TÉCNICO LISBOA

Permissionless blockchains. Bitcoin

Highly dependable systems

Lecture 7

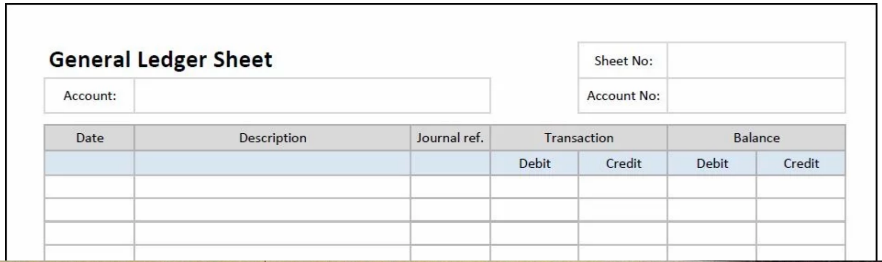
Lecturers: Miguel Matos and Paolo Romano

Last Lecture: Consensus

- Fundamental building block in distributed systems

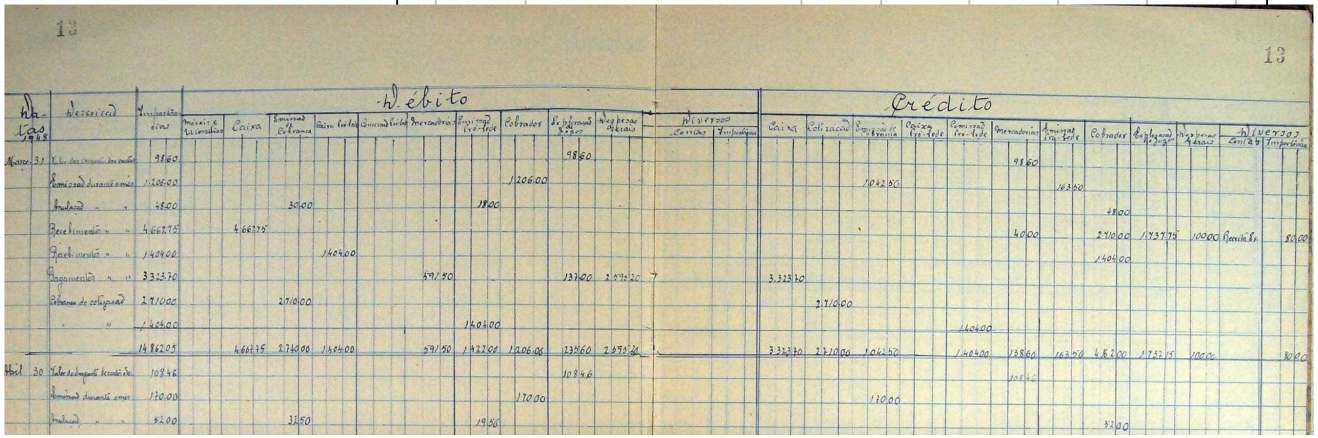
- Through the State Machine Replication approach, it can be used to replicate any (deterministic) system, including:
  - databases, file-systems, video-games,…
  - as well as **<u>distributed ledgers</u>**

Distributed Ledgers

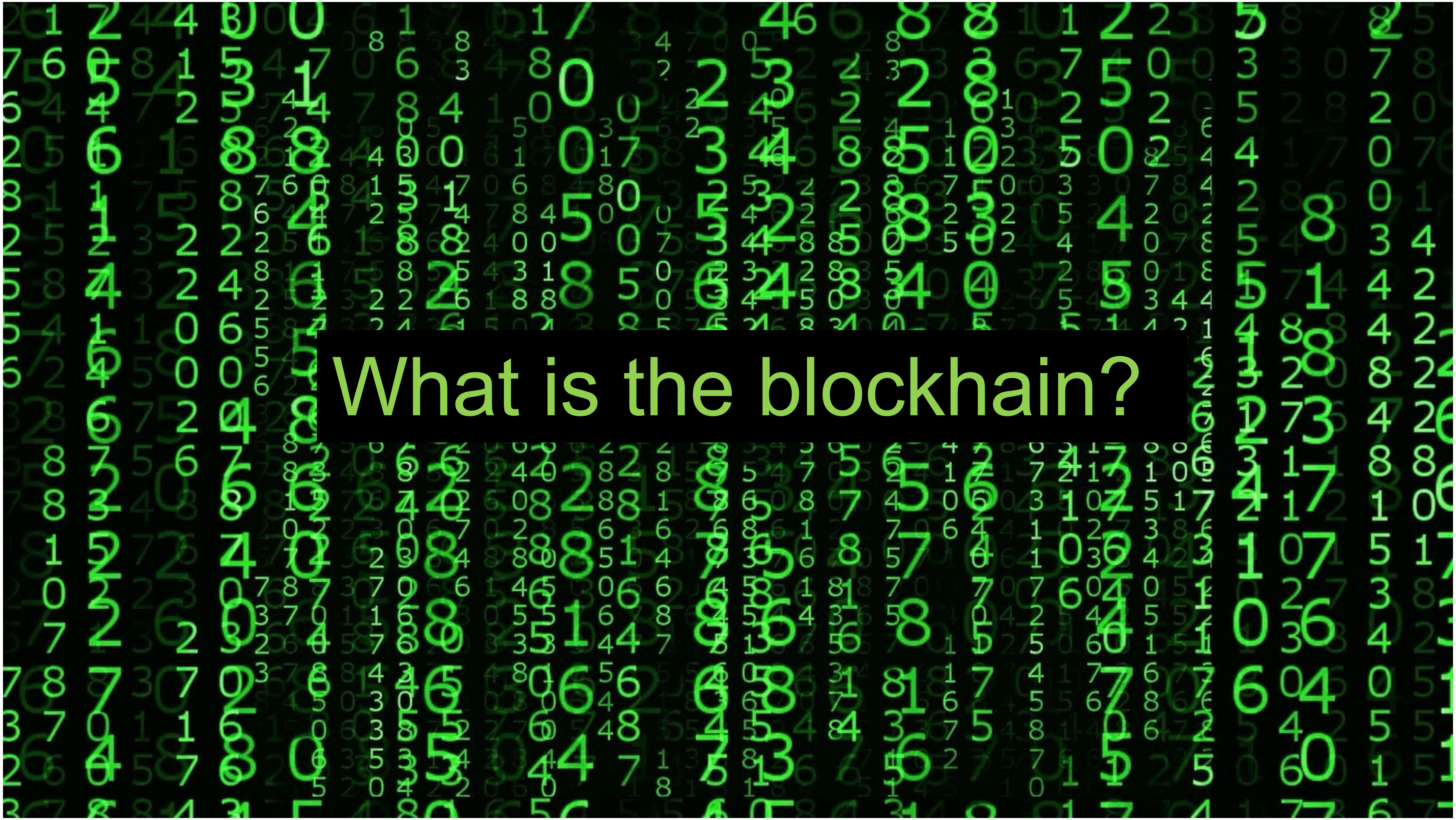

General Ledger Sheet

Account:		Sheet No:	
		Account No:	

Date	Description	Journal ref.	Transaction		Balance	
			Debit	Credit	Debit	Credit

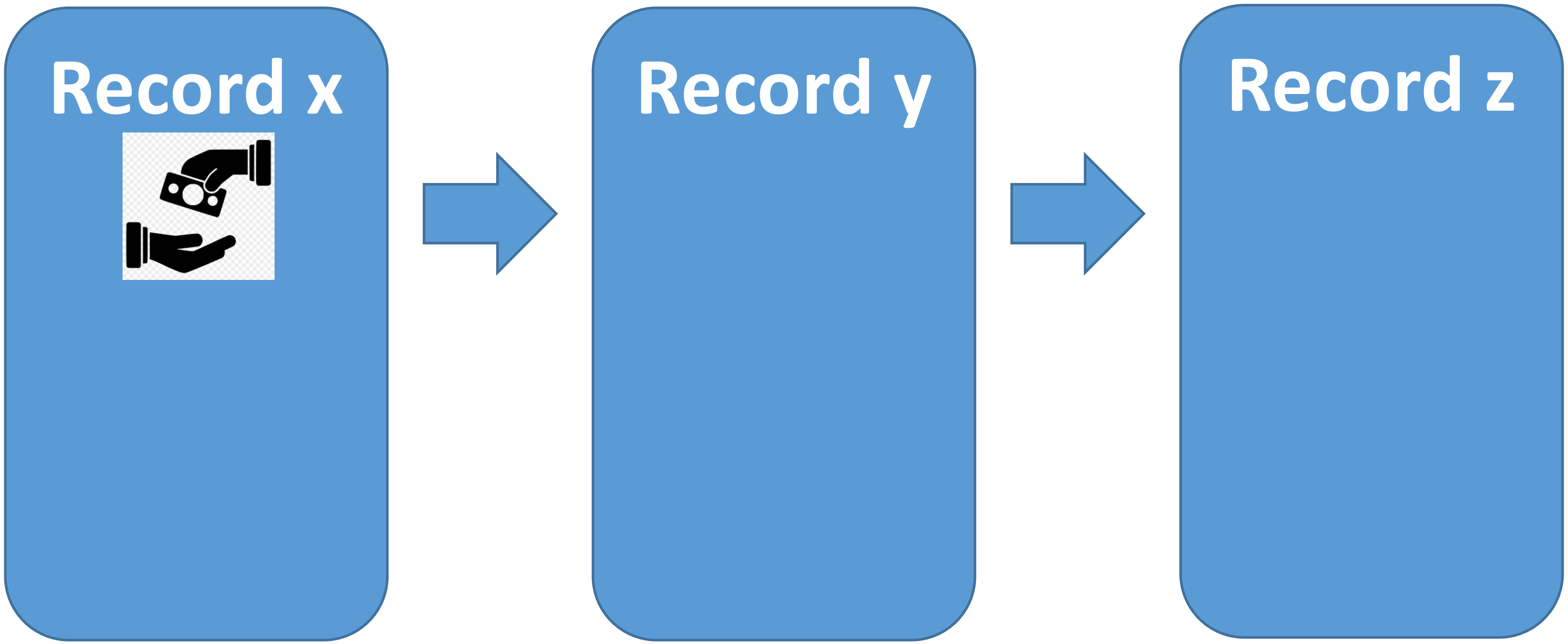


- Ledger
  - Record of transactions
  - Tamper proof
  - Ordered
- Distributed Ledger with a fixed set of participants (aka “permissioned”), without centralized authority or control
  - E.g., consortium of companies, special entity approves new members
  - Solution: replicate the ledger
  - Use consensus to decide which operation to append next to the ledger
- Challenge: in an open Internet environment, membership is open and dynamic (aka “permissionless”)

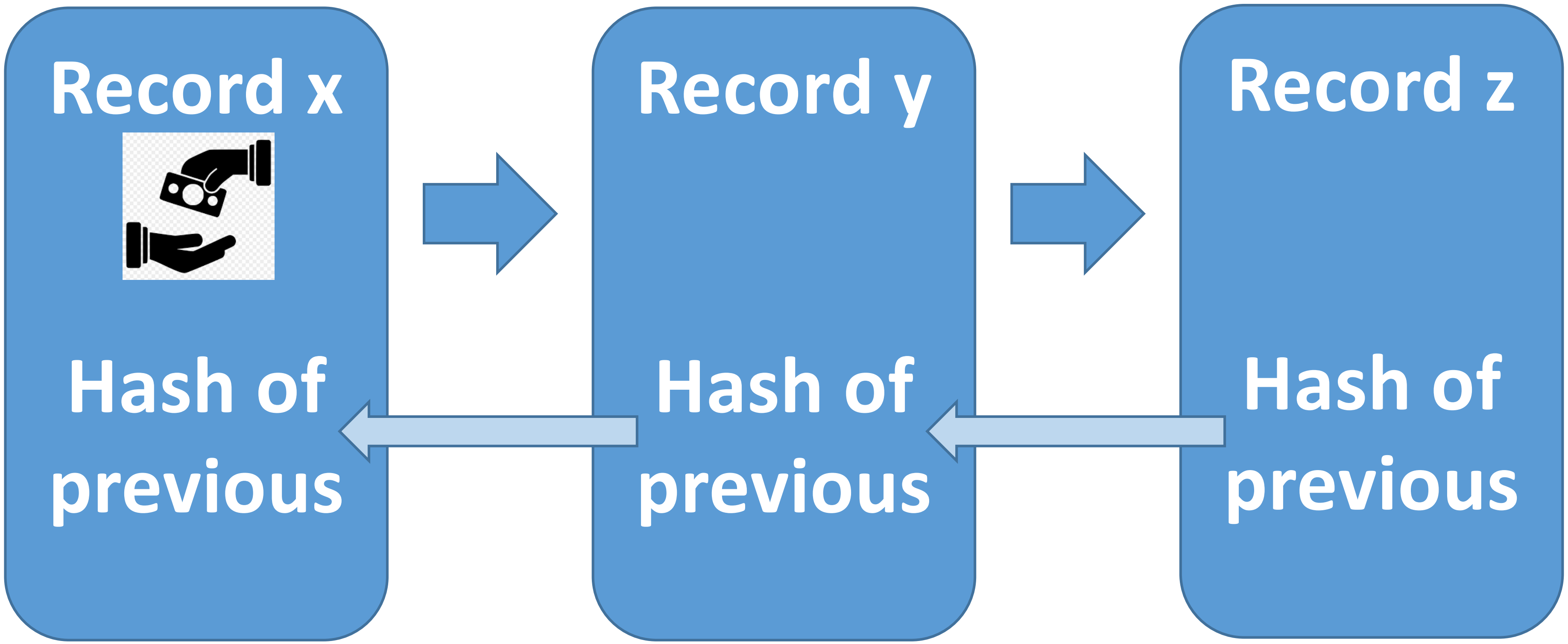


What is the blockhain?

Merely a list of events that were recorded



But also append-only and tamper-proof



Origins associated with Bitcoin, but…

Video TV News Tech Rec Room Food World News

MOTHERBOARD
TECH BY VICE

The World's Oldest Blockchain Has Been Hiding in the New York Times Since 1995

This really gives a new meaning to the "paper of record."

J. Cryptology (1991) 3: 99–111

Journal of Cryptology

How To Time-Stamp a Digital Document[1]

Stuart Haber and W. Scott Stornetta
Bellcore, 445 South Street,
Morristown, NJ 07960-1910, U.S.A.
stuart@bellcore.com stornetta@bellcore.com

Abstract. The prospect of a world in which all text, audio, picture, and video

NOTICES & LOST AND FOUND
(5100-5102)

Universal Registry Entries:
Zone 2 -
dS8492cgVOFAoP9kyE1XzMOrQ
HgEwzkVbVafNvlkUz99qvq8/ME
p5y9EFSG8XxzMBalGQQ==
Zone 3 -
JnFCg+HCmvhj8GmmUP7VZna71
NgZup/RfuKUQNzCHWXMuqLK
durxHQV5pSHLqBGPRIy+mg==
These base64-encoded values represent the combined fingerprints of all digital records notarized by Surety between 2009-06-03Z 2009-06-09Z.
www.surety.com 571-748-5800

In whom do we trust?

- Publishing a weekly hash in the NYT was a way to avoid having centralized trust on Surety's internal records.
- In permissioned blockchains, this trust is replaced with voting
  - Provably correct if colluding attackers do not exceed 1/3 of the members
- Problem: in permissionless settings, anyone can vote. What is the meaning of “anyone”?

Depends on who (and when) you ask

- Voted in 1911 (using a loophole)

On the Internet, it's easy to get voting rights

- Entities != identities
- Easy to create multiple public/private key pairs or IP addresses
- Known as a Sybil attack
- Force any decision, e.g.:
  - Double-spend
  - Break agreement

Proof-of-Work

- Invented in the context of fighting e-mail spam [Hashcash 1997]
  - Every time an e-mail is about to be sent, sender must solve a puzzle that requires computational power
  - Puzzle depends on the contents of the e-mail
    - Hard to produce – hence requires time
    - Easy to verify – if the puzzle is not correct, reject message
  - Limits the number of e-mails that can be sent
- From the original bitcoin paper:

  “The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote.”

  “The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.”

Crypto puzzles used in PoW

- Find **nonce** such that
  - Hash(e-mail || nonce)
- has **X** leading zeros
  - Hash(”hello world” || 0000) = 0xfad....
  - Hash(”hello world” || 0001) = 0xdeb....

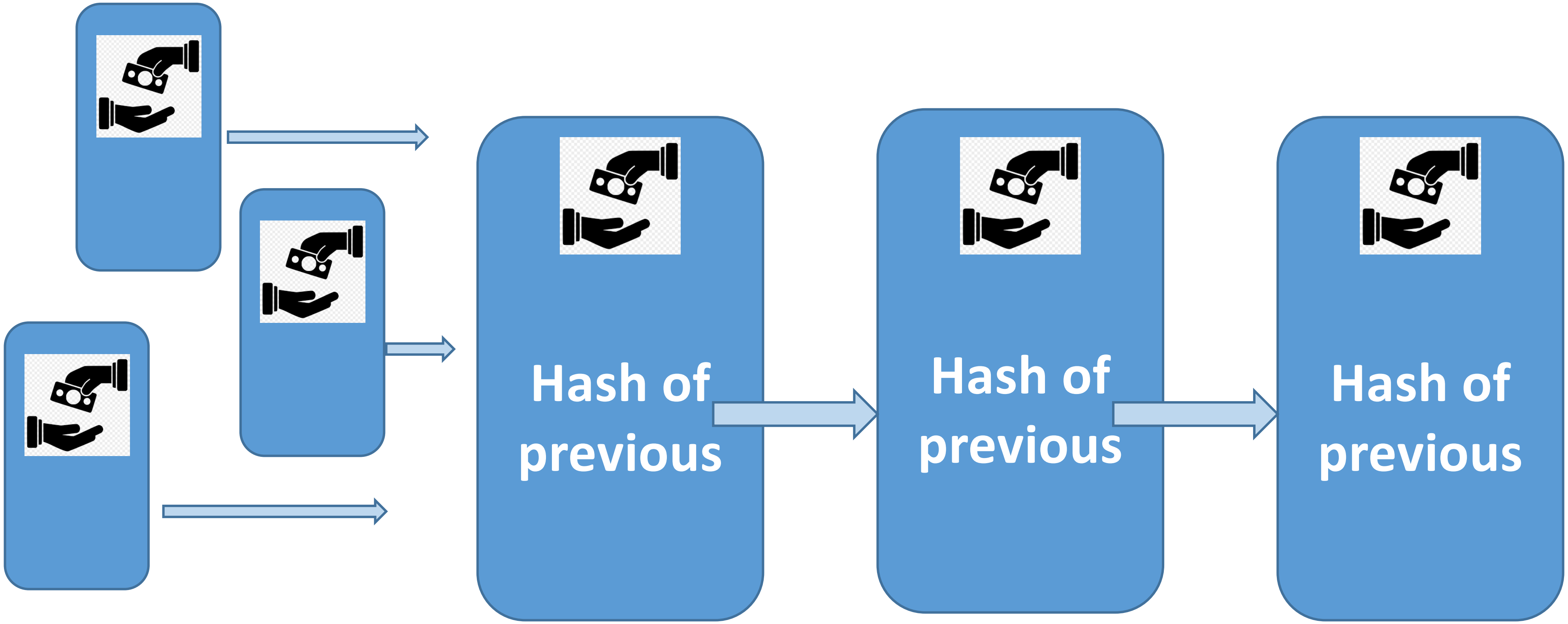
    …
  - Hash(”hello world” || 0999) = 0x**000**fe... <u>Ok for X=3</u>

- X is the difficulty of the puzzle

Proof-of-Work

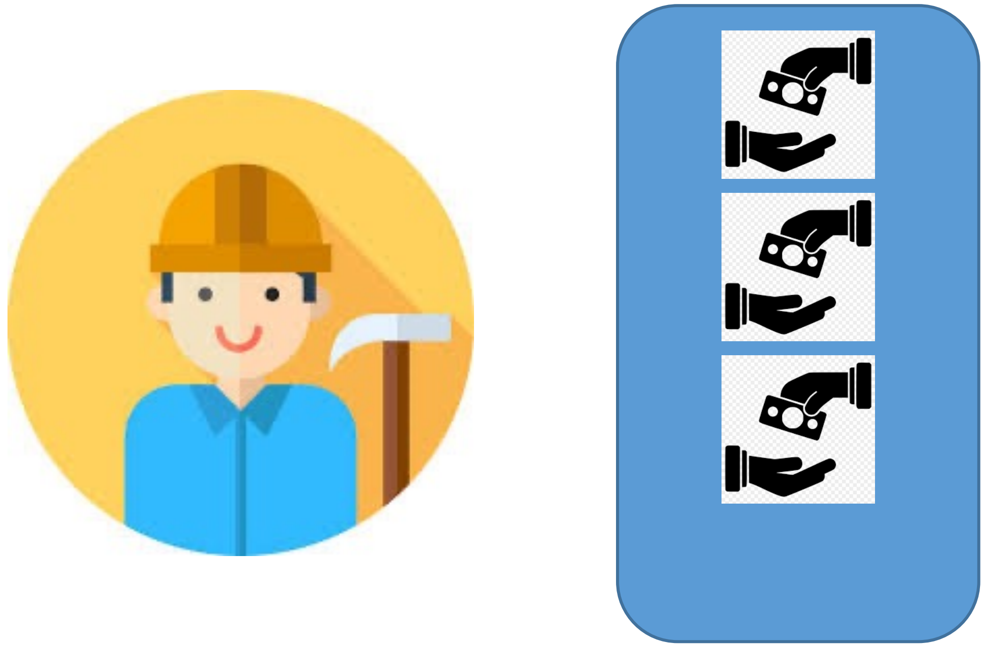
- A secure hash function is hard to invert
  - Best approach is to use brute force
  - Puzzle solver needs to try many combinations (which takes time) until finding the solution
  - Trying more combinations per unit of time implies more CPU power
- Puzzle verifier needs to execute the hash function only once to verify if the puzzle is correct
  - Verifying correctness of digest is very fast

How is PoW used by bitcoin?



- Everyone needs to agree on next record to be appended

Miners aggregate transactions in blocks (allows for better throughput)



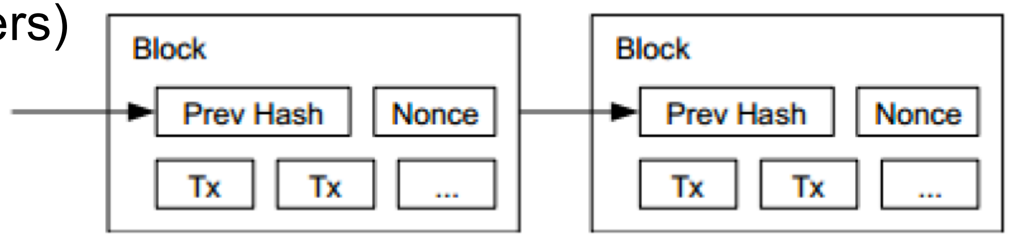
Miners compete to get block appended (winner gets reward, newly minted BTC)

PoW in bitcoin

- Each miner has a transaction pool
  - Set of transactions received over the network
  - Or created by the node itself
- Miners continually try to create a valid block by solving a puzzle:

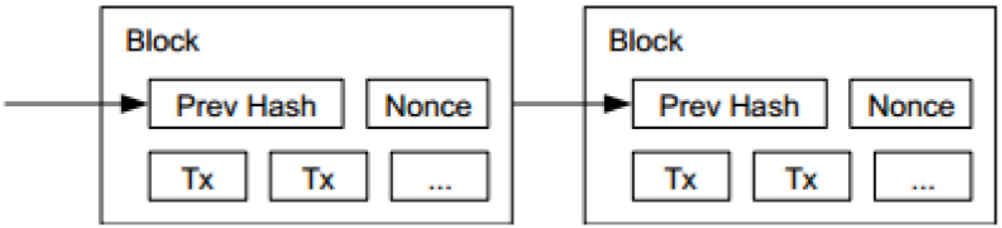
  SHA256(h || T || K || nonce) < D

  h – hash of last block; T – digest of txns; K – pub. key (owner gets reward); D – difficulty;
  - When solved, broadcast new block to all miners
- A block includes:
  - A subset of transactions, chosen by the miner, in the transaction pool
  - A pointer to the previous block
  - The identity of the miner (among others)
  - The solution to the puzzle

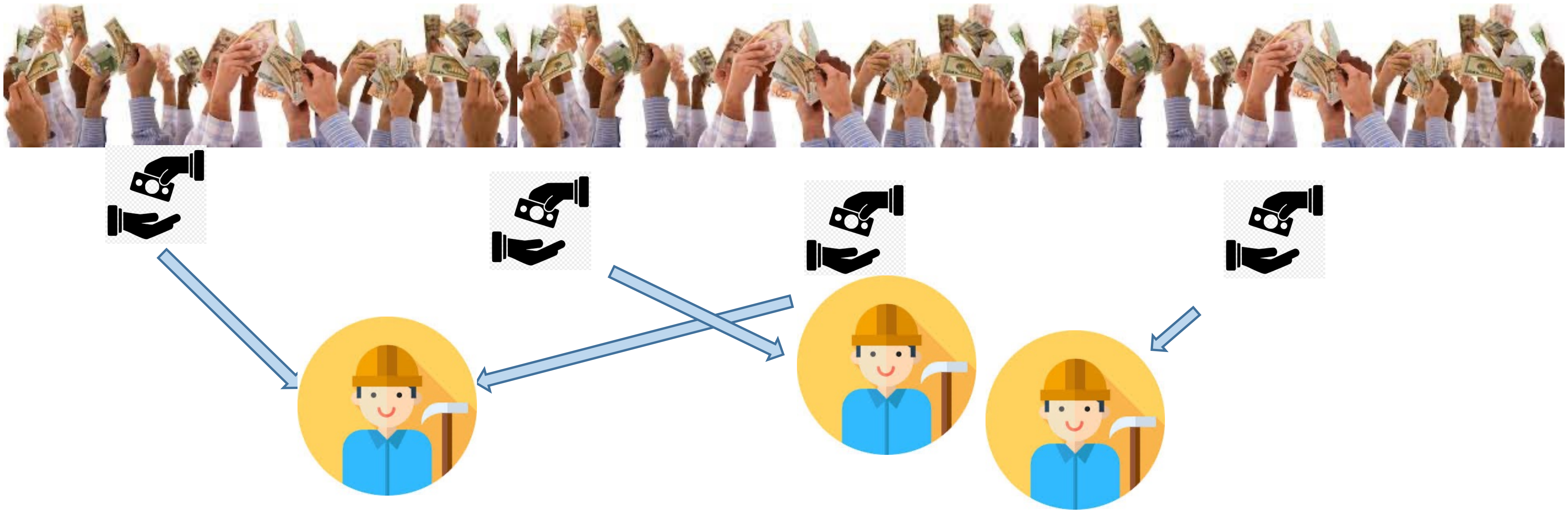


Block dissemination

- Every node keeps a copy of every block of transactions
  - Can be a scalability bottleneck
  - Led to distinction between full nodes and compact nodes
- Upon receiving a new block:
  - Miners validate the block (incl. money not being already spent)
  - If valid, they append it to the local blockchain
  - Start mining the next block with a pointer to the accepted block



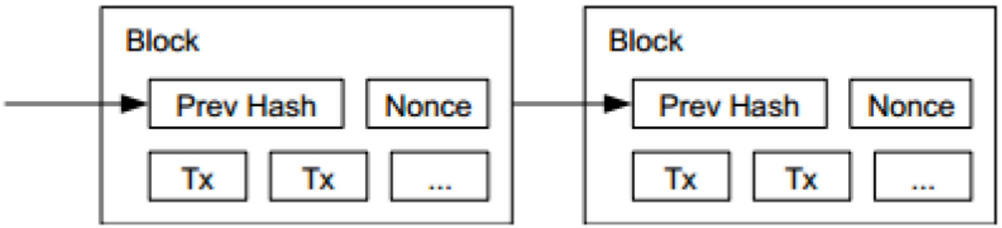
Clients != miners != compact nodes



- Clients submit transactions to miners (broadcast to all miners through p2p layer)
- Miners solve PoW
- Full nodes keep a complete copy of the blockchain
- Compact/lightweight nodes keeps the current state (more on this later) but only a small suffix of the blockchain

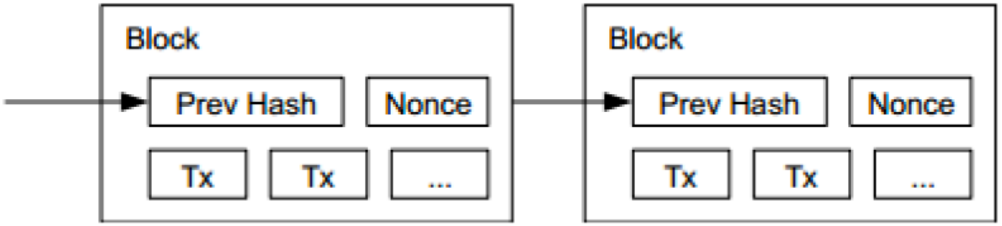
Properties

- Prevents Sybil Attacks
  - Solving the puzzle can effectively be seen as *vote*
  - Puzzle requires computational power – cannot be counterfeit without breaking the hash function
  - Controlling x% of the CPUs ~ controlling x% blocks
  - Still, some cryptocurrencies witnessed "**51% attacks**"



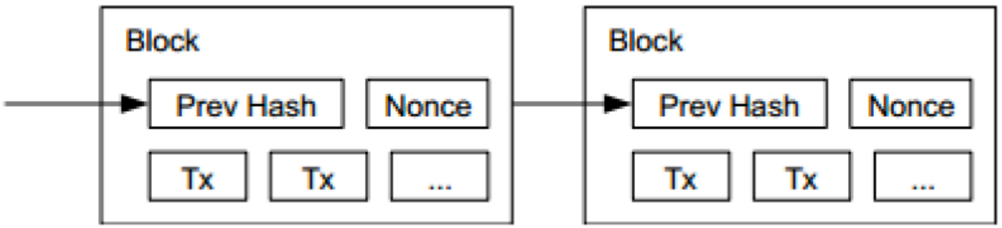
Properties

- Tamperproof
  - Pointer to the previous block
  - Changing a previous block would invalidate the subsequent chain
  - Rewriting a part of the chain requires solving all the associated puzzles
    - The longer the chain one wants to revert, the larger the needed computing power



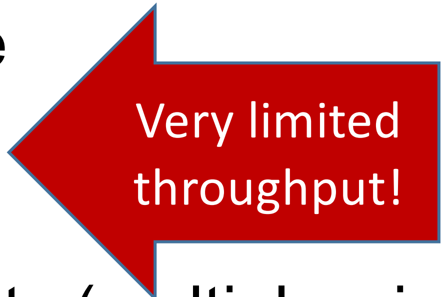
Properties

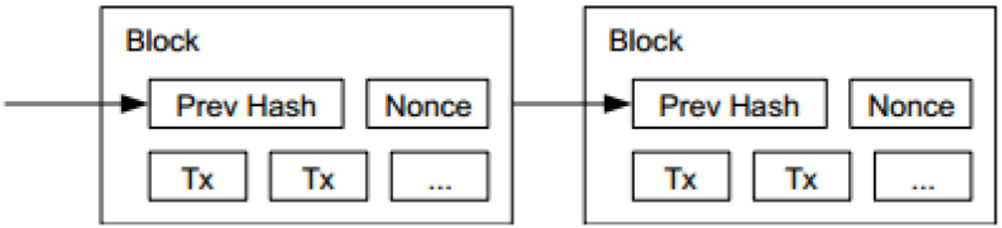
- Provides incentives for miners to participate
  - The miner of a block creates (mines) a new coin
  - Coin is given to the miner
  - Transactions in the block also pay a transaction fee to the miner



Properties

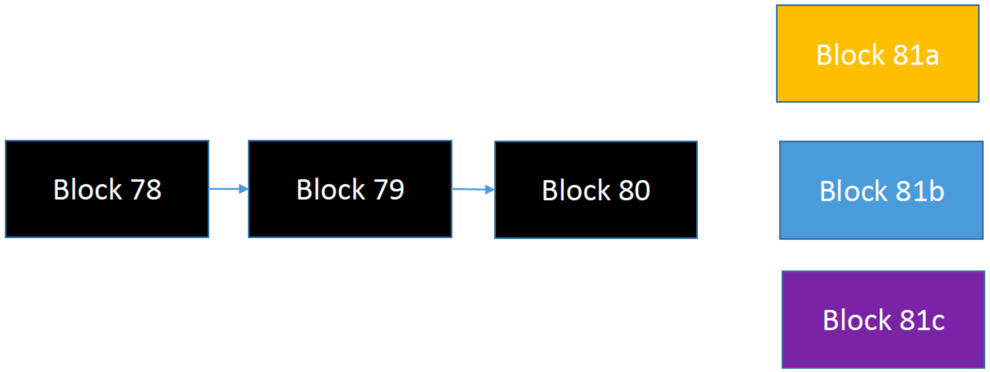
- Generating a block can be seen as a random Poisson process
  - Puzzle difficulty is automatically adjusted to meet a target rate of new blocks per unit of time
    - Bitcoin ~1 block every 10 minutes
    - Ethereum ~1 block every 20 seconds
  - Required to avoid frequent conflicts (multiple winners), but problematic as system becomes more widely used





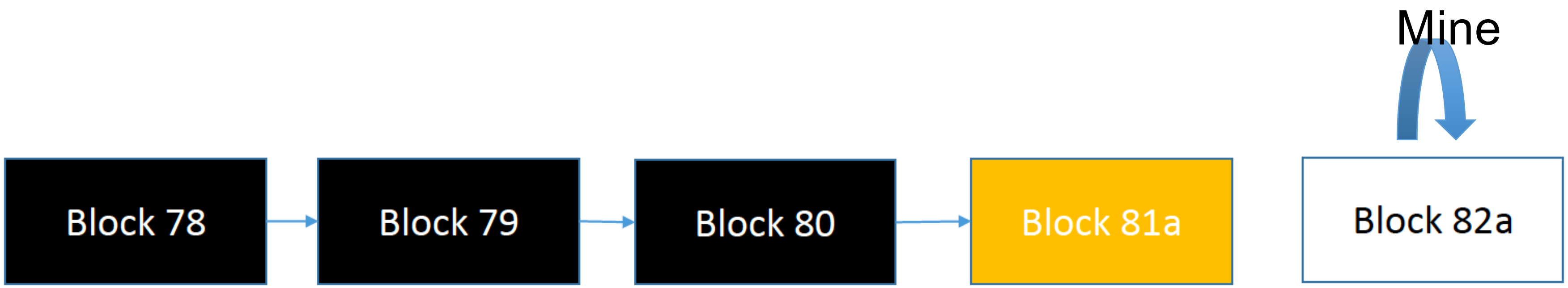
A fork in the road…

- What if a miner receives several valid new blocks to be appended
  - Which one to choose?
  - Miners can choose any block they want
- Resulting in miners trying to mine in parallel chains
  - Forks!



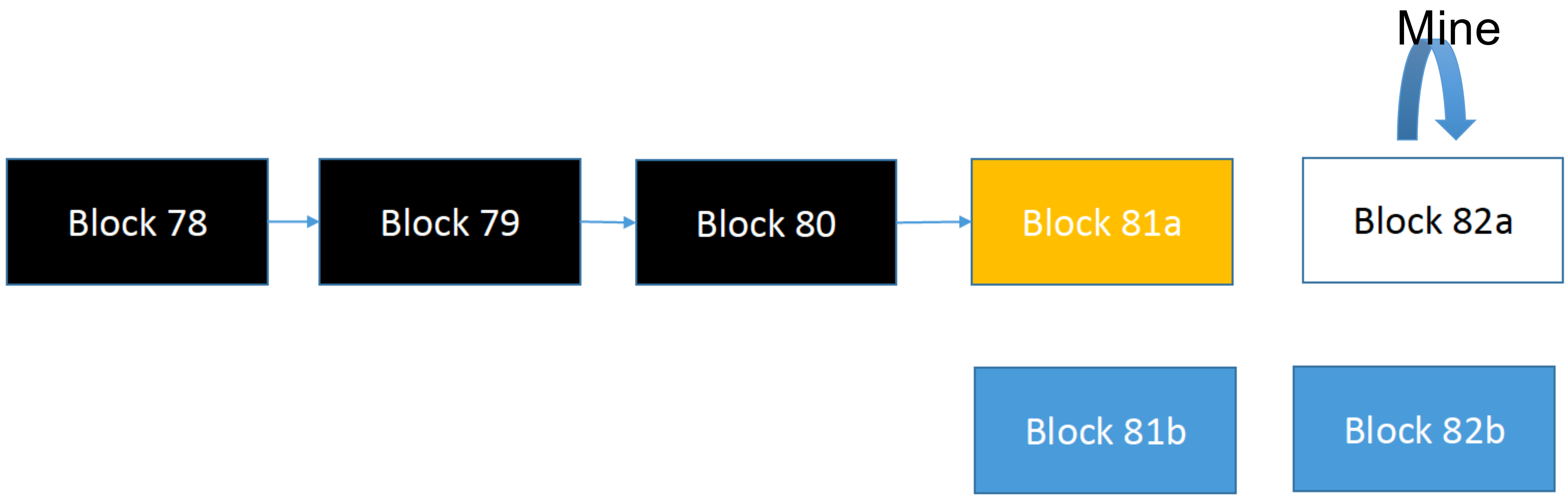
A fork in the road…

- Start mining on block 82a



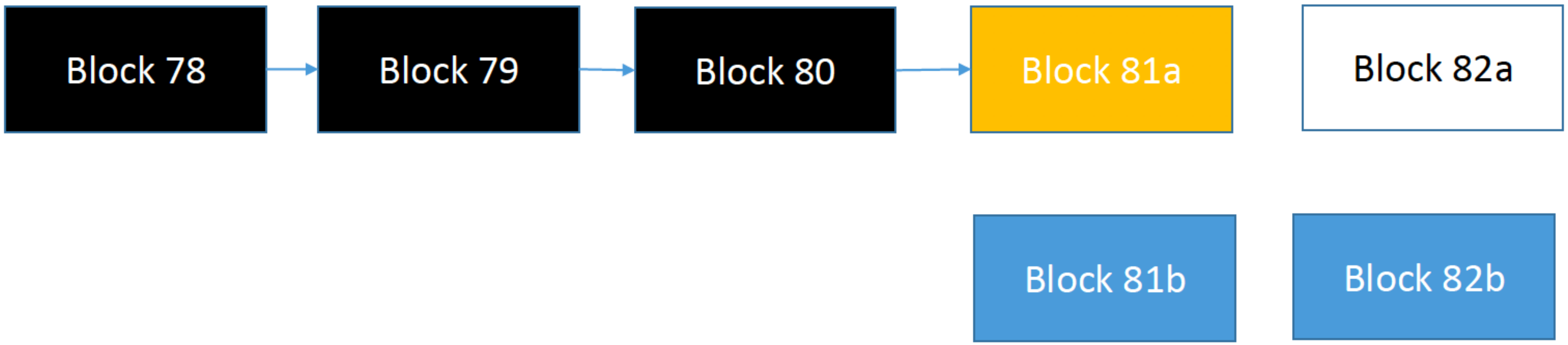
A fork in the road…

- What if blocks 81b and 82b appear?



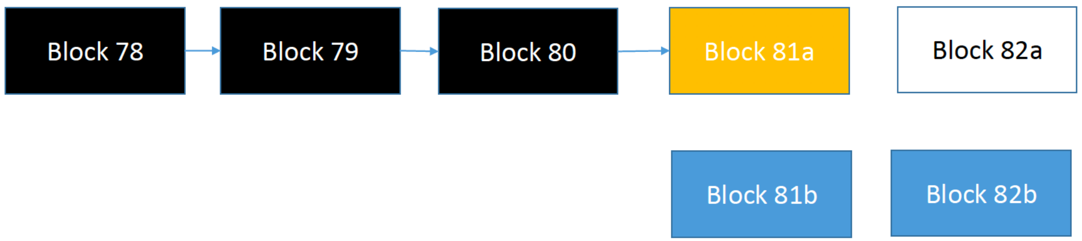
Longest chain rule

- Try to mine new Block 83
- Always build on the longest chain, i.e., the one with the most PoW
- Reflects the will of honest miners, forcing dishonest ones to out-compute all the honest miners



Problem: lack of “finality”

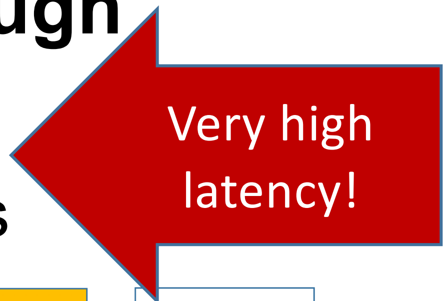
- Each miner picks the transactions to be included in a block
  - Thus, transactions in Blocks 81a and 81b differ
- Suppose you have an e-commerce site
  - You see a payment transaction Tx1 in Block 81a
  - Ship the goods
  - Tx1 no longer appears in neither Block 81b nor 82b
  - What to do?

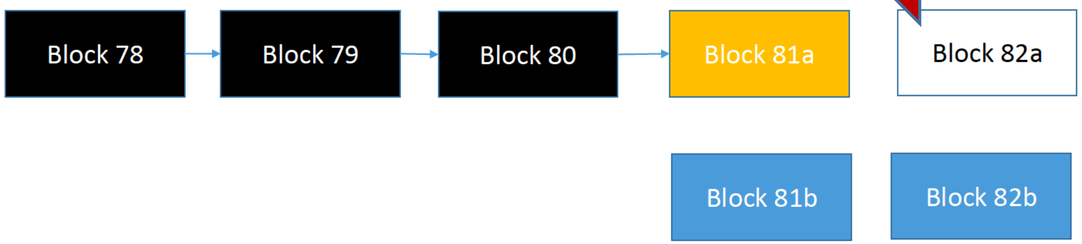


Embrace the probabilistic nature

<u>Probabilistic</u> solution

- Need to wait for the tx's block to be deep *enough* in the chain
- *Enough* meaning the **probability** of a given chain suffix being replaced is **low enough**
  - Bitcoin ~6blocks → 1 hour
  - Ethereum ~ 11blocks → 4 minutes





Double-spending attacks

Can an attacker deliberately achieve the following:

1. Approve a transaction tx that moves coin to merchant
2. Merchant waits for tx's block to be deep enough and ships goods
3. Attacker releases a forked chain of blocks, including moving the same coin to another merchant
4. Forked chain of blocks is longer than first chain

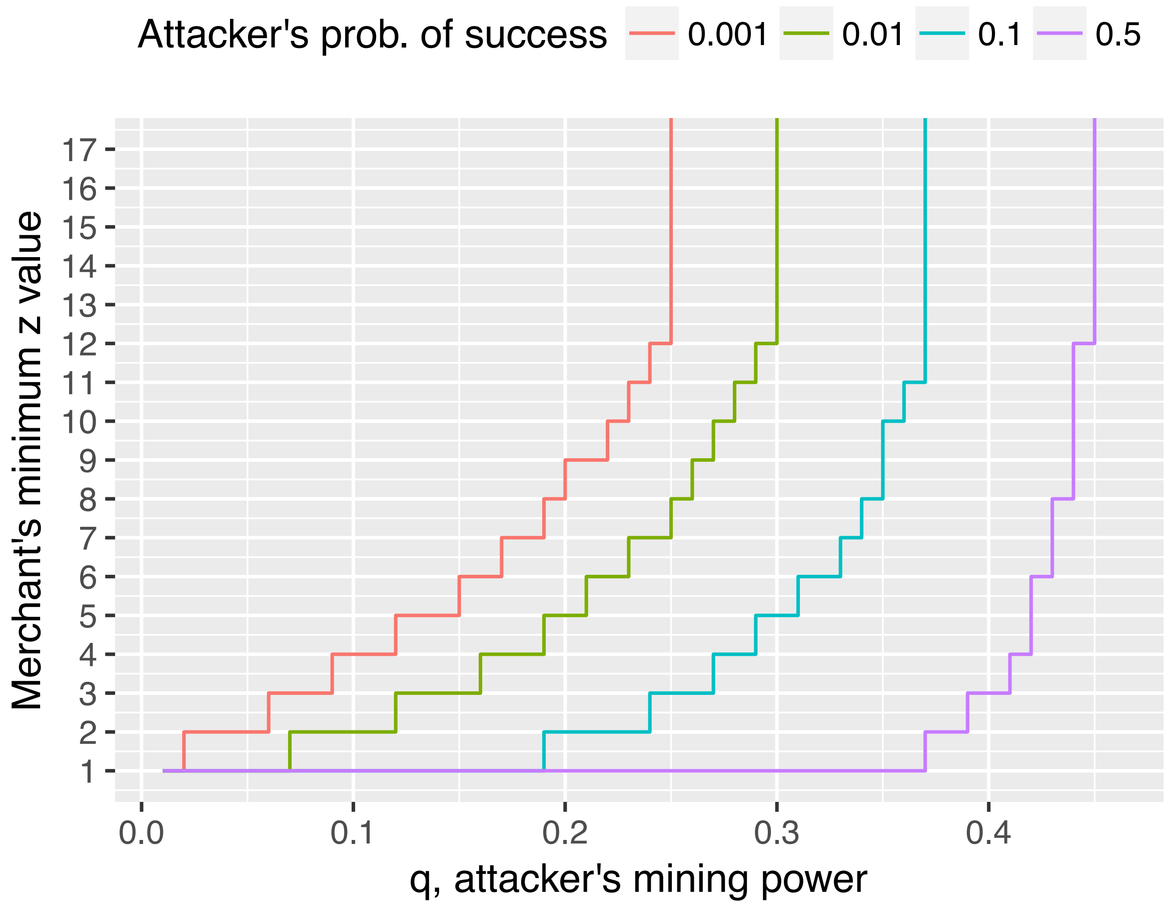
Reverting a chain of z blocks

- Assume an attacker tries to remove one of his txs that is stored in a block at depth z
- Equivalent to Binomial Random Walk (*):
  - p = prob. some honest node finds the next block
  - q = prob. the attacker node finds the next block
- Probability, Q(z), that attacker will ever catch up from z blocks behind:
  - 1, if $p <= q$
  - $(q/p)^z$, if $p>q$

Any attacker with >50% than the total computational resources can eventually catch up even if starting from an arbitrarily large gap (z)

(*) More details in: An Explanation of Nakamoto's Analysis of Double-spend Attacks, Ozisik and Levine

How safe is Blockchain?




```
P < 0.001
q=0.10    z=5
q=0.15    z=8
q=0.20    z=11
q=0.25    z=15
q=0.30    z=24
q=0.35    z=41
q=0.40    z=89
q=0.45    z=340
```

Sources: Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto;
An Explanation of Nakamoto's Analysis of Double-spend Attacks, Ozisik and Levine

Rental attack

- Rent VMs to launch a 51% attack
- Overly expensive for Bitcoin and Ethereum
  - but turns out to be surprisingly cheap for other coins

More sophisticated form (Finney attack)

- Suppose merchant accepts unconfirmed transactions
- Attacker successfully pre-mines a block including a transaction that sends some of his coins back to self, without broadcasting it
- Attacker uses same coins for purchase at this merchant
- After the merchant accepts them, attacker broadcasts his block (overriding the payment)

Higher bar for mining → mining pools

- CPU → GPU → ASIC
  - Gets more effective and sophisticated as currency value increases
- As mining got more difficult, mining pools emerged
  - increase chances, hedge risk, split the reward among pool members
  - Strong incentive to participate
- Mining pools grew significantly (possibly reaching more than 51% of mining power)
  - Implies that bitcoin may not be that decentralized after all

Possible attack from mining pool: Feather-forking attack

- Suppose mining pool controls, e.g., only 20%, and wants to censor Alice
- Mining pool announces it will refuse to mine chains that include Alice in last N blocks
- Creates a disincentive for every miner to work on blocks that include Alice's txns, as these may fail to be included in the longest chain

Selfish mining

- Pool nodes withold newly found blocks, continuing to work on their private chain
  - i.e., they create their own private fork
- When public chain catches up, publish private chain
- Creates an incentive to join the selfish mining pool (more profitable than following the protocol)

Let others do the mining?

- Malware can conduct mining without the owner of the machine noticing it
- But even without using malware, just by visiting a website, "drive-by mining" is possible (using JavaScript or WebAssembly)

MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense

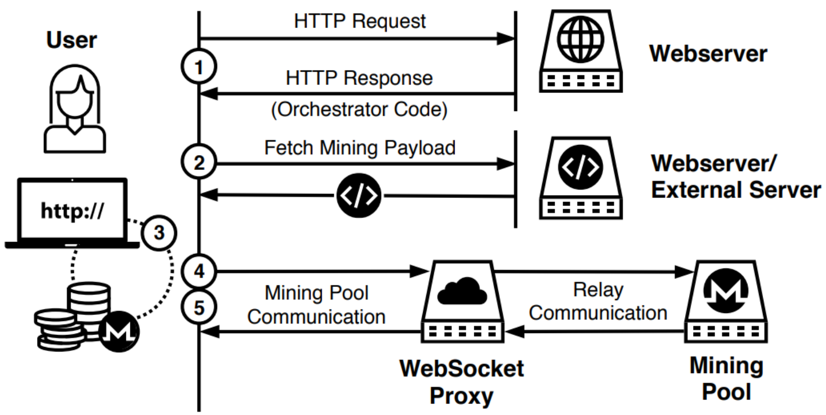


Table 1: Summary of our dataset and key findings.

Crawling period	March 12, 2018 – March 19, 2018
# of crawled websites	991,513
# of drive-by mining websites	1,735 (0.18%)
# of drive-by mining services	28
# of drive-by mining campaigns	20
# of websites in biggest campaign	139
Estimated overall profit	US$ 188,878.84
Most profitable/biggest campaign	US$ 31,060.80
Most profitable website	US$ 17,166.97

Privacy aspects

- How was privacy of buyers and sellers handled in traditional commercial transactions?
- How is it handled in bitcoin?

Last but not least

The New York Times

Bitcoin Uses More Electricity Than Many Countries. How Is That Possible?

In 2009, you could mine one Bitcoin using a setup like this in your living room.

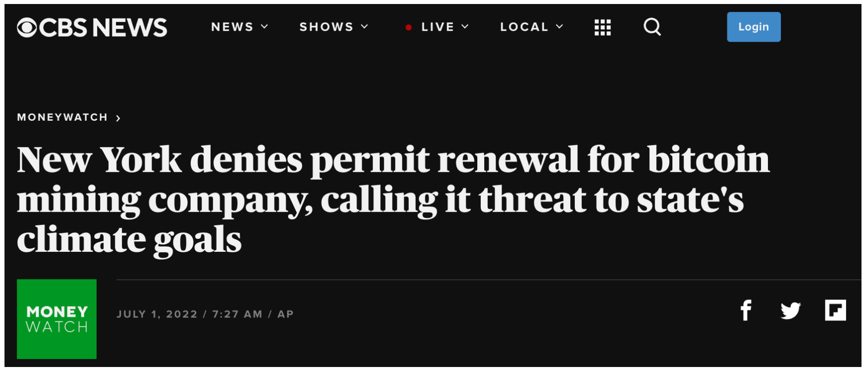
Today, you'd need a room full of specialized machines, each costing thousands of dollars.

The dark side of PoW

- PoW for cryptocurrencies consumes more energy than Argentina, Bitcoin is comparable to Greece
- ~2X Google, Meta, Microsoft, Amazon, Apple (combined)
  - While their value to society is incomparable
- Bitcoin emits 65 megatons of CO_2 annualy
- Single bitcoin transaction emits almost 1 ton of CO2 (6 orders of magnitude more than a credit card transaction)

Becoming less and less green

- Until recently, 75% of mining was in China, due to cheap electricity and hardware
  - Government banned it in 2021
- Nowadays 1/3 of mining is in the US, 2nd place is Kazakhstan
  - Shifted from hydropower to gas and coal
  - Reviving old, polluting power plants for mining
  - Regulation is now starting to kick in



Recall consensus specification

- **Termination:** Correct processes eventually decide.
- **Validity:** Any value decided is a value proposed
- **Integrity:** No correct process decides twice.
- **Agreement:** No two correct processes decide differently.

PoW blockchains vs consensus

- Which properties does PoW consensus satisfy?
- Consider:
  - **decide** as accepting a block at the blockchain head
  - **propose** as submitting a transaction to be eventually included in a blockchain block

PoW blockchains vs consensus

- **Termination:** Correct processes eventually decide.
  - Arguably, eventually a new block is appended to the chain
  - How fast depends on the solvability of the crypto-puzzle

PoW blockchains vs consensus

- **Integrity:** No correct process decides twice.
  - No. The accepted block at a given weight might change several times due to forks

PoW blockchains vs consensus

- **Agreement:** No two correct processes decide differently.
    - No. The accepted block at a given weight might change several times due to forks

PoW blockchains vs consensus

- **Validity:** any value decided is a value proposed
  - No, if Byzantine processes can generate transactions that were never proposed
- **Weak Validity:** if some processes are faulty, an arbitrary value may be decided
  - OK, although block structure cannot be **completely** arbitrary:
    - e.g., blocks containing illegal transactions are rejected
- **Strong Validity:** a correct process may only decide a value that was proposed by some correct process or the special value ☐.
  - No, if Byz. processes can generate transactions that were never proposed

PoW blockchains vs. consensus

- Trade-off between liveness and safety
- Classical consensus emphasizes safety
  - Agreement is always respected but might not terminate
- PoW consensus emphasizes liveness
  - The system always makes progress (i.e., decides on *something*) but safety is at stake:
    - Guarantees are only of **probabilistic** nature
    - FLP is not contradicted… why?

Alternative Approaches

Proof-of-Stake

- Use stake (coins) in the system as the non-counterfeitable resource
- Users *vote* with their coins in the system rather than with CPU power
- Chance to select the next block proportional to the stake size
- Problems
  - Rich get richer
  - Proved to converge only in specific scenarios
- Ethereum moved to this model (more on this next lecture)

Alternative Approaches

Proof-of-Space

- Use disk space as the non-counterfeitable resource
- Users pre-generate a very large data structure in disk
- Subsequently answer queries for random subsets of the data
- Problems
  - Requires additional techniques – small PoW, PoET
- Examples: SpaceMint; Chia

Alternative Approaches

Proof-of-Elapsed-Time

- Relies on a Trusted Execution Environment to elect a leader that will choose the next block
  - Users “sleep” for a random period
  - User who wakes up first can select the next block to be include in the blockchain
    - limits the rate at which malicious nodes can generate blocks

- Problems
  - Need to rely on trusted hardware, e.g., Intel SGX, ARM Trust
    - Which has known vulnerabilities

Alternative Approaches

Committee based

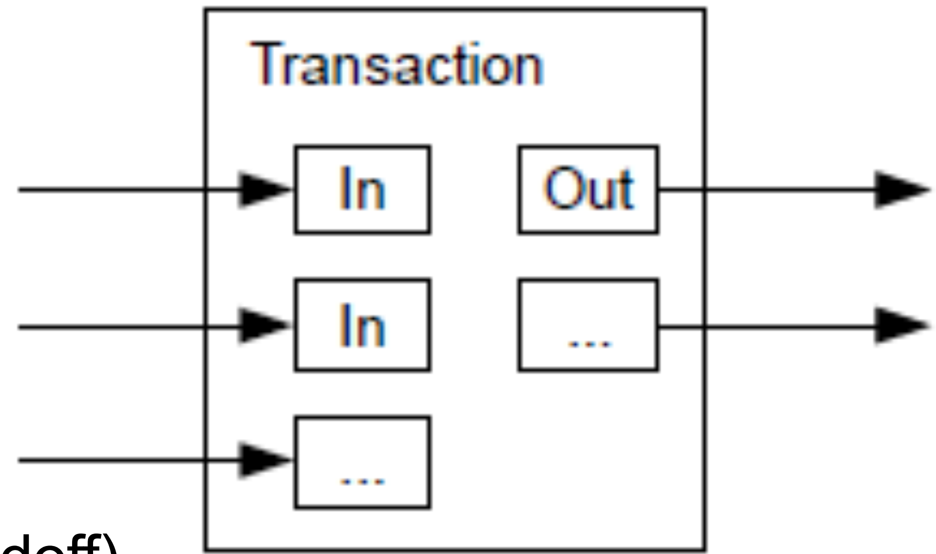
- Elect a committee among the set of participants
  - Using for instance PoW
- Committee relies on classical consensus to select the next block

- Problems
  - Committee members can behave arbitrarily
  - Committee members prone to attacks
- Examples: Solida, Algorand

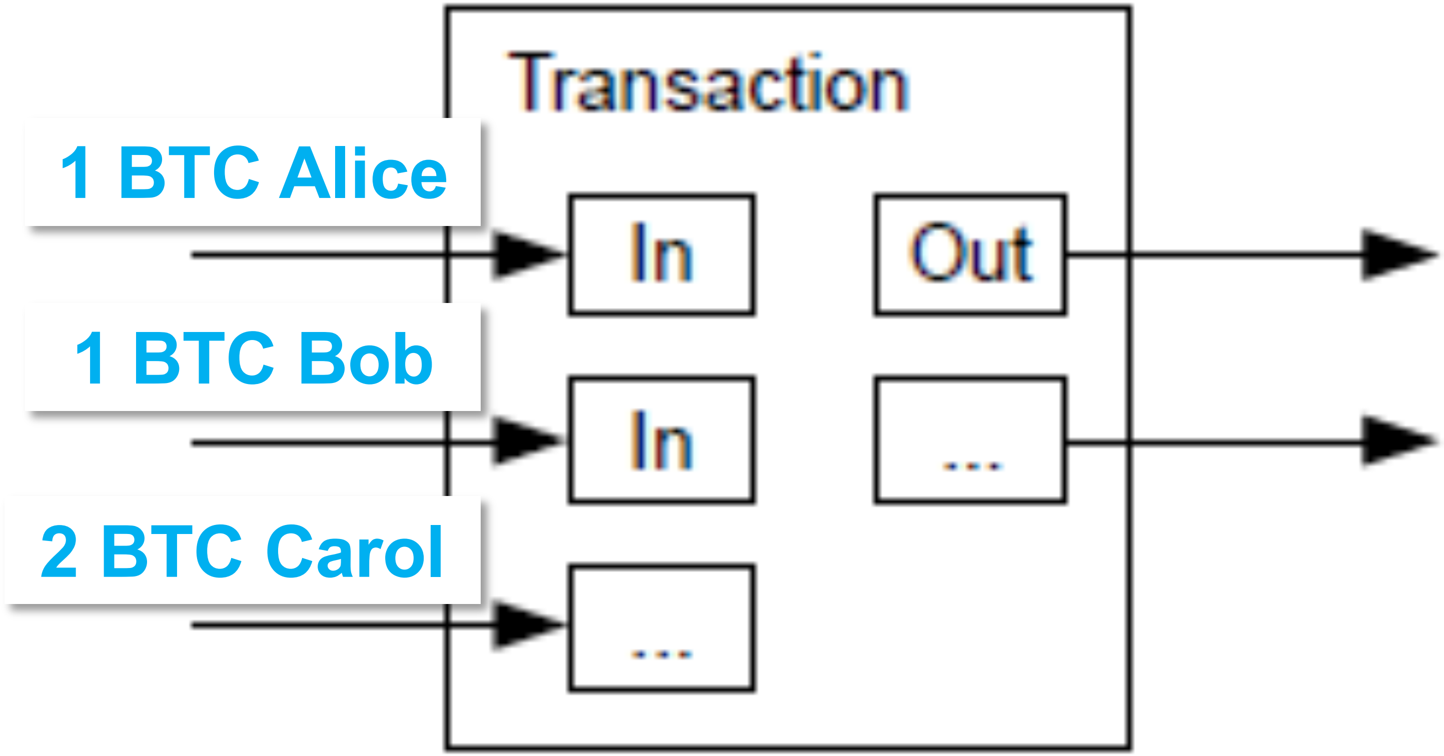
Bitcoin data structures

Each block in the blockchain is a set of transactions

- Transactions can have multiple inputs and <=2 outputs
- Possible to specify a set of deterministic operations, with conditions on transfer (smart contracts, later this week)
- Each input describes debit
  - Sender → signs handoff of amount (signed with private key)
- Each output describes credit
  - Recipient → public key of entity (corresponding to private key that later can sign the subsequent handoff)



Bitcoin UTXO Model



UTXO model

- Blockchain stores all transaction since beginning of bitcoin (hundreds of GBs of information, increases as more txns occur)
- Additionally, keeps track of outputs that haven’t been spent yet (about 4 GB of information, increases as more users join)
  - Called “unspent transaction output” (UTXO)
  - “Soft state” → can be reconstructed and verified from blockchain
  - Alternative would be to store account balances (has pros and cons)
- New transactions remove UTXOs from this set, and create new ones that are added to the set

Bitcoin scalability and Utreexo

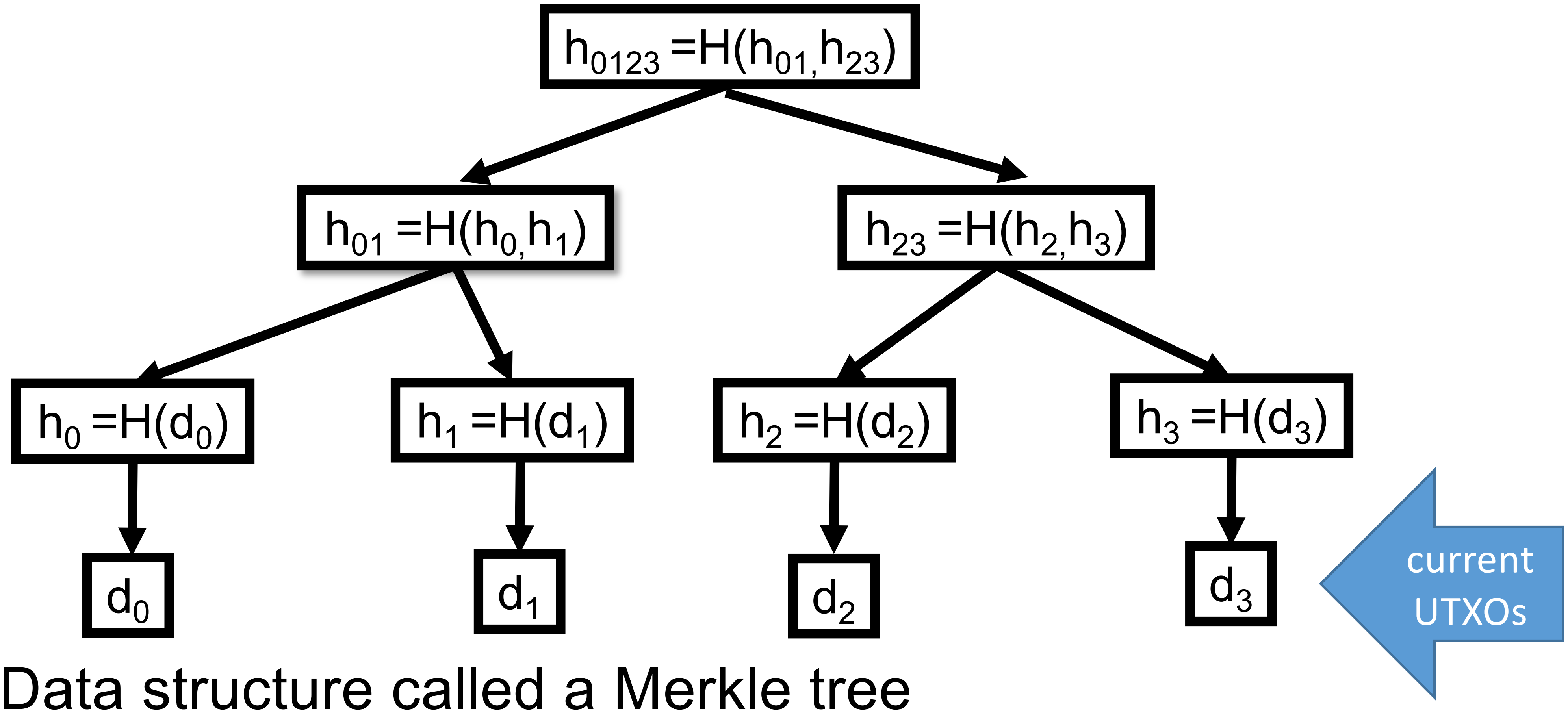
- Resources required for running a full node are a problem (barrier to entering the system)
- Initial block download can take 6 hours (SSD) to 1 week (HDD), but this is only done once per node
- Continuously maintaining a (growing) set of all UTXOs is more of a concern
- Light clients → do not store state nor validate transactions
  - At the cost of weaker security properties than full nodes
- Led to the proposal of Utreexo
  - Make it easier to run full nodes
  - New set of data structures, and new type of node (compact state node)

Compact state nodes using Utreexo

- Observation: most UTXOs aren’t needed for years (from creation until being spent)
  - No need to store all the UTXOs
- Store only a summary (in crypto terms, an accumulator)
- Require transaction issuer to send additional information (proofs) to verify transactions
- Trades storage requirements for bandwidth (mostly) and CPU (not much)

How to compute a summary of UTXOs?

(simplified)



- Data structure called a Merkle tree
- Compact nodes only store root of the tree

Upon receiving a UTXO, how to check it is part of the current set?

- Transaction issuer must send proof
- E.g., proof that d_2 is in the set of UTXOs?
- "d_2, h_3, h_{01}"
- From this info, receiver computes h_2, h_{23}, h_{0123}
- Then checks whether h_{0123} matches the currently held root of the tree
- (Tree maintenance after adds and deletes + supporting legacy transactions gets a bit more intricate – see paper for details)

Summary

- Classical consensus and permissionless blockchains address a different set of requirements
  - Safety vs Liveness tradeoff
- Many open challenges
  - Transaction latency and throughput
  - Application safety (correctness, censorship, etc)
  - Energy concerns
  - and many others

Further Reading

- S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System
- A. Ozisik, B. Levine. An Explanation of Nakamoto's Analysis of Double-spend Attacks. arXiv:1701.03977
- I. Eyal, E. G. Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. arXiv:1311.0243
- R Konoth, E Vineti, V Moonsamy, M Lindorfer, C Kruegel, H Bos, G Vigna. MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense. CCS 2018: 1714-1730
- Y Gilad, R Hemo, S Micali, G Vlachos, N Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. SOSP 2017.
- Thaddeus Dryja. Utreexo: A dynamic hash-based accumulator optimized for the Bitcoin UTXO set. Cryptology ePrint Archive, Paper 2019/611

Acknowledgements

- Rachid Guerraoui, EPFL
- Maurice Herlihy, Brown U.