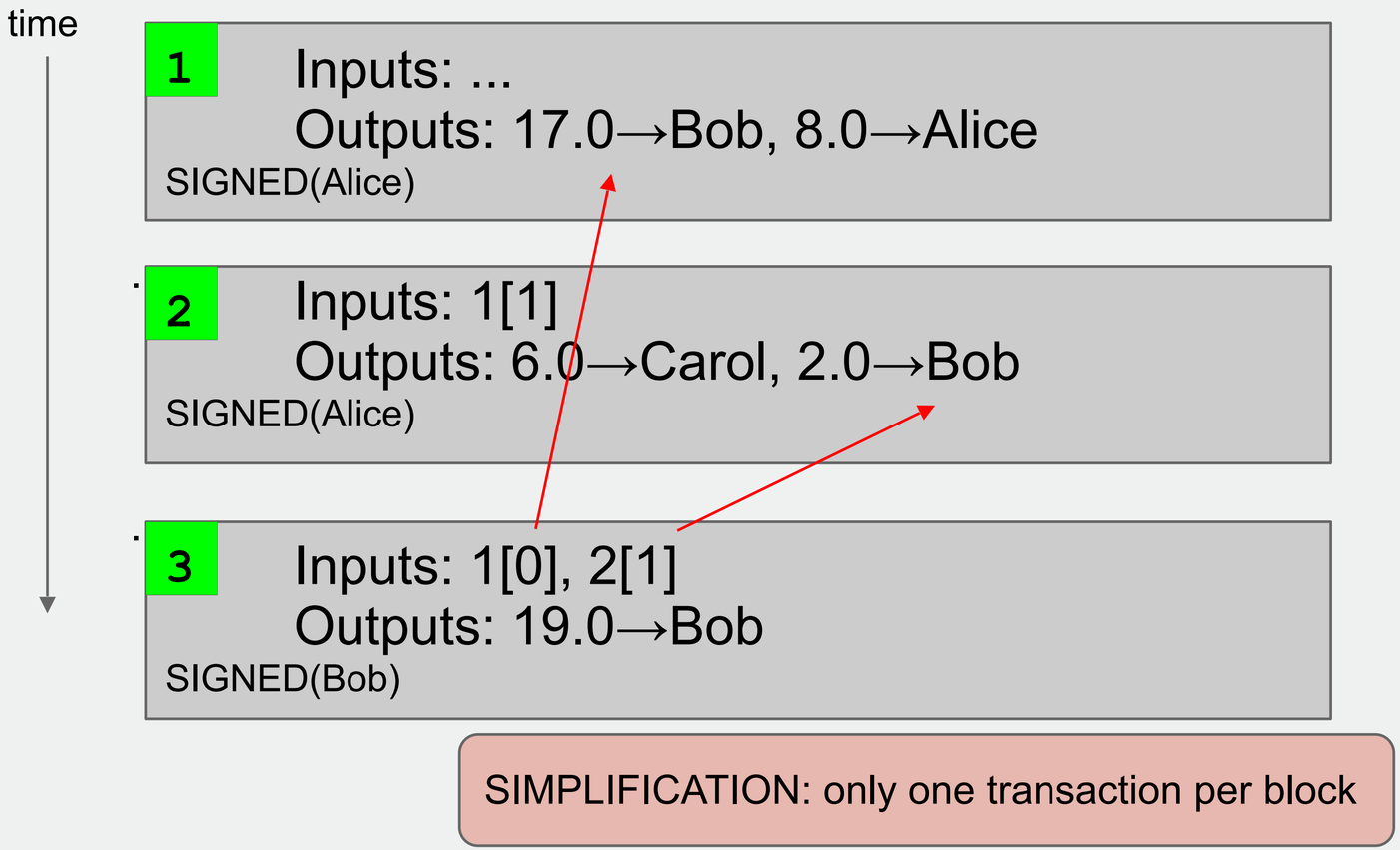


SIMPLIFICATION: only one transaction per block

A UTXO-based ledger (Bitcoin)

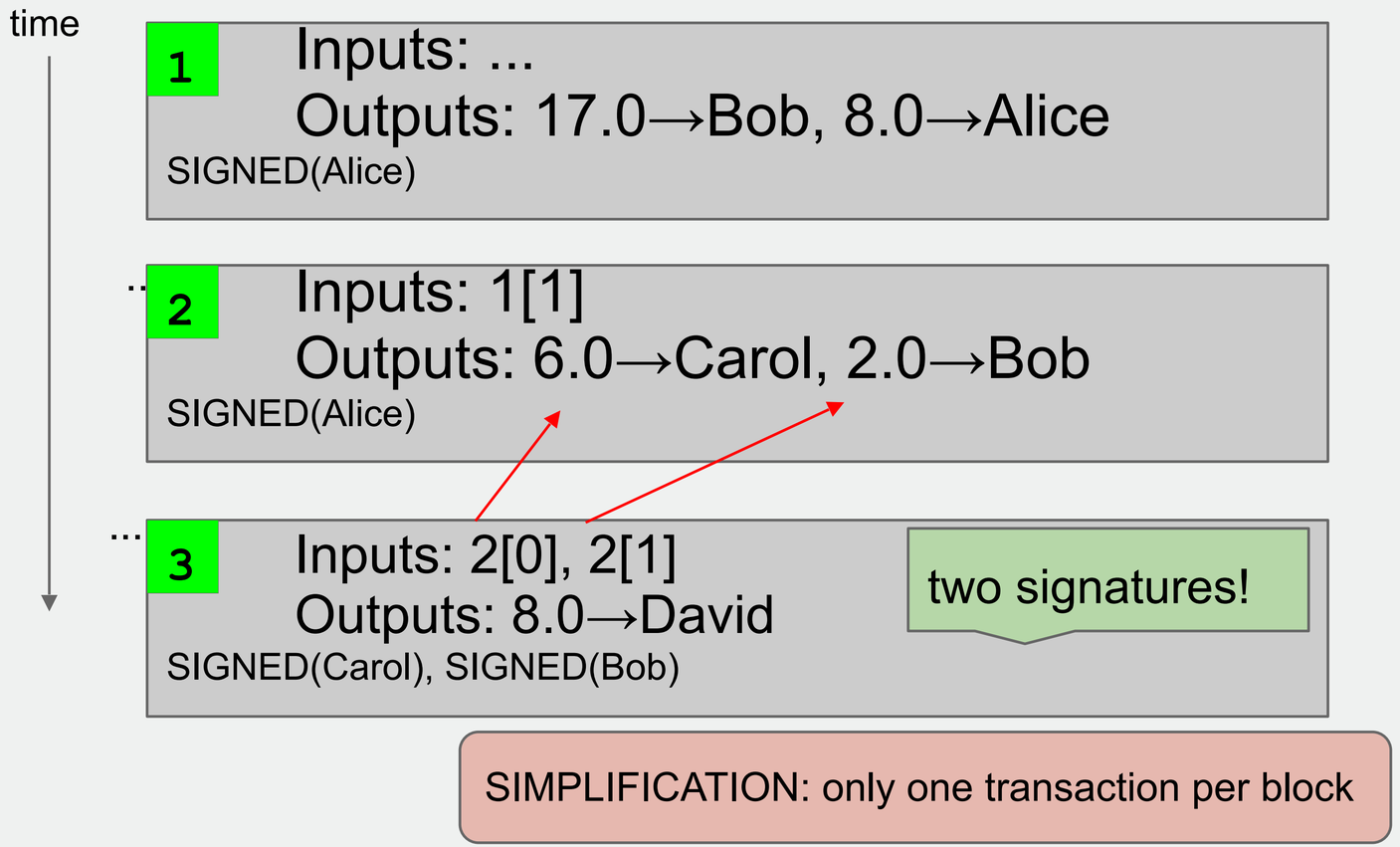


Merging value



SIMPLIFICATION: only one transaction per block

Joint payments



The real deal: a Bitcoin transaction

The real deal: transaction metadata

The real deal: transaction inputs

Where do we take the money from?

The real deal: transaction outputs

output value

recipient address??

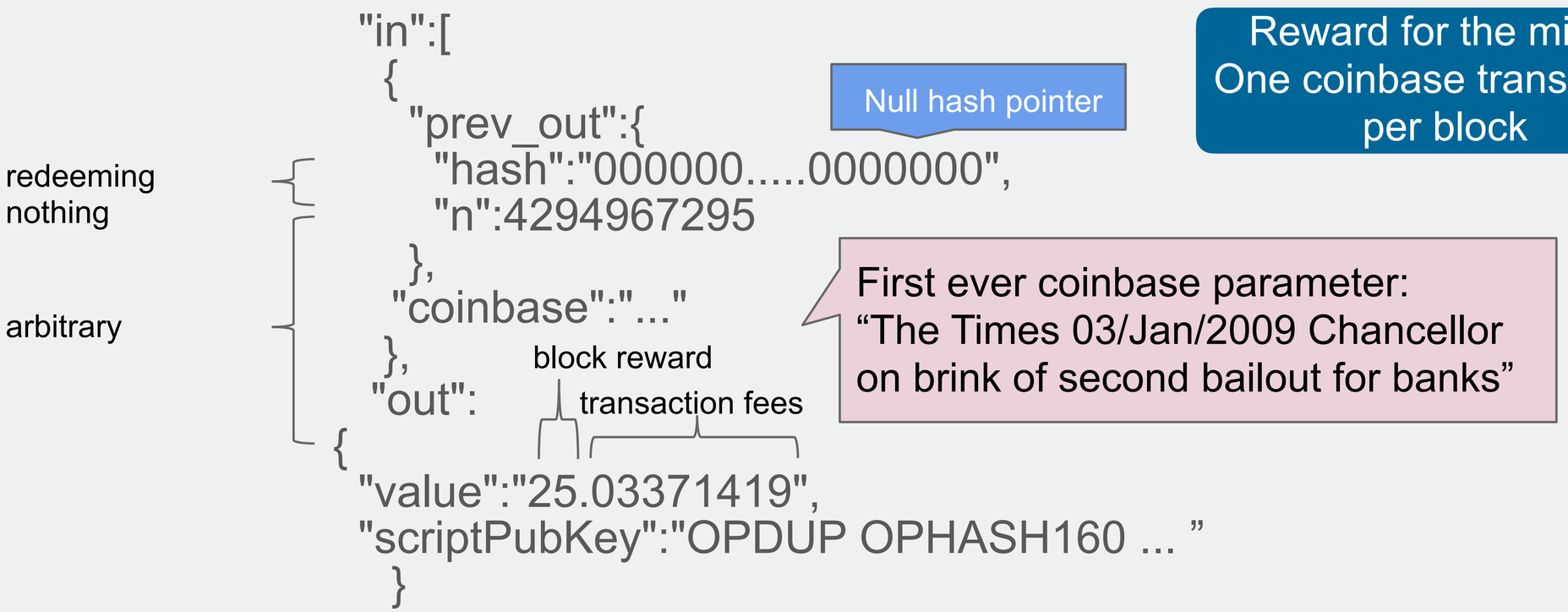
(more outputs)

```
"out":[
    {
      "value":"10.12287097",
      "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
    },
      ...
  ]
```

Whom do we give the money to?

more on this soon...

The real deal: coinbase transaction



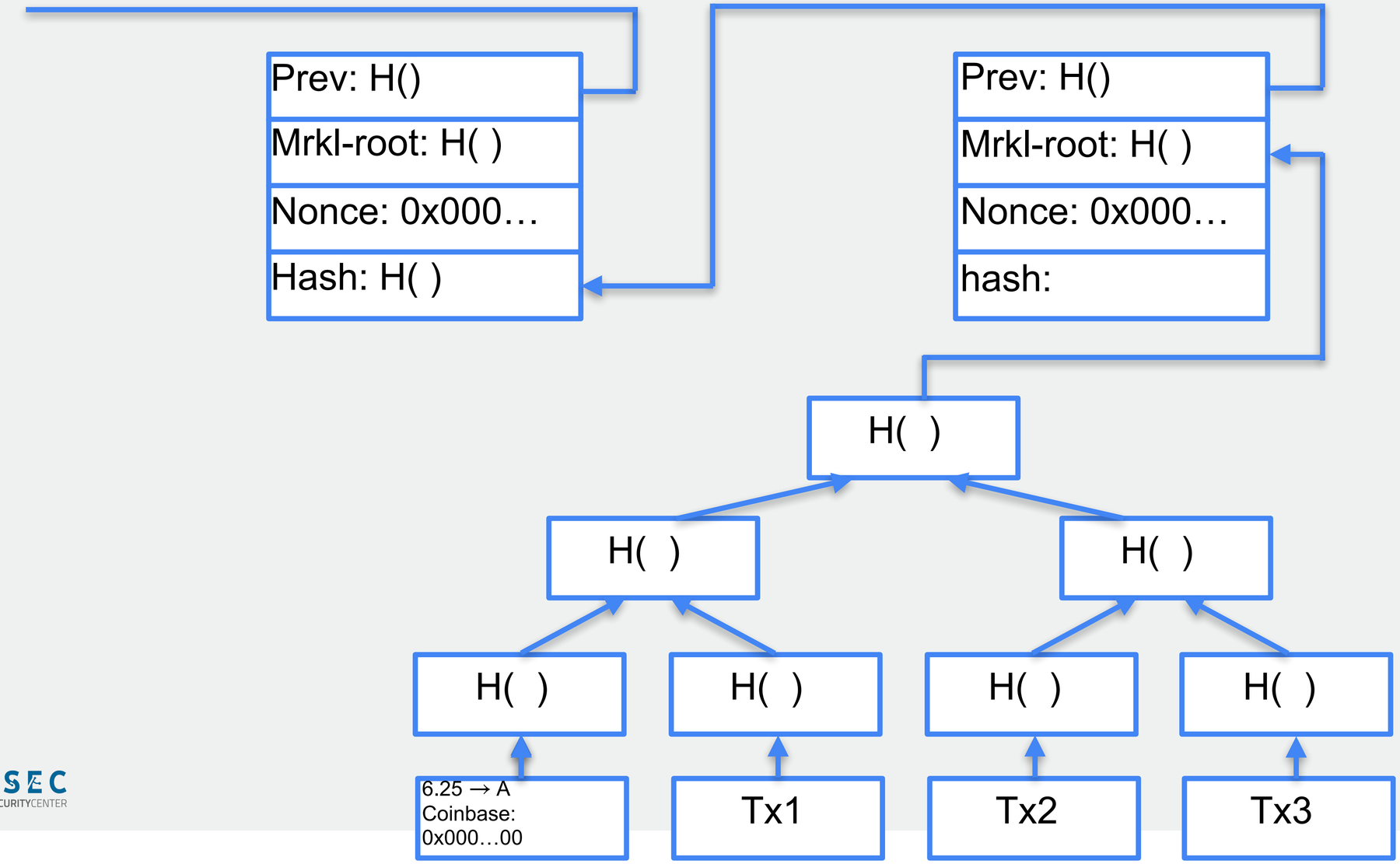
The task of Bitcoin miners

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

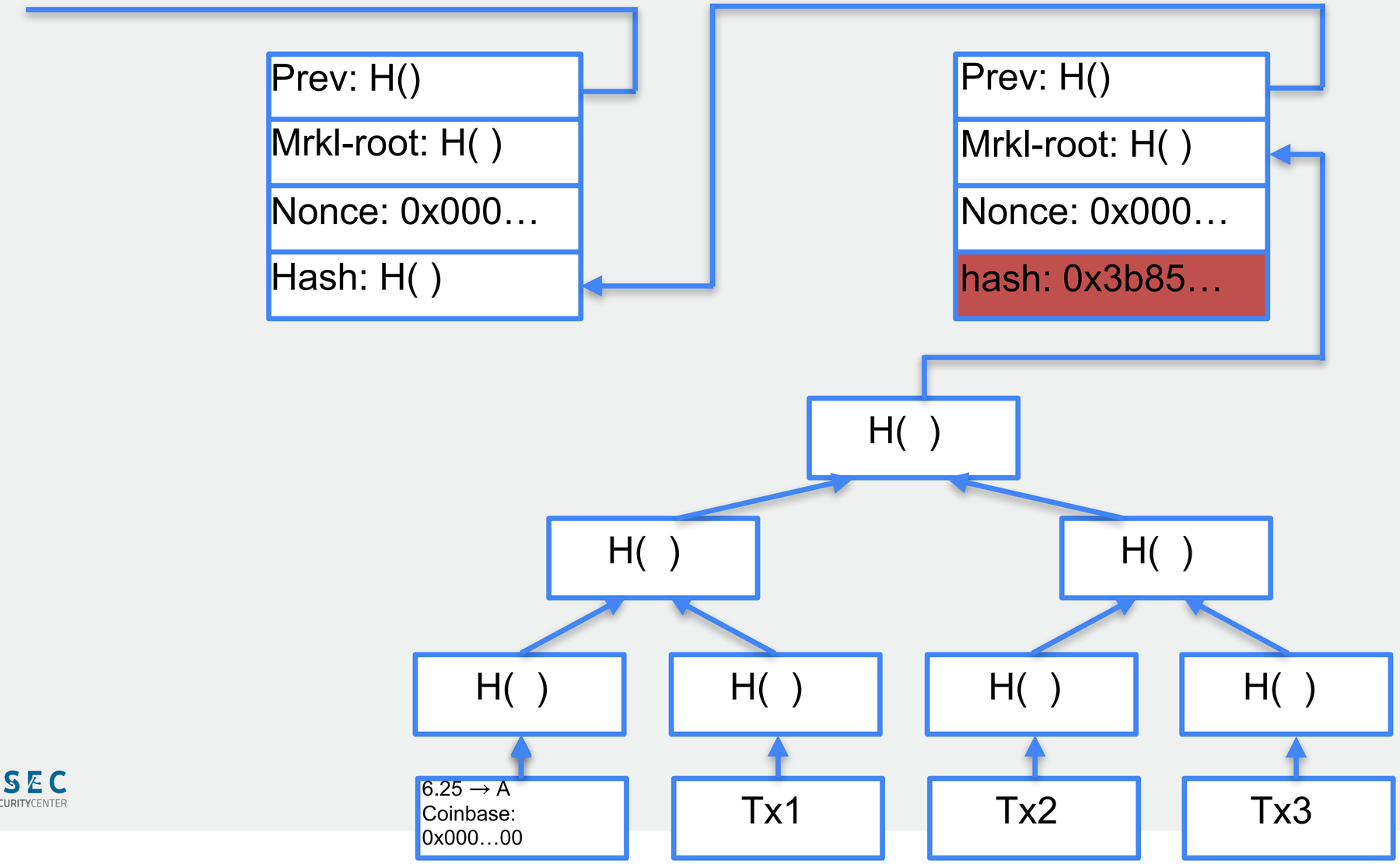
Mining Bitcoin in 6 easy steps

- Join the network, listen for transactions
  - Validate all proposed transactions
- Listen for new blocks, maintain block chain
  - When a new block is proposed, validate it
- Assemble a new valid block
- Find the nonce to make your block valid
- Hope everybody accepts your new block
- Profit!

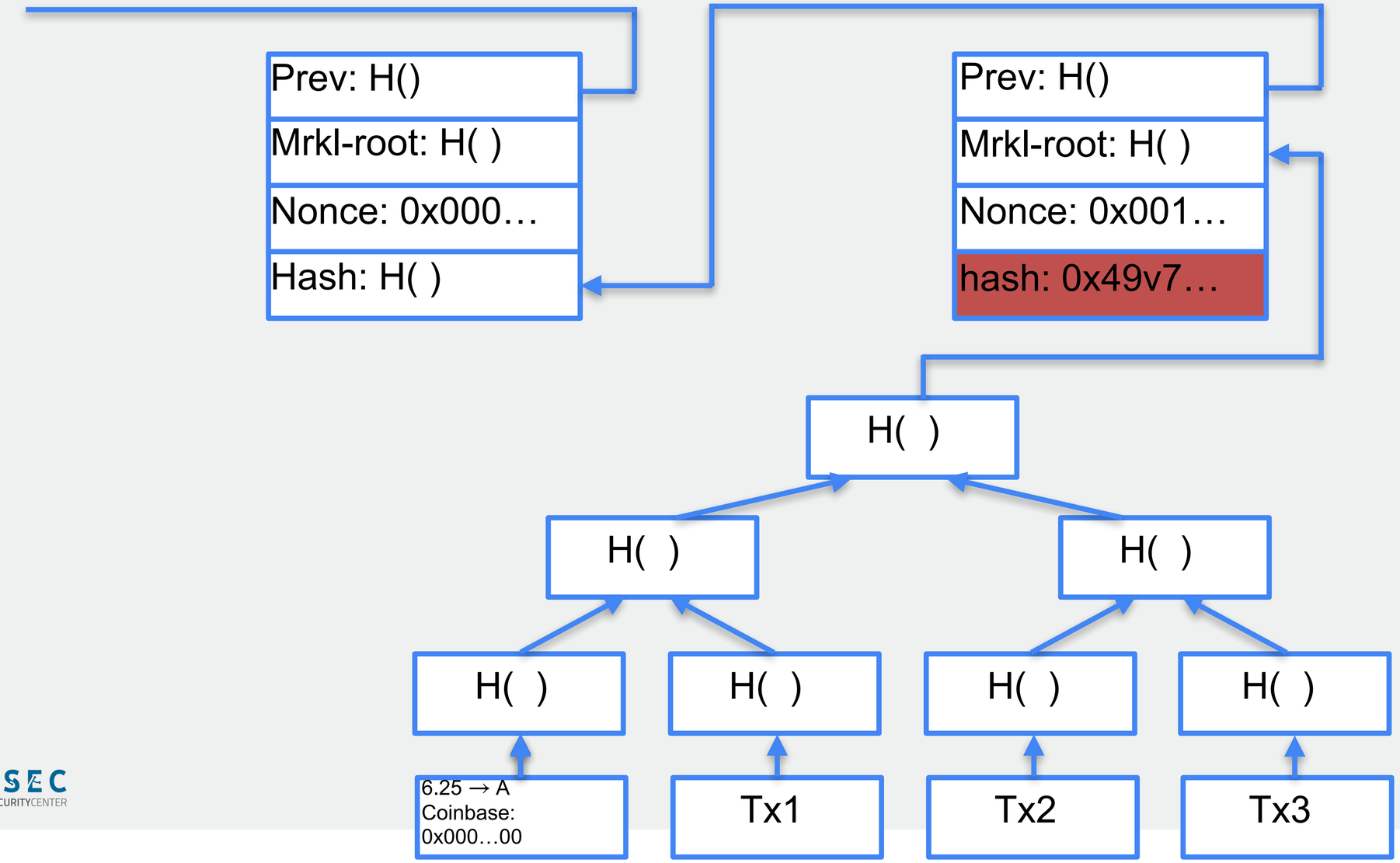
Finding a valid block



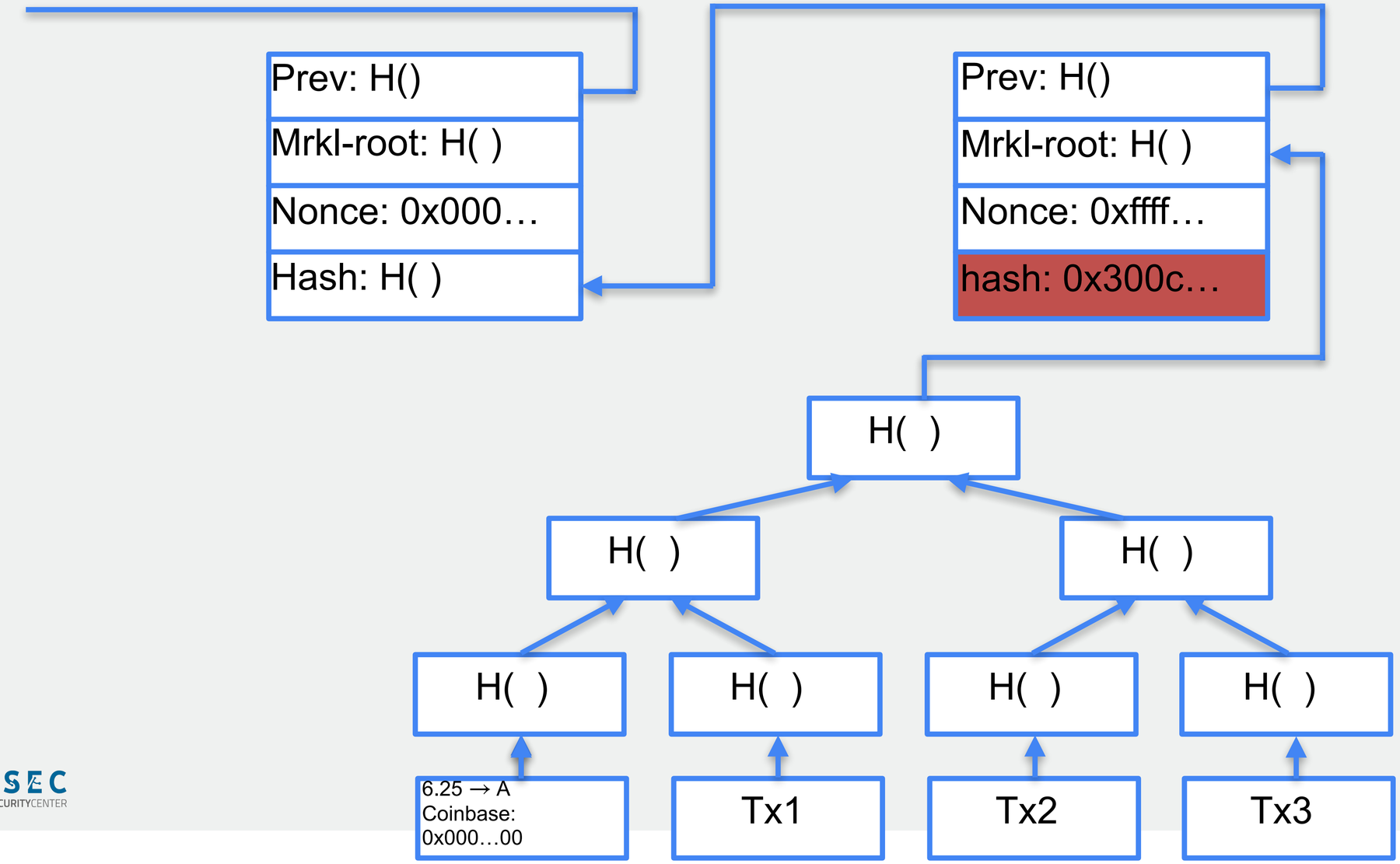
Finding a valid block



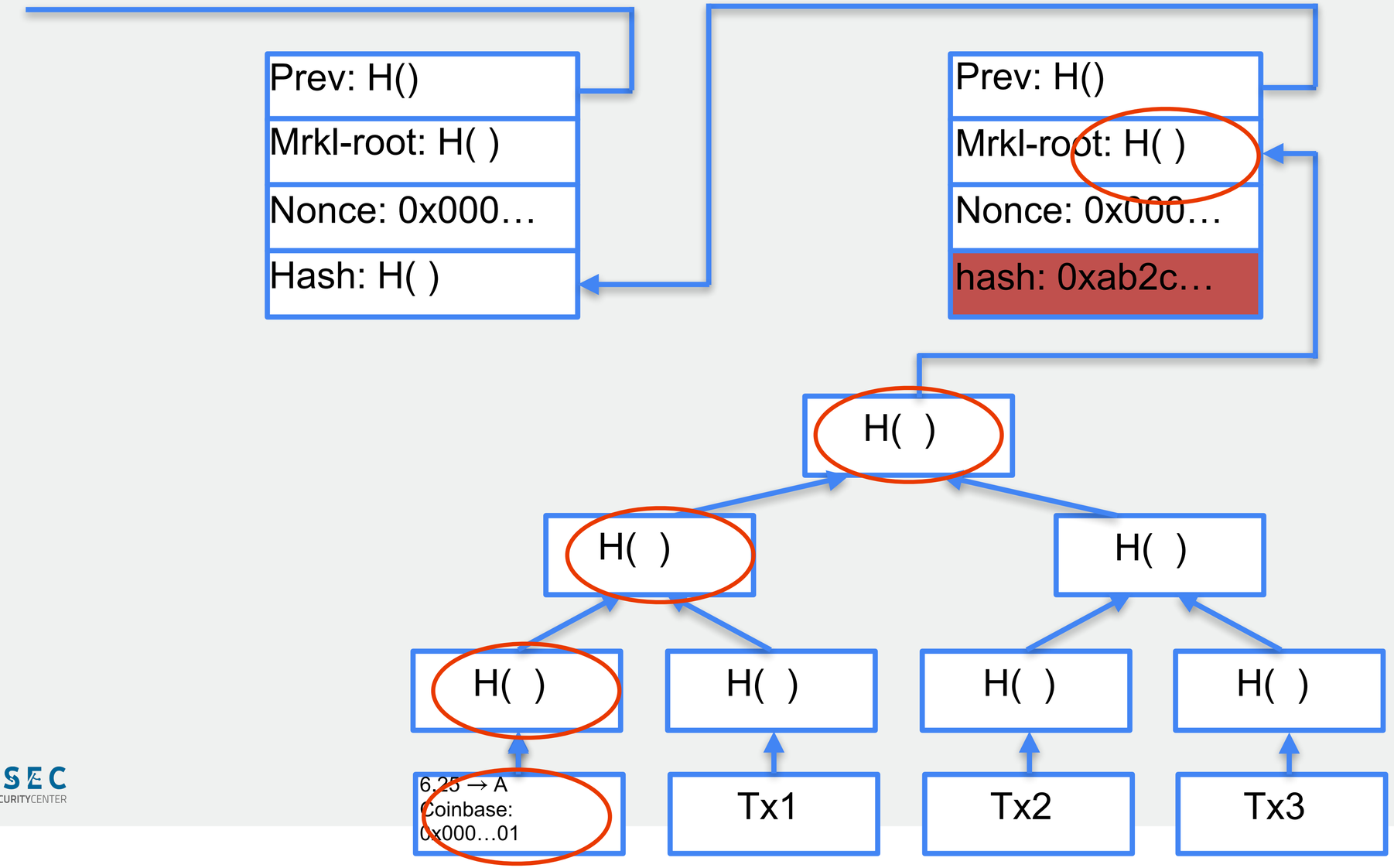
Finding a valid block



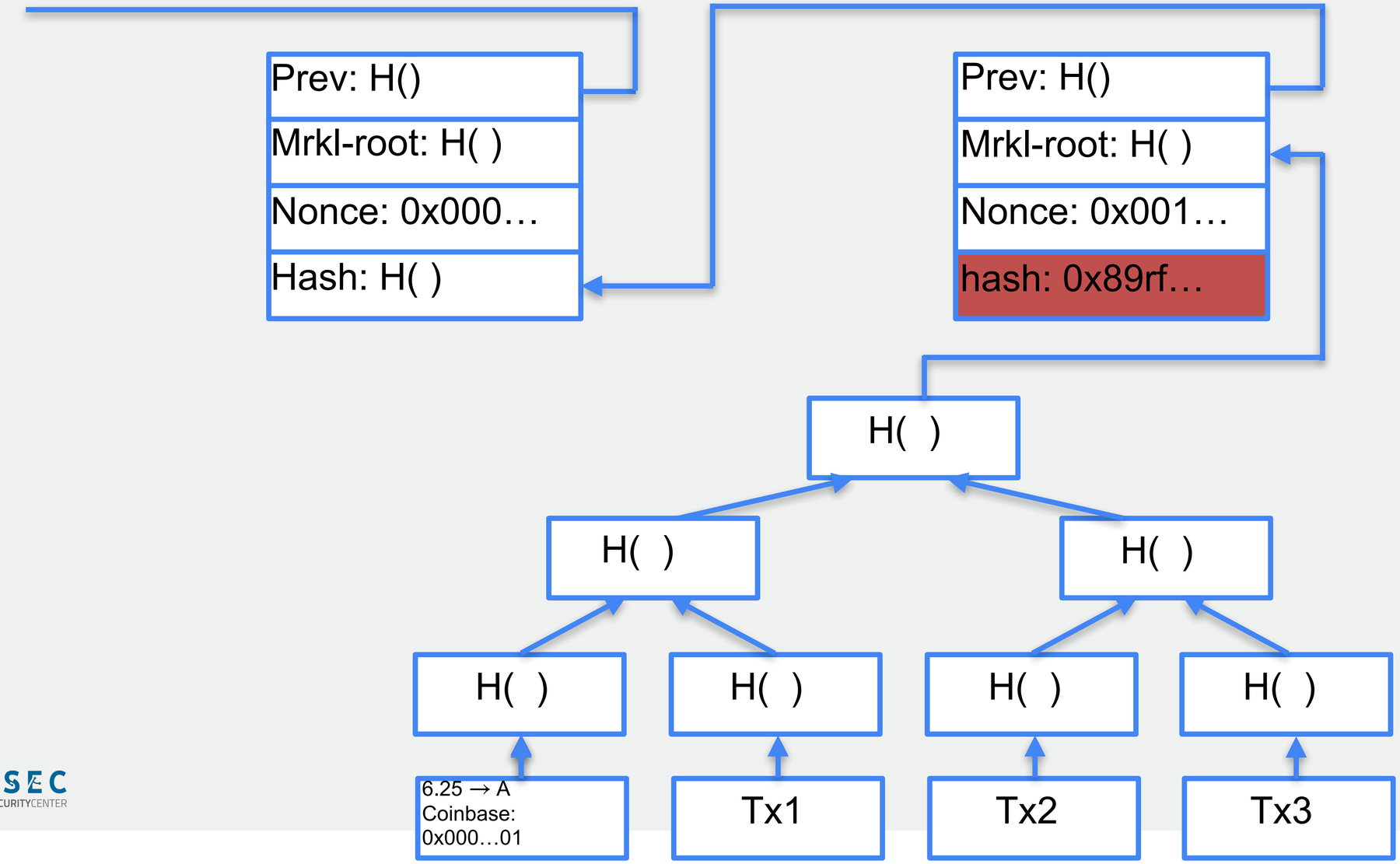
Finding a valid block



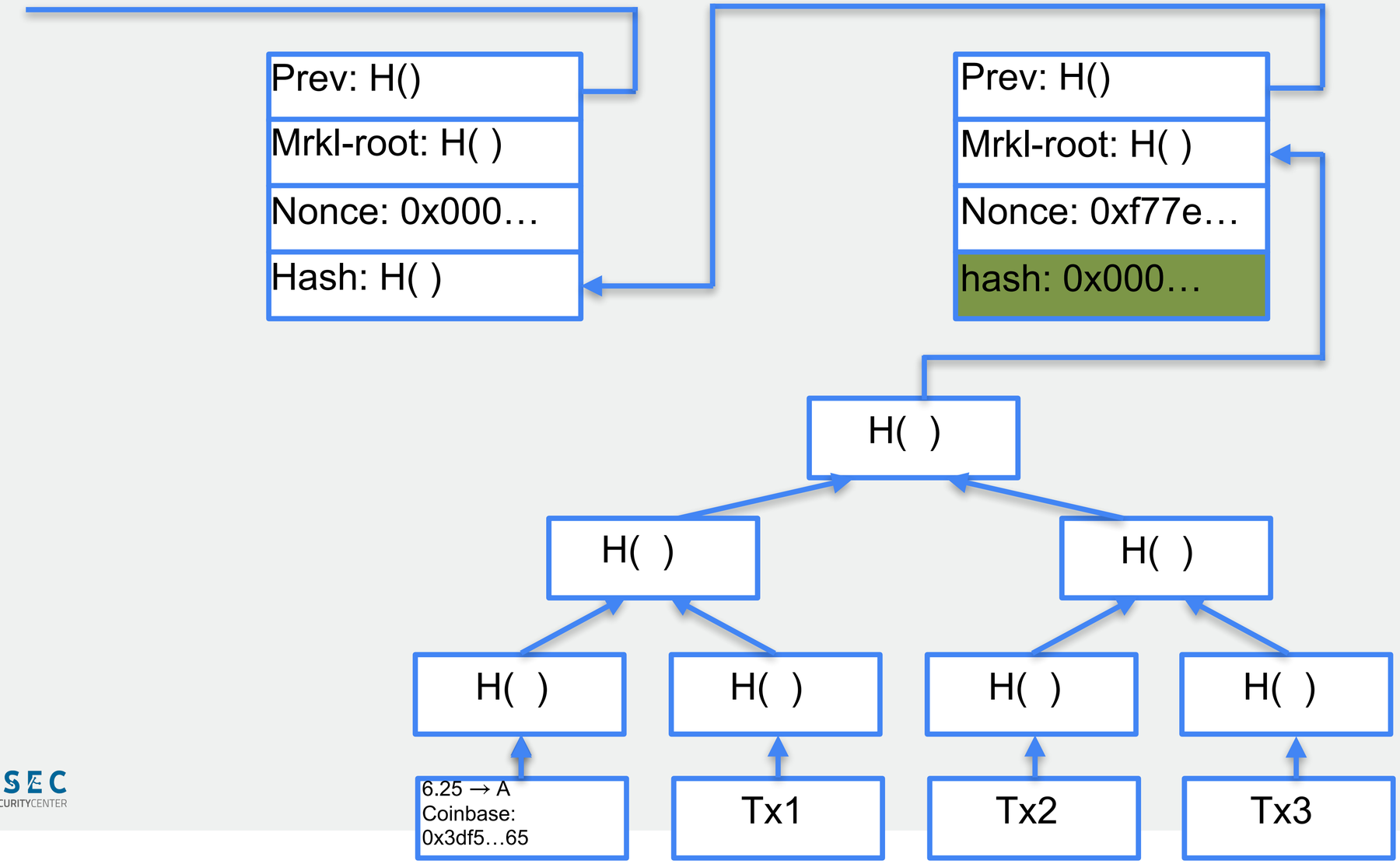
Finding a valid block



Finding a valid block



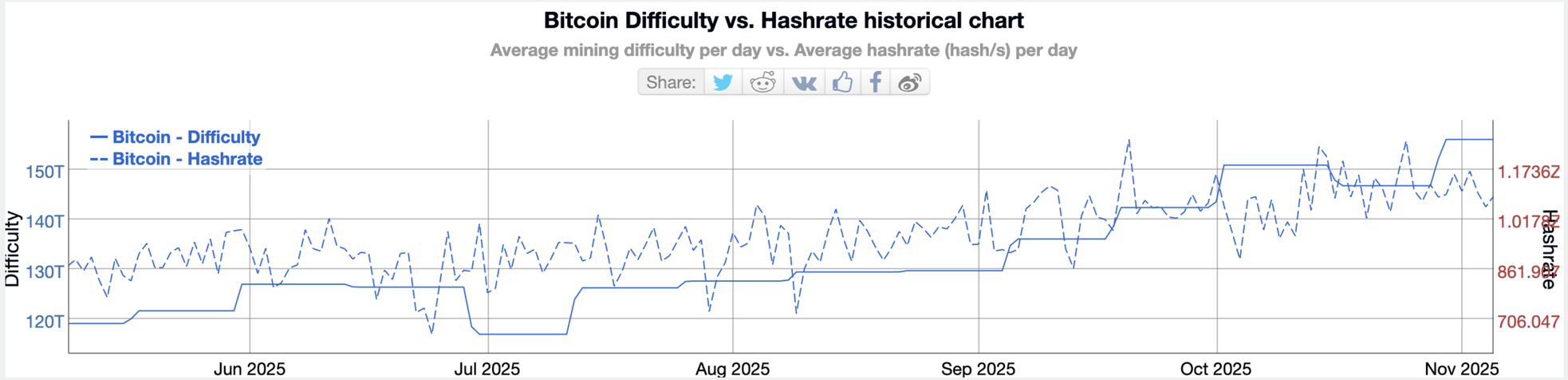
Finding a valid block



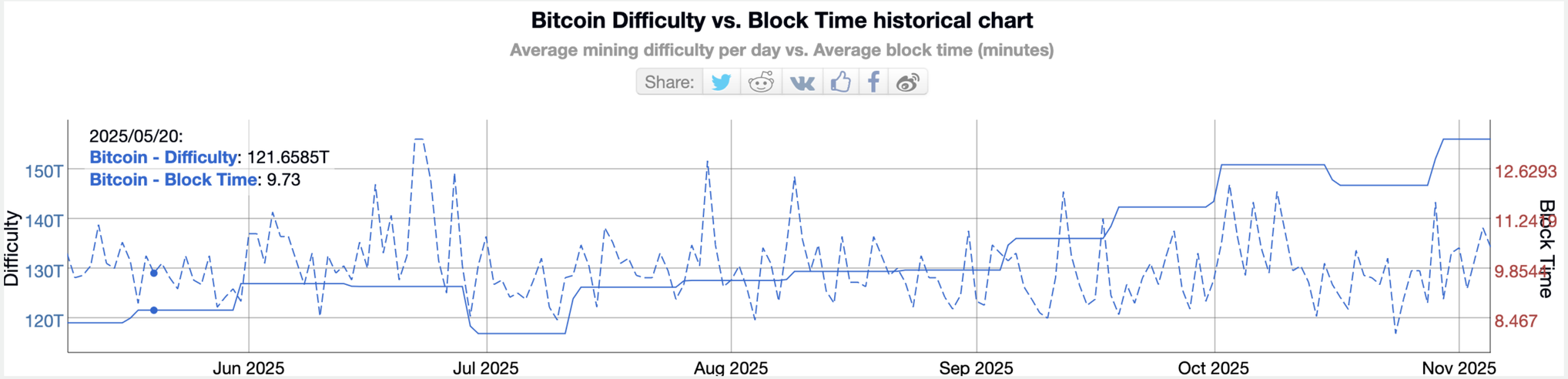
Setting the mining difficulty

Every two weeks compute:

$$\text{new-difficulty} = \text{previous-difficulty} \cdot \frac{2 \text{ weeks}}{\text{time to mine last 2016 blocks}}$$



Block time

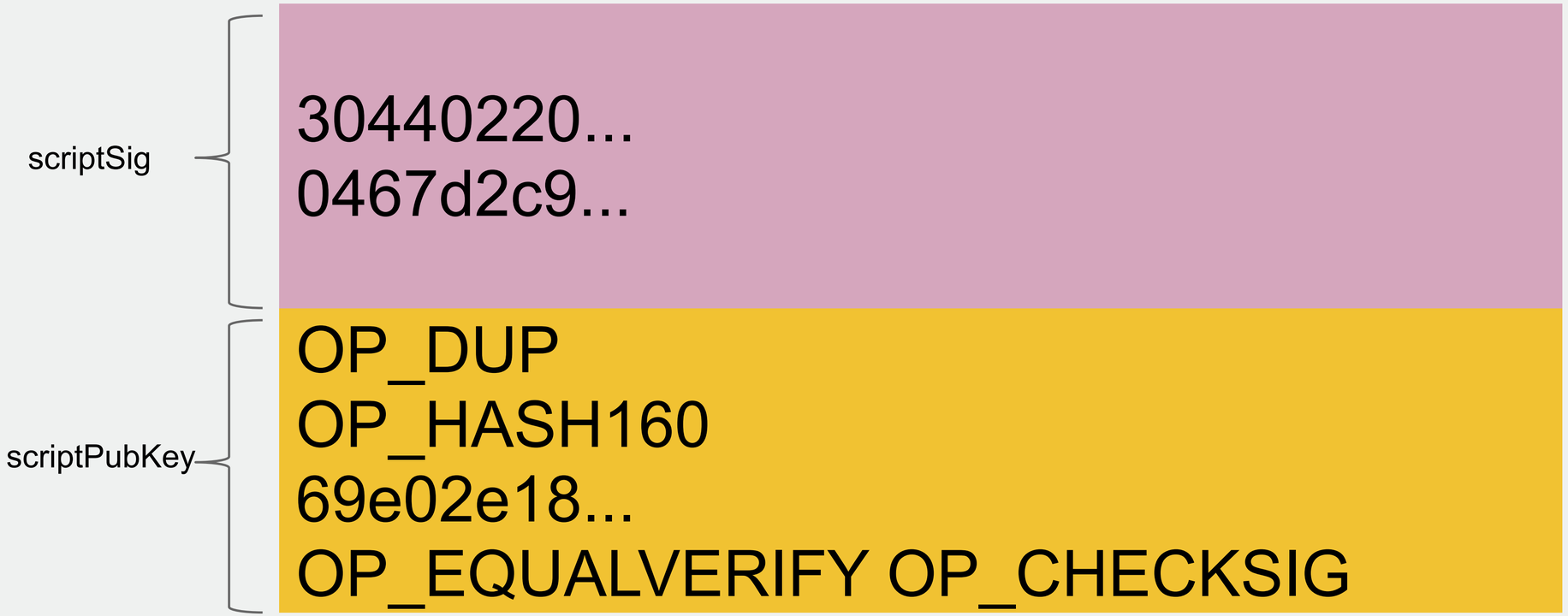

Bitcoin Difficulty vs. Block Time historical chart
Average mining difficulty per day vs. Average block time (minutes)
Share:
2025/05/20:
Bitcoin - Difficulty: 121.6585T
Bitcoin - Block Time: 9.73
Difficulty
150T
140T
130T
120T
Block Time
12.6293
11.2459
9.8544
8.467
Jun 2025
Jul 2025
Aug 2025
Sep 2025
Oct 2025
Nov 2025

Bitcoin scripts

Output “addresses” are really scripts

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

Input “addresses” are also scripts



TO VERIFY: Concatenated script must execute completely with no errors

Bitcoin scripting language (“Script”)

256 opcodes total (15 disabled, 75 reserved)

- **Crypto:** OP_CHECKSIG, OP_SHA256, ...
- **Arithmetic:** OP_ADD, OP_SUB, ...
- **Flow control:** OP_IF, OP_EL…
- **Stack:** OP_DUP, OP_PICK, OP_TOALTST…

…

Many disabled opcodes

- OP_MUL, OP_DIV, OP_XOR, OP_CAT, ...

No loops, no state

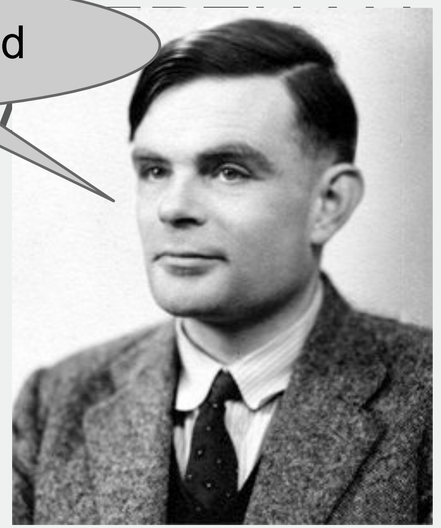
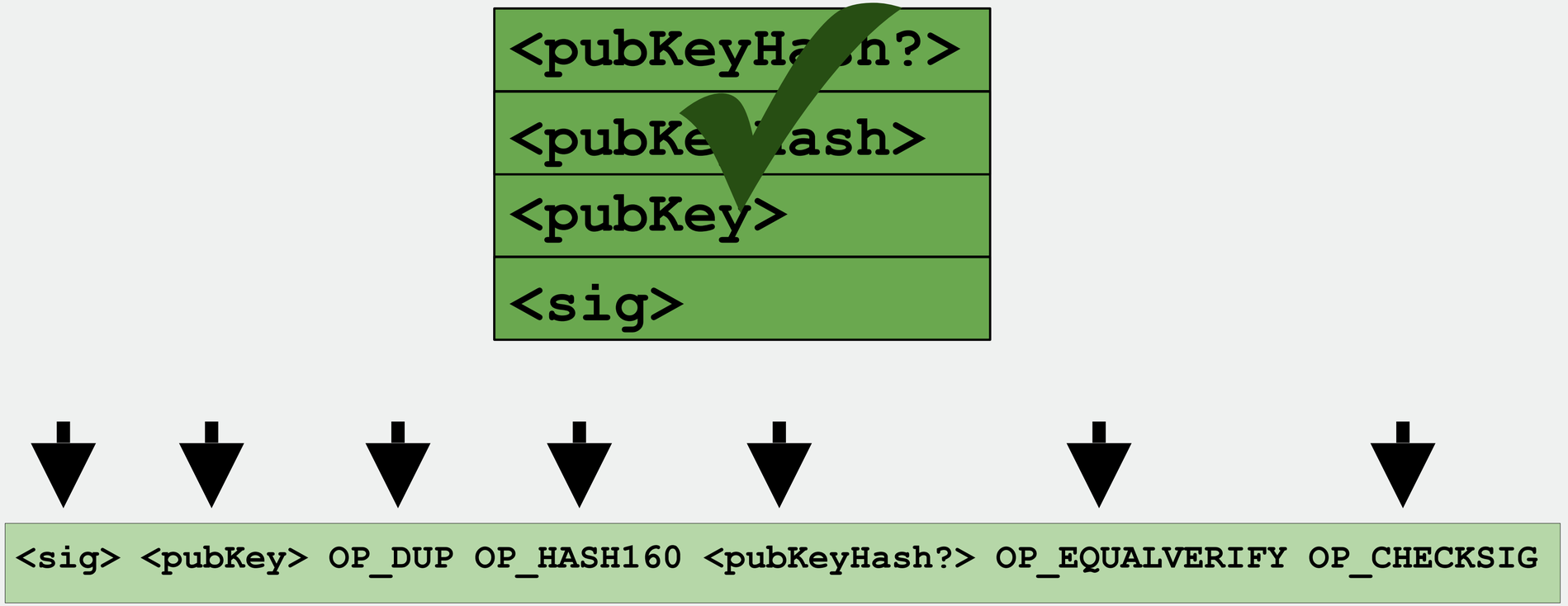


image via Jessie St. Amand

Bitcoin script execution example



Other useful opcodes

You can apply with Bitcoin scripts in various websites, e.g., https://paulkernfeld.com/bse/

Absolute TimeLocks

- tx valid until a certain block index

Relative TimeLocks

- tx valid after x blocks from the time the tx is posted

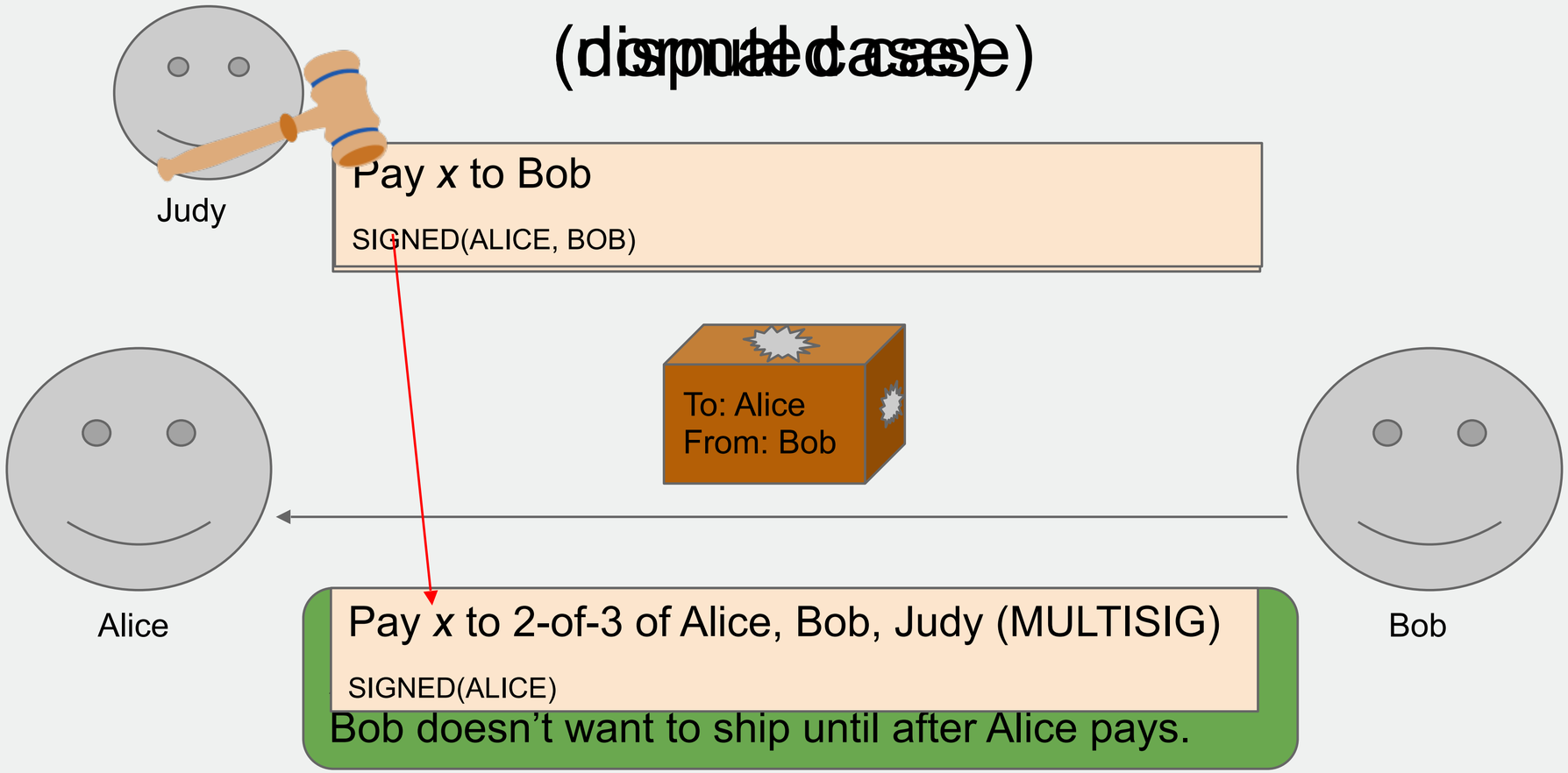
Hashed TimeLock Contract (HTLC)

- Tx can be spent only after showing the preimage of a certain hash before a certain relative timelock

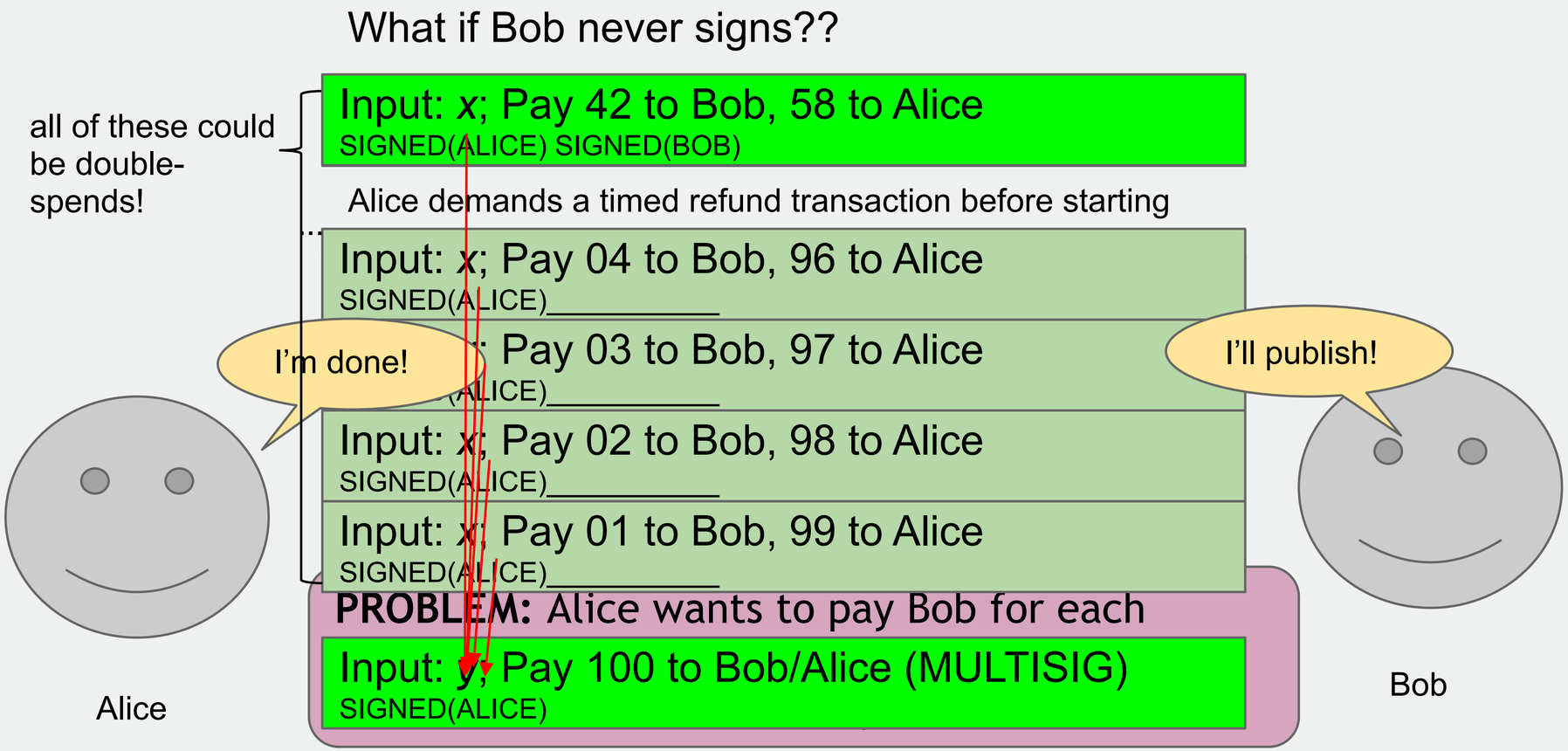
Applications of Bitcoin scripts

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Example 1: Escrow transactions



Example 2: Efficient micro-payments



lock_time


```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":315415,
      "size":404,
...
}
```


Block index or real-world timestamp before which this transaction can’t be published

More advanced scripts

- Multiplayer lotteries
- Hash pre-image challenges
- Coin-swapping protocols

“Scriptless Smart Contracts”

ACM CCS ‘18

BitML: A Calculus for Bitcoin Smart Contracts

Massimo Bartoletti
University of Cagliari
bart@unica.it

Roberto Zunino
University of Trento
roberto.zunino@unitn.it

https://github.com/bitml-lang/bitml-compiler

Almost Turing Complete

- With enabled covenants, Bitcoin Script is Quasi-Turing Complete (i.e., arbitrary computations without non-terminating loops)
- Covenants: mechanism allowing the sender to enforce constraints on the redeeming script, currently discussed as a soft fork

IEEE EUROS&P'18

Secure compilation of rich smart contracts on poor UTXO blockchains

Massimo Bartoletti
Università degli Studi di Cagliari
Cagliari, Italy
bart@unica.it

Riccardo Marchesin
Università degli Studi di Trento
Trento, Italy
riccardo.marchesin@unitn.it

Roberto Zunino
Università degli Studi di Trento
Trento, Italy
roberto.zunino@unitn.it

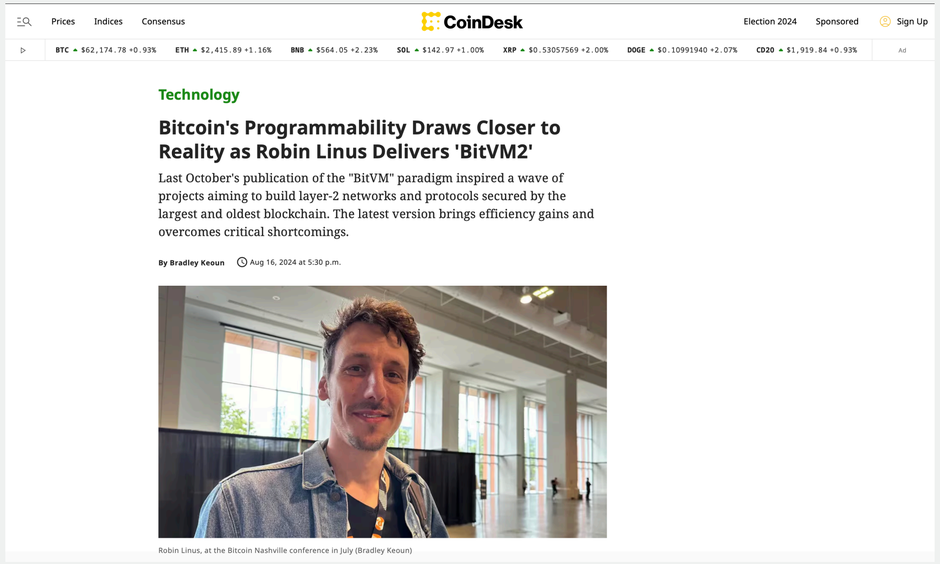
TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

Turing Complete!

Bitcoin Research Prize 2025

Bridging Bitcoin to Second Layers via BITVM2

Robin Linus*†, Lukas Aumayr‡§, Zeta Avarikioti¶§, Matteo Maffei¶, Andrea Pelosi**††¶,
Orfeas Thyfronitis Litos‖§, Christos Stefo¶, David Tse†, Alexei Zamyatin‡‡
*ZeroSync Association
†Stanford University
‡University of Edinburgh
§Common Prefix
¶TU Wien
‖Imperial College London
**University of Pisa
††University of Camerino
‡‡BOB



Prices Indices Consensus CoinDesk Election 2024 Sponsored Sign Up

BTC ▲ $62,174.78 +0.93% ETH ▲ $2,415.89 +1.16% BNB ▲ $564.05 +2.23% SOL ▲ $142.97 +1.00% XRP ▲ $0.53057569 +2.00% DOGE ▲ $0.10991940 +2.07% CD20 ▲ $1,919.84 +0.93% Ad

Technology

Bitcoin's Programmability Draws Closer to Reality as Robin Linus Delivers 'BitVM2'

Last October's publication of the "BitVM" paradigm inspired a wave of projects aiming to build layer-2 networks and protocols secured by the largest and oldest blockchain. The latest version brings efficiency gains and overcomes critical shortcomings.

By Bradley Keoun Aug 16, 2024 at 5:30 p.m.

Robin Linus, at the Bitcoin Nashville conference in July (Bradley Keoun)

- Check out all details at bitvm.org
- More on that later in the course…

Limitations & improvements

Hard-coded limits in Bitcoin

- 10 min. average creation time per block
- 1M bytes in a block
- 20,000 signature operations per block
- 100 M *satoshi* per bitcoin
- 21M total bitcoins maximum
- 50,25,12.5… bitcoin mining reward

These affect economic balance of power too much to change now

Throughput limits in Bitcoin

- 1 M bytes/block (10 min)
- > 250 bytes/transaction
- 7 transactions/sec ☹

Scalability expanded in Lecture 9

Compare to:

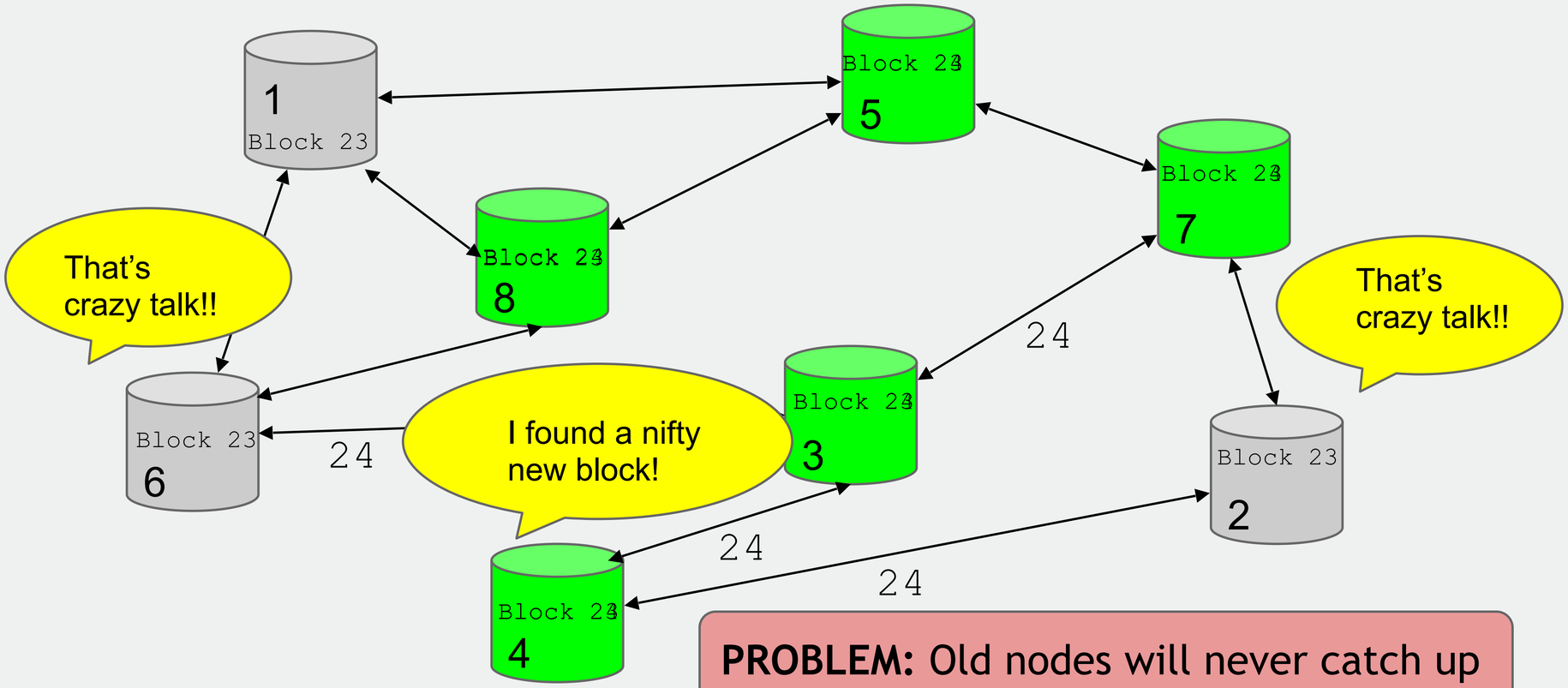
- VISA: 2,000-10,000 transactions/sec
- PayPal: 50-100 transaction/sec

Hard forks

- Hard forks *extend* the set of valid blocks
  - New opcodes
  - Changes to size limits
  - Changes to mining rate
  - Many small bug fixes

- E.g., BitCash extended the block size to 8Mb and removed SegWit
  - De facto doubling the number of coins in circulation, as one Bitcoin obtained before the fork can be spent as BitCash too :)
  - Why did Bitcoin refused to increase the block size? What is the problem? (If in doubt, ask ChatGPT :))

“Hard-forking” changes to Bitcoin



Soft forks

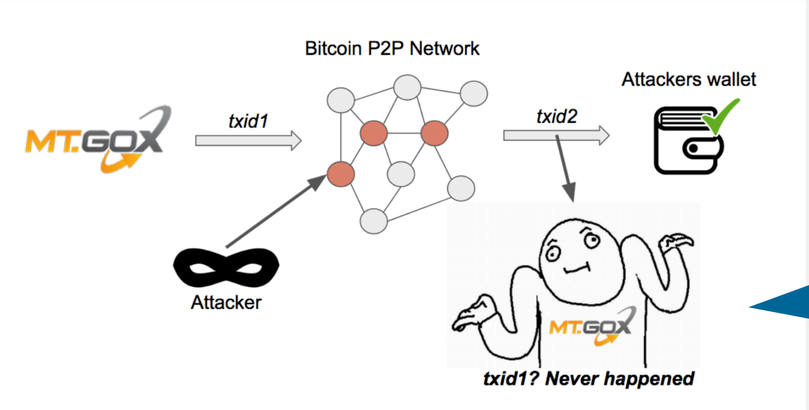
- Soft forks *limit* the set of valid transactions/blocks
- Need majority of nodes to enforce new rules
- Old nodes will approve

- E.g., Pay to Script Hash (2012), SegWit (2017), Taproot (2021)

RISK: Old nodes might mine now-invalid blocks

Soft fork example: SegWit

- Main motivation: malleability and scalability
- Malleability: in principle, a node could change the script (e.g., by adding a NOP) without invalidating the signature, but changing the transaction identifier (txid=hash of transaction data)
- Bad if people us taxids to identify transactions, like in MT.Gox Exchange 2013



The attacker gets the money but his account is not decreased, as txid1 never reaches the blockchain

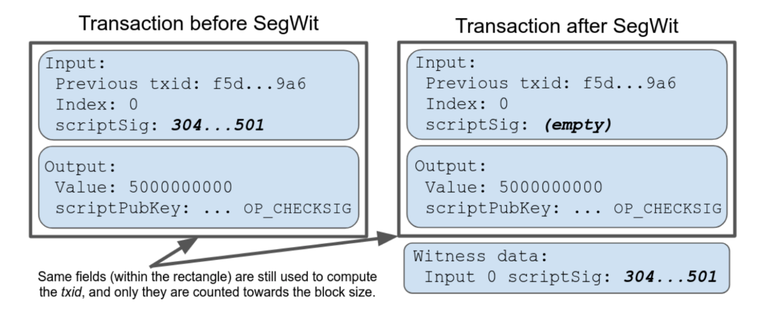
https://arxiv.org/abs/1403.6676

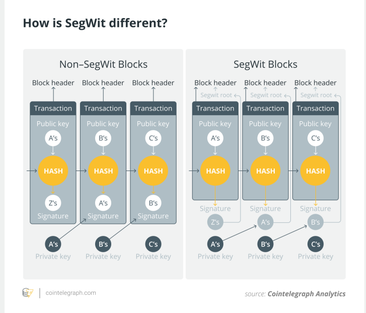
Soft fork example: SegWit

- Scalability:
  - the block has a max size of 1MB, so storing less information in a block means augmenting the throughput (i.e., more transactions in a block)
  - Also, if A pays to B (txa) and B to C (txb), both transactions could be gathered into one block, but if txb has to refer to idtxa and idtxa can change, then we have to wait for txa to make it on the blockchain before we can continue with txb…crucial for scriptless scripts and Lightning Network (more in the following lectures)

Soft fork example: SegWit

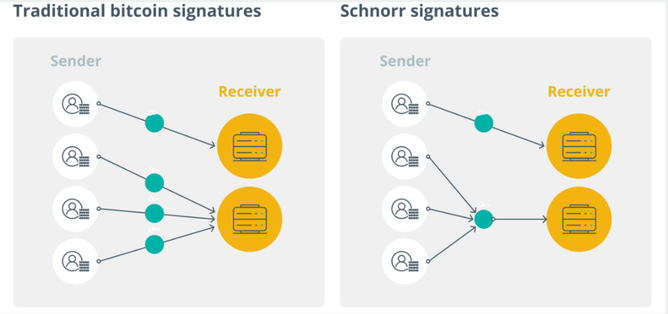
- Signature scripts pushed out of the transaction and included in a separate Merkle tree, which is then joined with the transaction into a unique Merkle tree (backwards compatibility)
- Blocks contained max 1MB txs, with SegWit max 4M weight units (WU)
  - Bytes in the witness count 1WU, the others 4WU (blocks contain more transactions)
- New opcodes become now soft forks, since witness data contain script versioning: old miners will just assume new scripts to be true :)



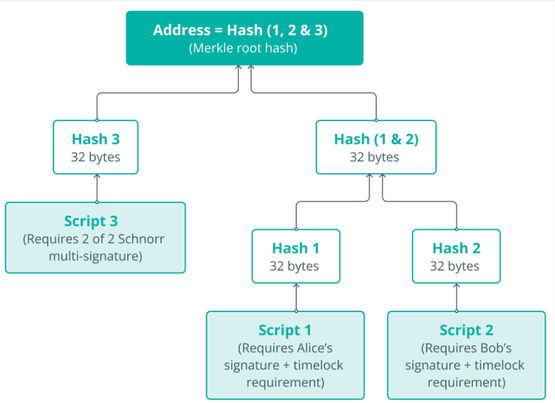


Taproot

- Schnorr signatures (replace ECDSA)
  - Efficient signature aggregation and validation
  - Single sigs indistinguishable from multi-sigs
  - Shorter sigs



- Pay-to-Taproot
  - Merkelized Abstract Syntax Tree (MAST): only reveal the executed part of the script (single script branch in a script Merkle tree)
    - Better privacy
    - Better scalability



TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER