Case Study 5: Bitcoin: A Peer-to-Peer Electronic Cash System

Ashvin Goel

Electrical and Computer Engineering
University of Toronto

Distributed Systems
ECE419

Thanks to MIT 6.824 course notes and many others

Overview

- Introduction to Bitcoin
- Bitcoin transactions
- Bitcoin blockchain
- Bitcoin mining

Why Bitcoin?

The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust. Banks must be trusted to hold our money and transfer it electronically, but they lend it out in waves of credit bubbles with barely a fraction in reserve. We have to trust them with our privacy, trust them not to let identity thieves drain our accounts. Their massive overhead costs make micropayments impossible.

- Satoshi Nakamoto, 2009

Why Bitcoin?

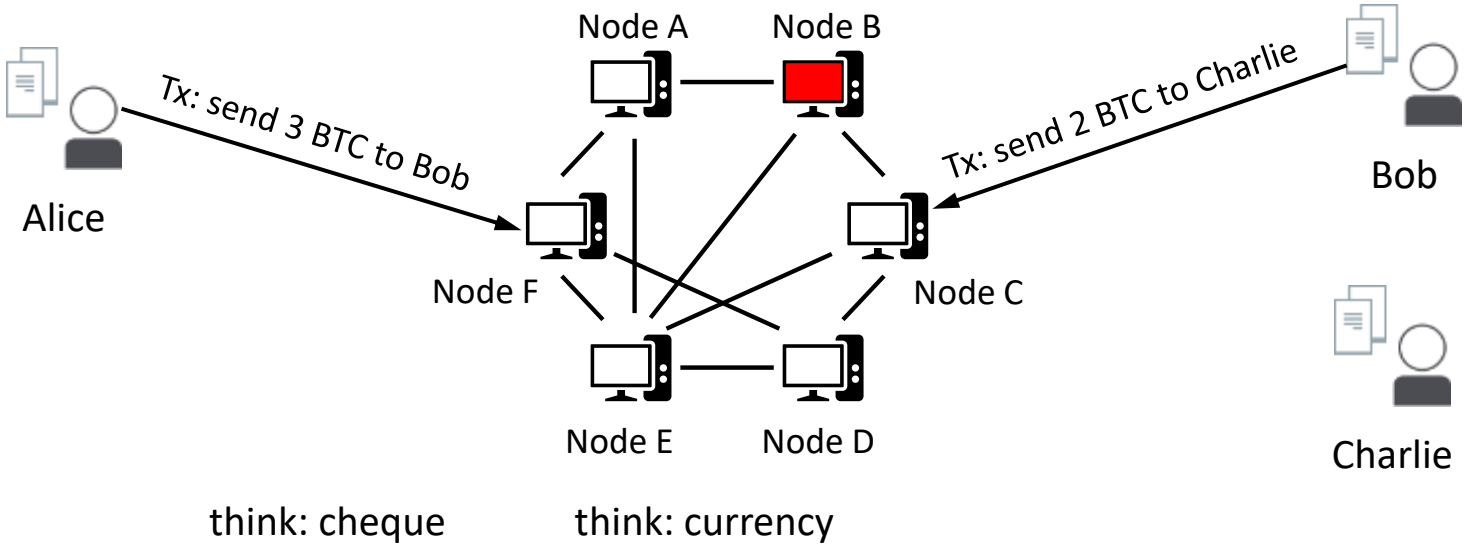
A generation ago, multi-user time-sharing computer systems had a similar problem. Before strong encryption, users had to rely on password protection to secure their files, placing trust in the system administrator to keep their information private. Privacy could always be overridden by the admin based on his judgment call weighing the principle of privacy against other concerns, or at the behest of his superiors. Then strong encryption became available to the masses, and trust was no longer required. Data could be secured in a way that was physically impossible for others to access, no matter for what reason, no matter how good the excuse, no matter what.

It's time we had the same thing for money. With e-currency based on cryptographic proof, without the need to trust a third party middleman, money can be secure and transactions effortless.

- Satoshi Nakamoto, 2009

What is Bitcoin?

- A decentralized digital currency system
  - Think: bank running on an ad hoc network of nodes
  - No central bank, i.e., no single administrative control
  - Nodes are untrusted, some may be malicious (byzantine)
  - Clients send financial transactions to network



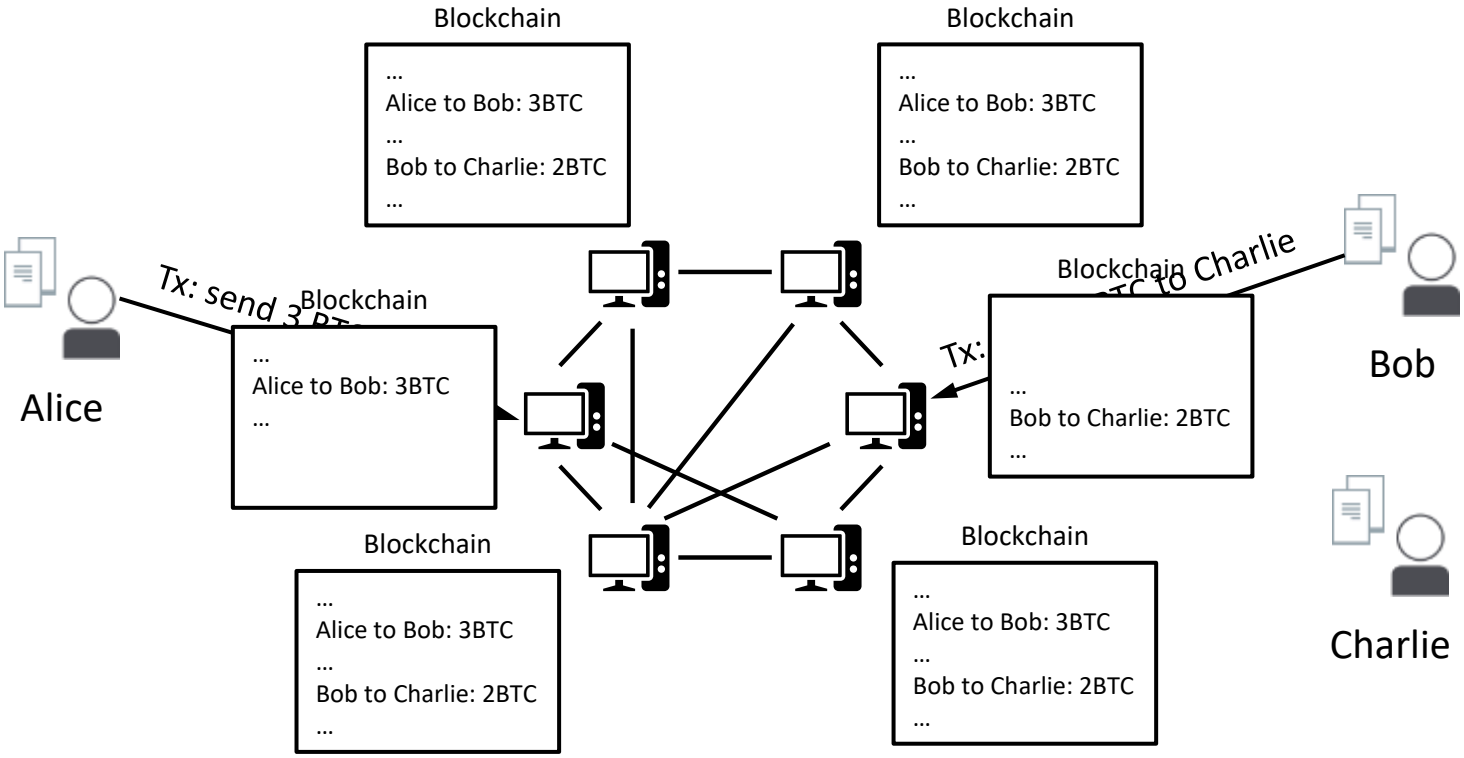
- A transaction moves bitcoins (BTC) from one account to another

What is Bitcoin?

think: cheque book

- Nodes log transactions in a public ledger called blockchain

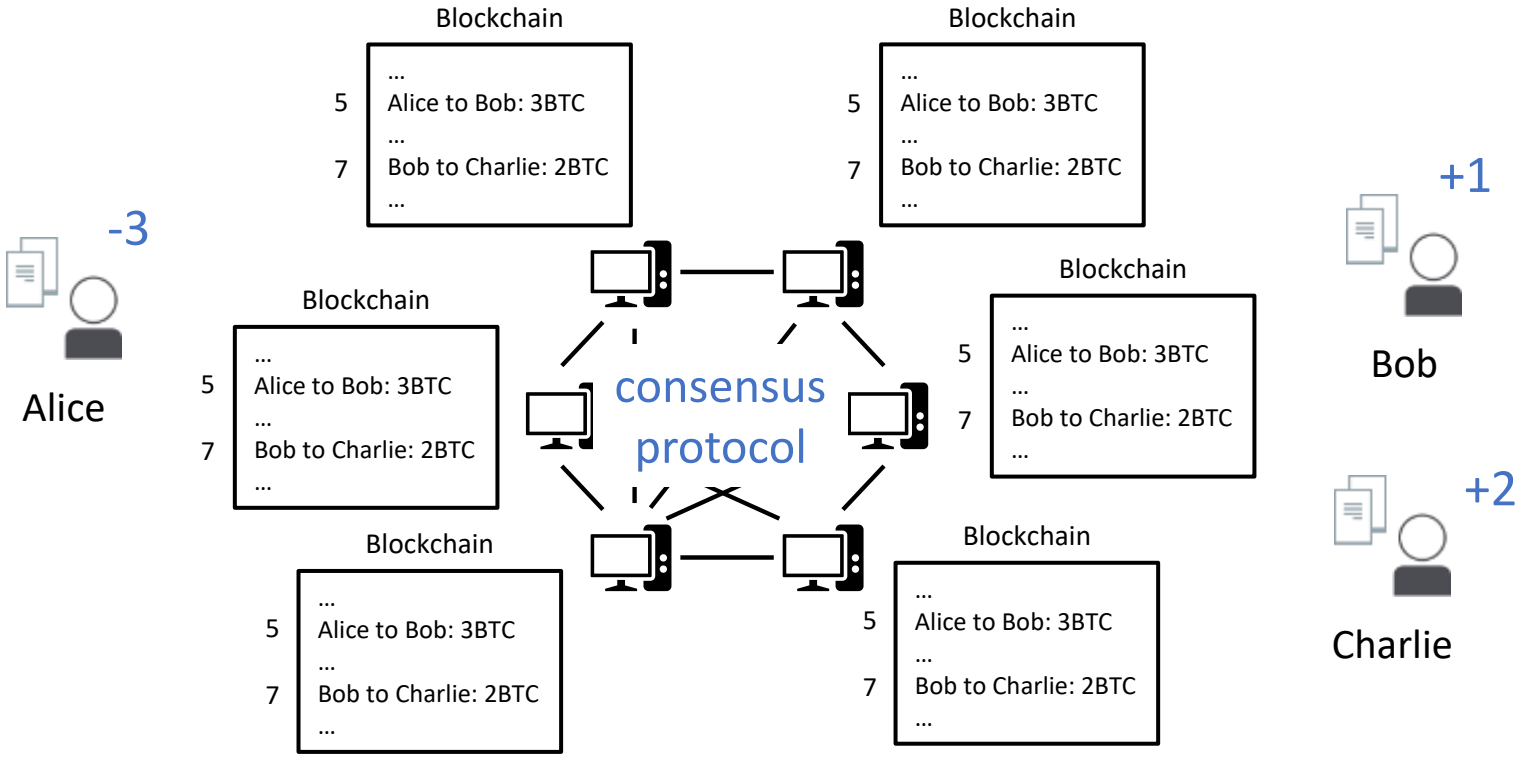
- Each node keeps a replica of the blockchain



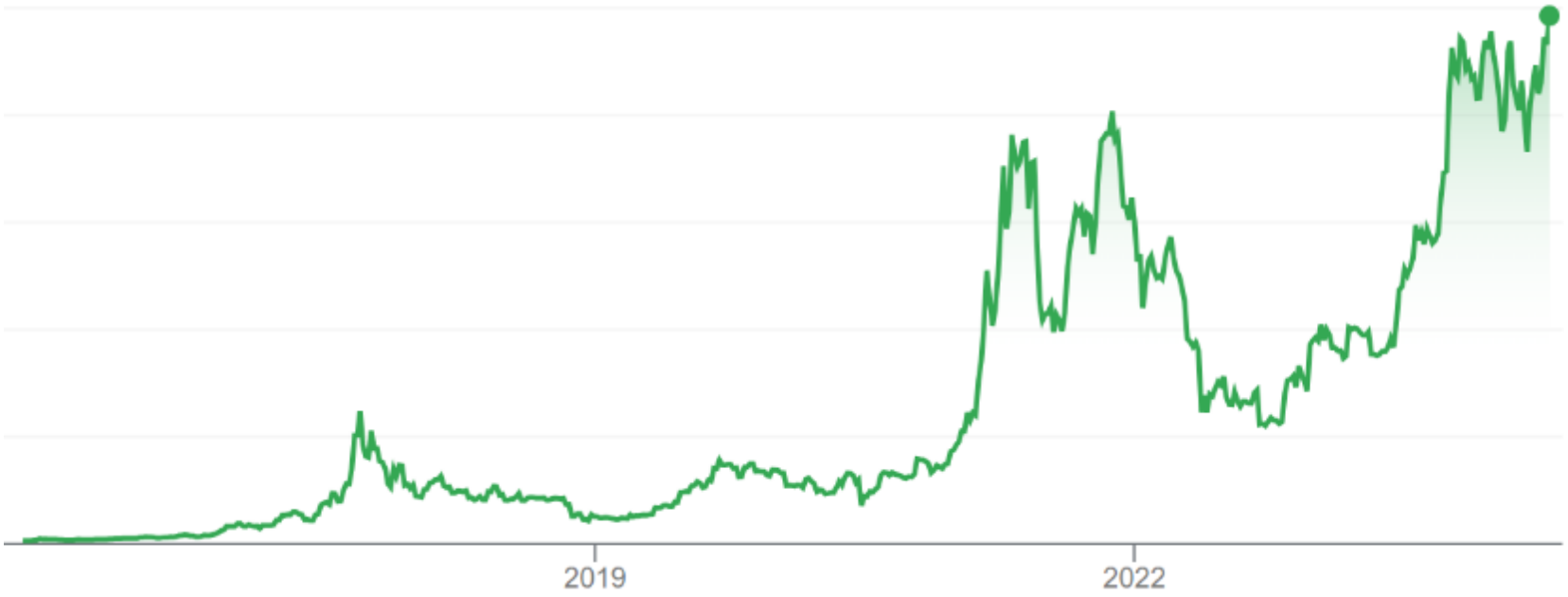
What is Bitcoin?

- Protocol ensures nodes agree on transaction order

  ⇒ Nodes agree about # of BTC owned by each account



Why is Bitcoin interesting?



Why is Bitcoin interesting?

- Bitcoin implements a state machine replication protocol that provides fault tolerance under byzantine failures
- Why not use PBFT?
- Bitcoin is an an open/permissionless system
  - Anyone can join, number of nodes is not known
  - Nodes are untrusted, can spoof their identity, e.g., IP address
    - A node can create lots of ids and subvert agreement (sybil attack)
  - So, quorum consensus is not possible
- Bitcoin consensus scheme is novel and interesting
  - Depends on financial rewards to incentivize honesty
  - Bitcoin's success was a surprise

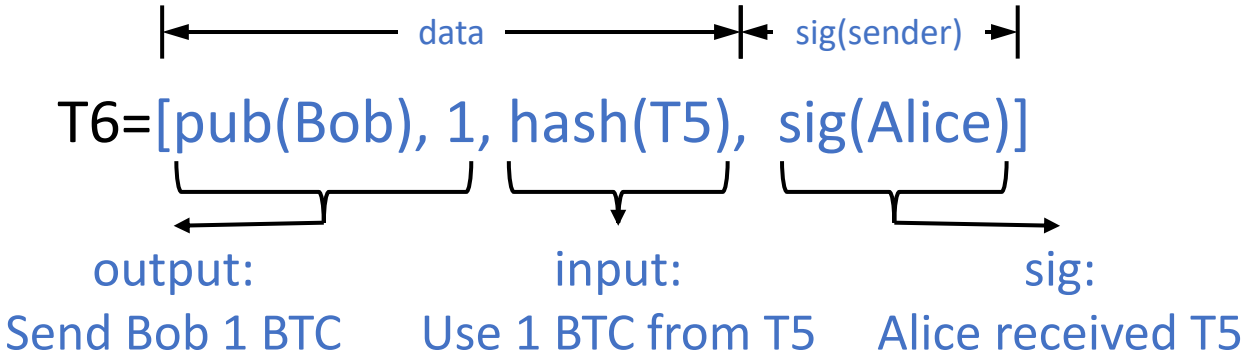
Bitcoin Transactions

Identities, signatures (simplified)

- User identity based on public-key cryptography
  - pub(user): user's identity for receiving bitcoin transactions
  - pri(user): used to sign transactions when sending bitcoins
- Transaction identity based on cryptographic hash
  - hash(T): Hash of transaction uniquely identifies a transaction
  - Bitcoin uses SHA-256 for hash function
- Bitcoin uses digital signatures
  - Sender sends signed Transaction T=[data, sig(sender)]
    - sig(sender)=E(pri(sender), hash(data))
  - Receiver verifies signature on transaction T
    - D(pub(sender), sig(sender)) is hash(data)

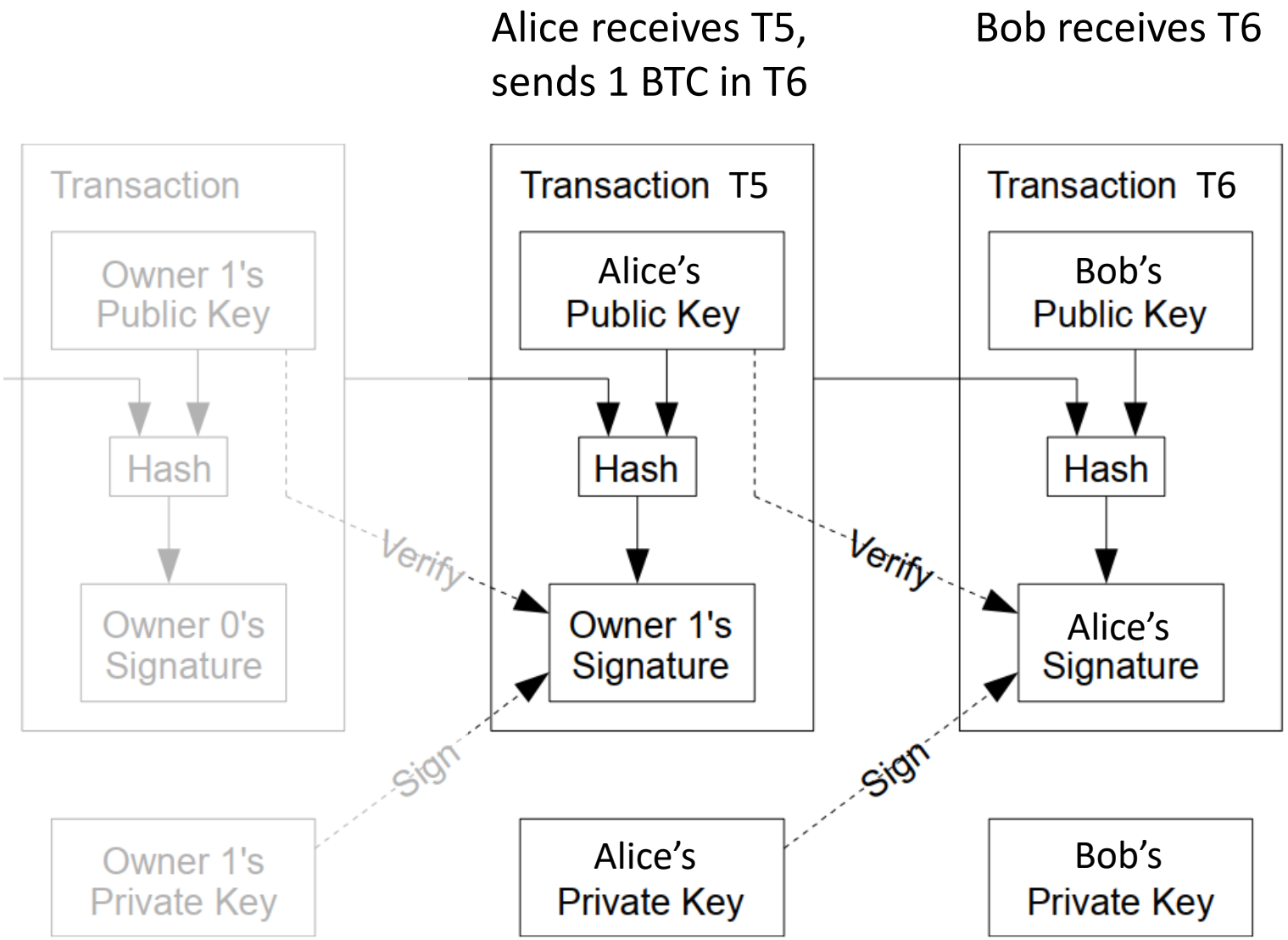
Bitcoin transactions (simplified)

- Say Alice previously received 1 BTC in T5=[pub(Alice), 1, …]
- Alice wants to send 1 BTC to Bob, generates T6



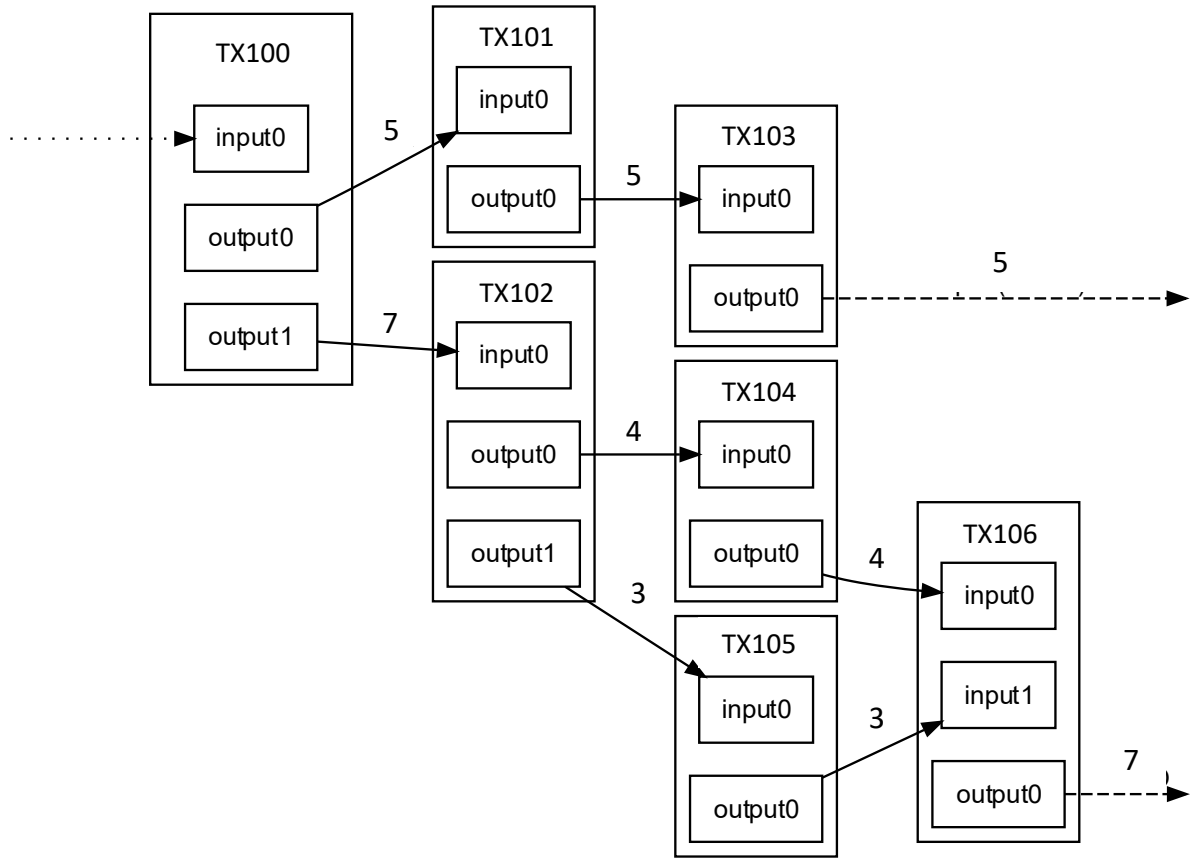
- Alice sends T6 to Bob, Bob validates T6
  - output: pub(Bob) is Bob's public key
  - input: T5 exists in the blockchain
  - sig: verifies T6 is signed by Alice, who received T5
  - If all ok, Bob receives 1 BTC in T6, delivers Porshe to Alice

Bitcoin transactions (simplified)



Bitcoin transaction graph

- Transactions can have multiple inputs and outputs



https://developer.bitcoin.org/

Bitcoin challenges

- Q1: How to prevent forgery (create fake bitcoin)?
- Q2: How to prevent stealing?
- Q3: How to prevent double spending?
- Q4: How to create bitcoins?
- Q5: How to avoid too many bitcoins (inflation)?

Q1: How to prevent forgery?

- When user “owns” bitcoins, we mean
  - User has a public key to which bitcoins were sent
  - User has corresponding private key that authorizes user to send bitcoins previously sent to the user
- Bitcoin only handles and validates transactions, not bitcoins!
  - So, it is not possible to create “fake” bitcoins
- User uses a wallet program to track their bitcoins

Q2: How to prevent stealing?

- In previous example, Bob receives 1 BTC in T6
- Can someone steal and spend the 1BTC from T6?
- Bob's private key is needed to sign any transaction that uses T6 as input, since recipient verifies signature
  - So, attacker needs Bob's private key to steal Bob's BTC
  - Could try to steal it from Bob's phone, etc.
- A real problem, hard to solve
  - Requires keeping the private key safely and not loosing it!

Q3: How to prevent double spending?

- Darth sent T6=[pub(Bob), 1, hash(T5), sig(Darth)] to Bob
- Can Darth create T6'=[pub(Charlie), 1, hash(T5), sig(Darth)] and send this transaction to Charlie?
  - Until now, nothing prevents Darth from double spending!
  - Both transactions will validate correctly
  - This is the core problem that Bitcoin solves
- Why can Darth double spend?
  - Bob and Charlie are not aware of the other's transaction
- Need to ensure users observe all previous transactions
  - A central bank can do so, but
  - Bitcoin is an ad hoc network of nodes

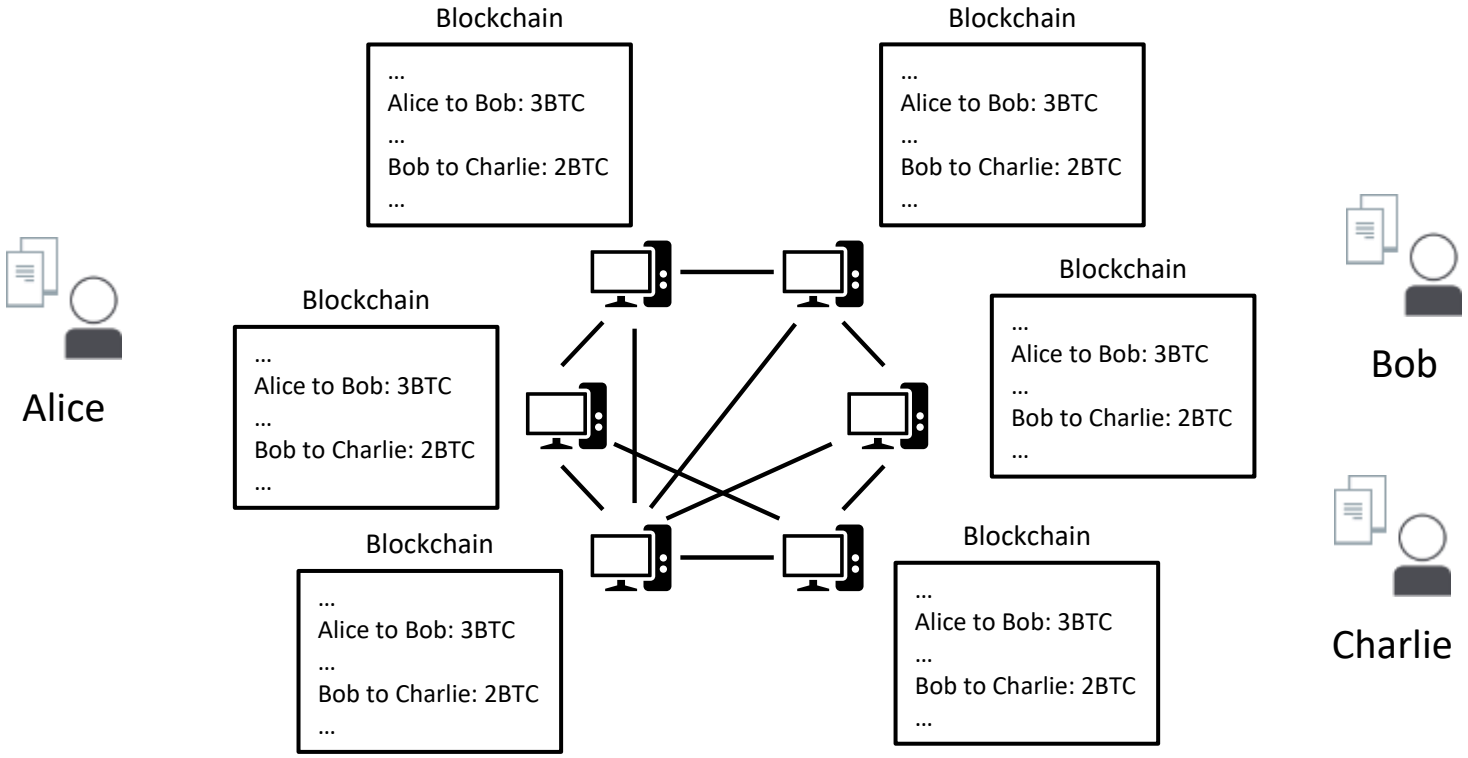
Public ledger

- Let's use a public ledger to log all transactions
- Two requirements for ledger
  - All nodes should observe same order of transactions in ledger
  - No one should be able to modify or hide transaction entries
- Say, T6 is logged before T6' in the ledger
  - Bob accepts T6 transaction since T5 has not been spent
  - Charlie rejects T6' transaction since T5 has already been spent
- In Bitcoin, the public ledger with the properties described above is called a Blockchain

Bitcoin Blockchain

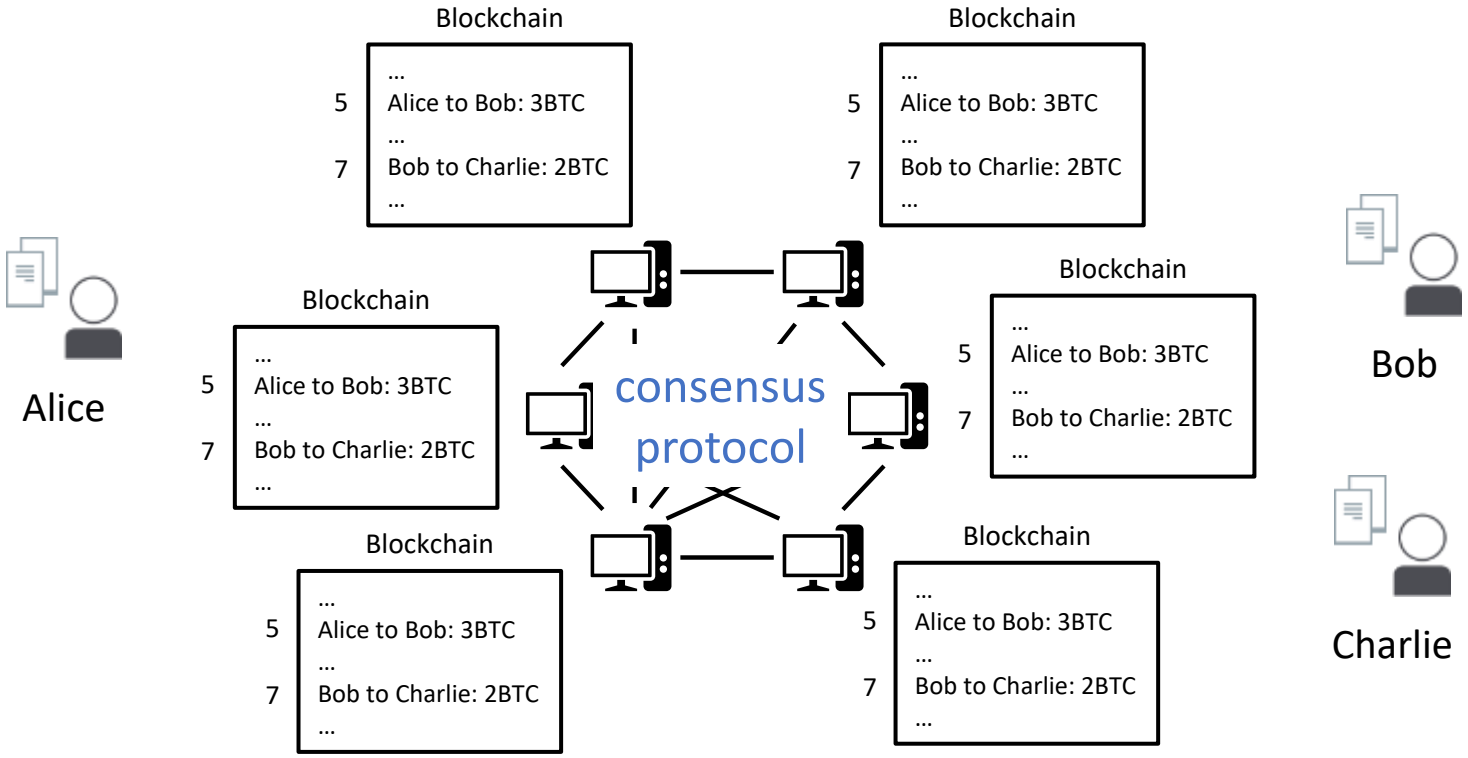
Blockchain

- Bitcoin blockchain contains all transactions in the system
- Each node keeps a replica of the blockchain



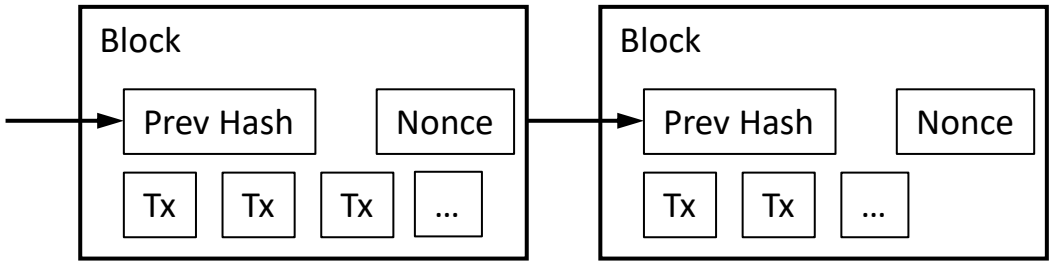
Blockchain goal

- Goal: agreement on transaction order in blockchain to avoid double spending
- Problem: malicious nodes may cause replicas to diverge



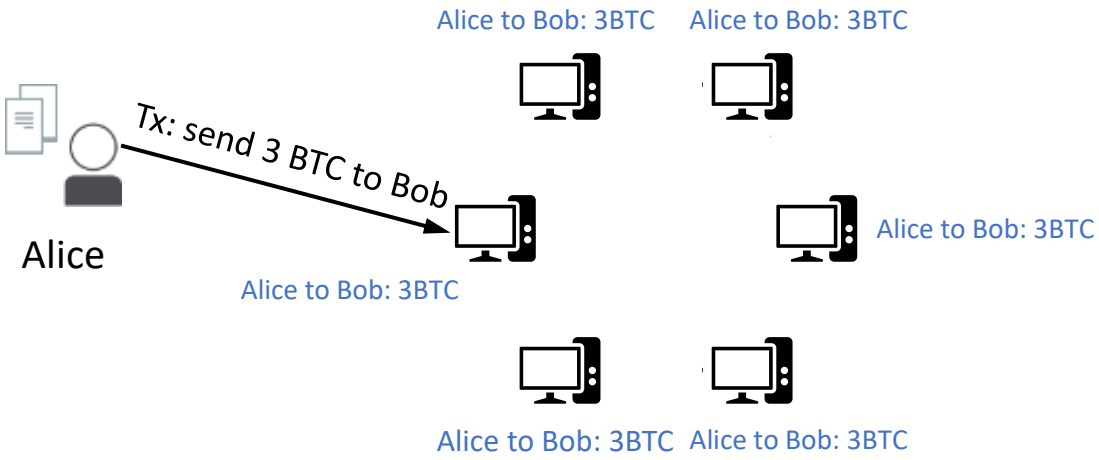
Blockchain structure

- Blockchain contains a sequence (or “chain”) of blocks
- Each block contains
  - A batch of transactions, for efficiency
  - Prev hash, i.e., Hash(previous block), to chain the block
  - Nonce, a random number, assigned when block is linked to chain
  - Current time, wall clock timestamp to prove transaction time
- Hash(block) is block id, identifies full prefix of blockchain



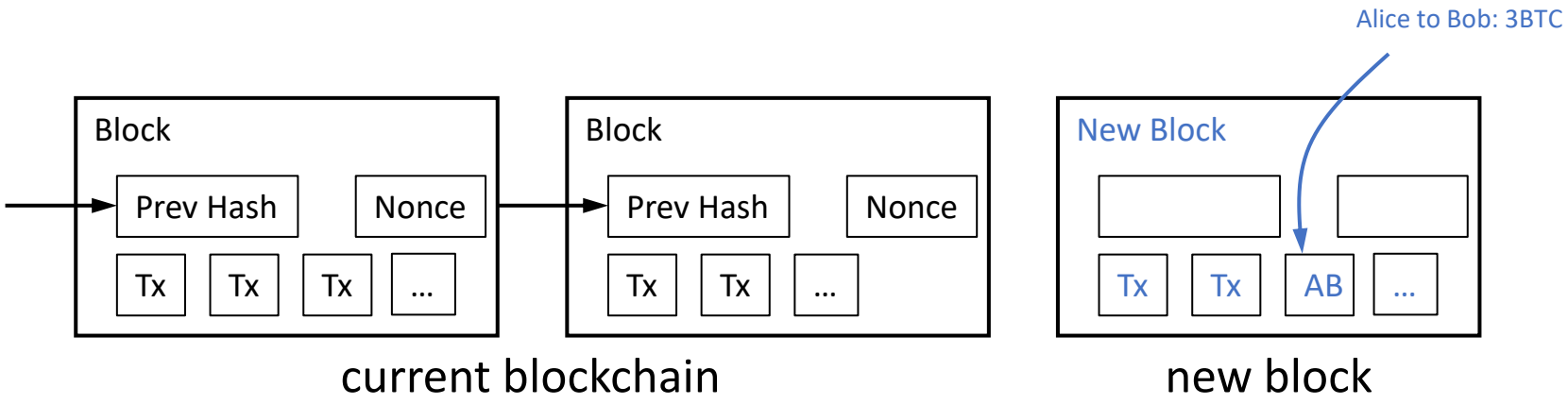
New transaction

- Nodes maintain connections to a few peers
- When Alice sends a transaction to a node, transaction is flooded to network
- Flooding ensures all nodes are aware of transaction



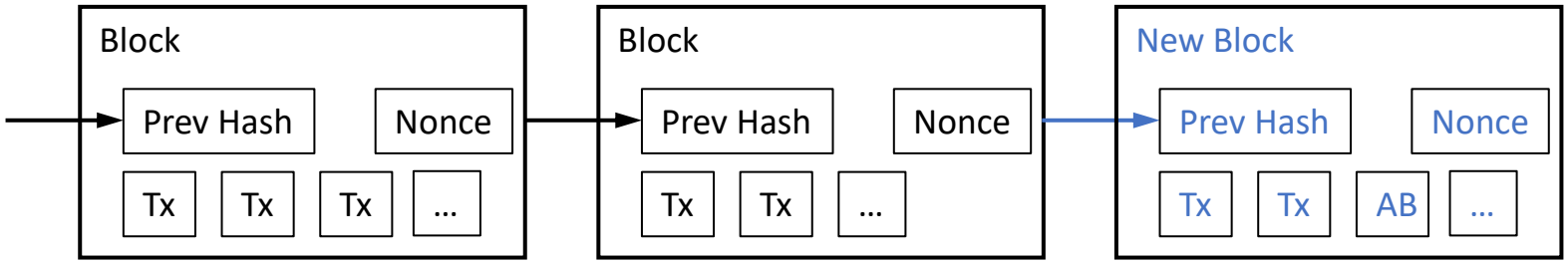
New block

- New transactions are added to a new block



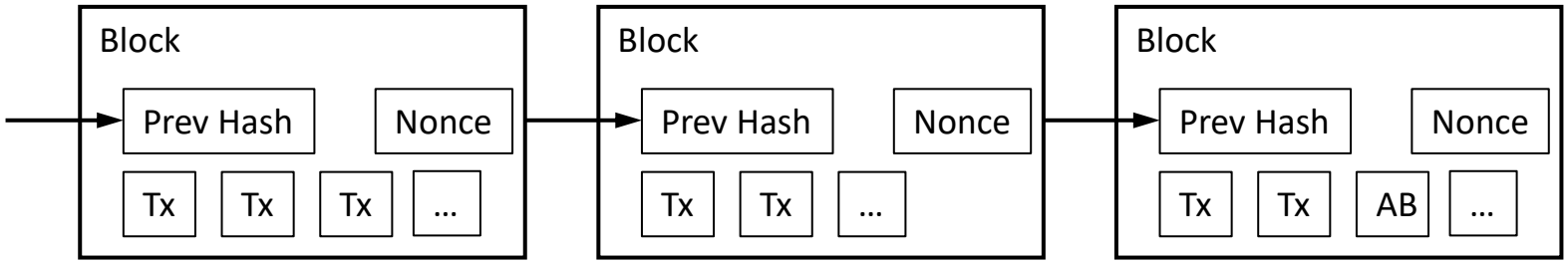
New block

- New transactions are added to a new block
- Periodically, a new block is linked to the blockchain
  - Contains transactions since previous block, immutable
- New block is also flooded to the network



New block

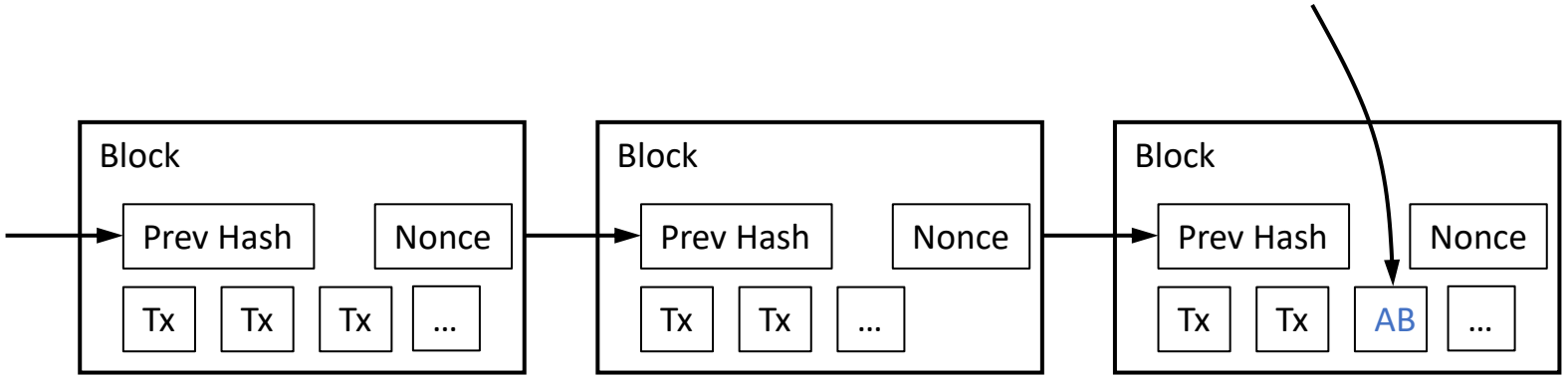
- New transactions are added to a new block
- Periodically, a new block is linked to the blockchain
  - Contains transactions since previous block, immutable
- New block is also flooded to the network
- Flooding ensures all nodes are aware of the new block



current blockchain

New block

- New transactions are added to a new block
- Periodically, a new block is linked to the blockchain
  - Contains transactions since previous block, immutable
- New block is also flooded to the network
- Flooding ensures all nodes are aware of the new block
- Bob trusts Alice's transaction when it is in the blockchain



current blockchain

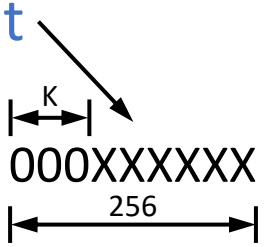
Who creates new block?

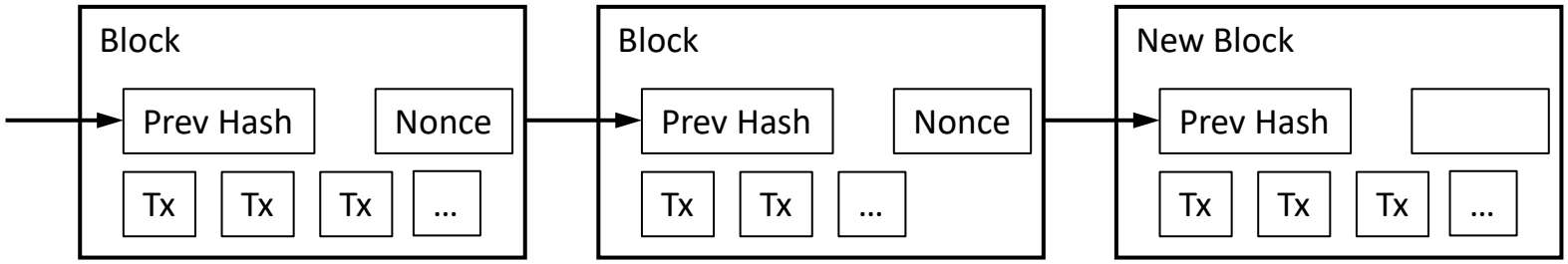
- Any node can create new block!
- But then how can the nodes agree on the new block?
  - Unlike PBFT, there is no long-term primary,
    and quorum consensus is not possible to choose a primary
- We need one node to “win” (i.e., be primary for the block)
  - Winner node creates and links new block to blockchain
  - Tells other nodes about the new block
  - Other nodes must somehow know they “lost”, follow winner
    by linking winner’s new block to their blockchain replica
- Let’s see how Bitcoin achieves this style of consensus by (artificially) making it computationally hard to win

Bitcoin Mining

Bitcoin mining

- To generate a valid block, nodes need to solve a puzzle:
  Find a Nonce value so that hash(new_block) < target
  - Requires a laborious, trial-and-error process
    - Choose random Nonce, check if hash(new_block) < target
    - If target has at least *k* leading zeroes, then # of steps exponential in *k*
    - Similar to flipping a zillion-sided coin until a head comes up, where each flip has a small independent chance of success
  - May take a month of CPU time for a typical desktop!
  - Called mining, i.e., finding a valid block (gold digging analogy)





Winner

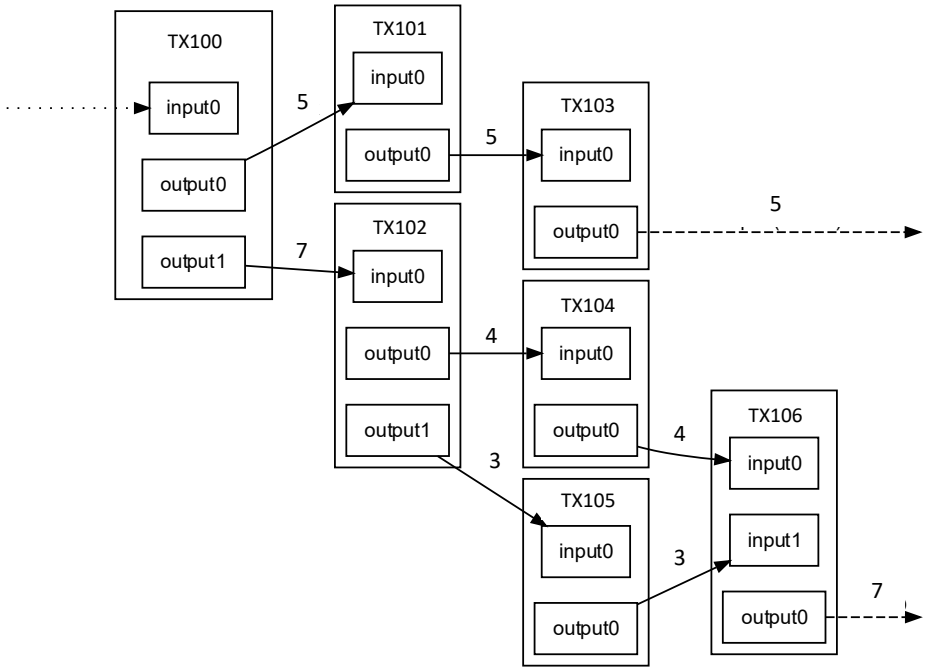
- 1000s of Bitcoin mining nodes (miners) try to win,
  i.e., be first to find a valid new block to link to blockchain
- Mean time to mine a new block is ~10 minutes
  - Bitcoin adjusts target periodically so mean time remains the same
  - We will see why 10 minutes later
- But variance is high => some node wins before others

Block validation

- When a new block is mined, it is flooded to other nodes
- When other nodes receive the new block, they check
  - Hash(block) < target (block is valid)
    - Unlike generating a valid block, validating a valid block is fast
  - Block with previous hash exists (block extends blockchain)
  - All transactions in the block are valid

Transaction validation

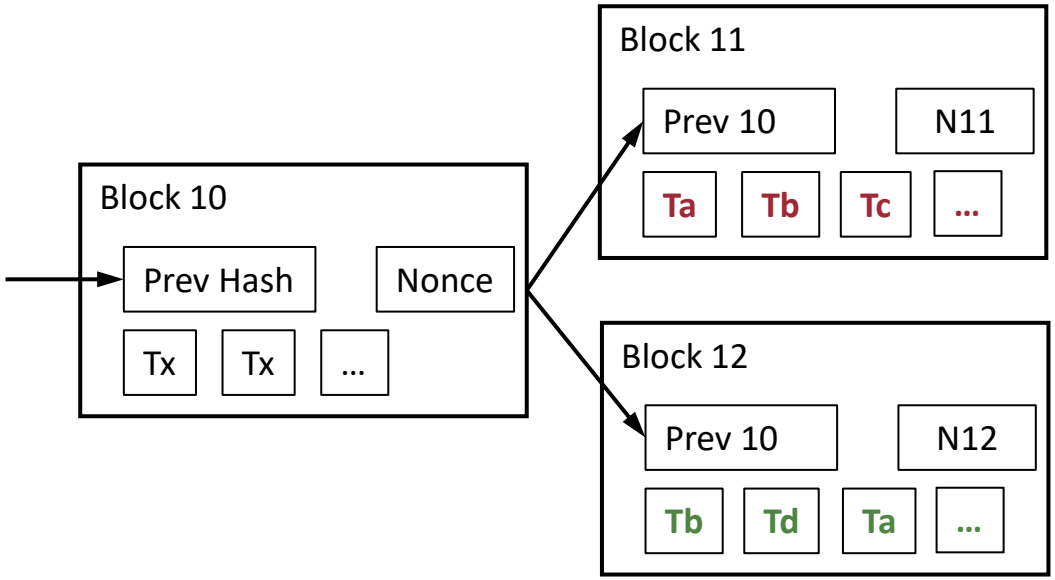
- A transaction is valid when
  - Its input transactions exist in blockchain
  - Its inputs are unspent transaction outputs (UTXOs)
    - What are the UTXO in the figure?
  - Signature is by owner of input transactions
- Validation is performed by
  - Nodes receiving new block
  - Recipient of transaction



https://dveloper.bitcoin.org/

Blockchain fork

- Can there be two “winners”?
- Two nodes can generate valid blocks simultaneously, e.g., a node generates B12 while B11 is being flooded
- Nodes mine successor block from whichever block they hear from first, so chain forks!



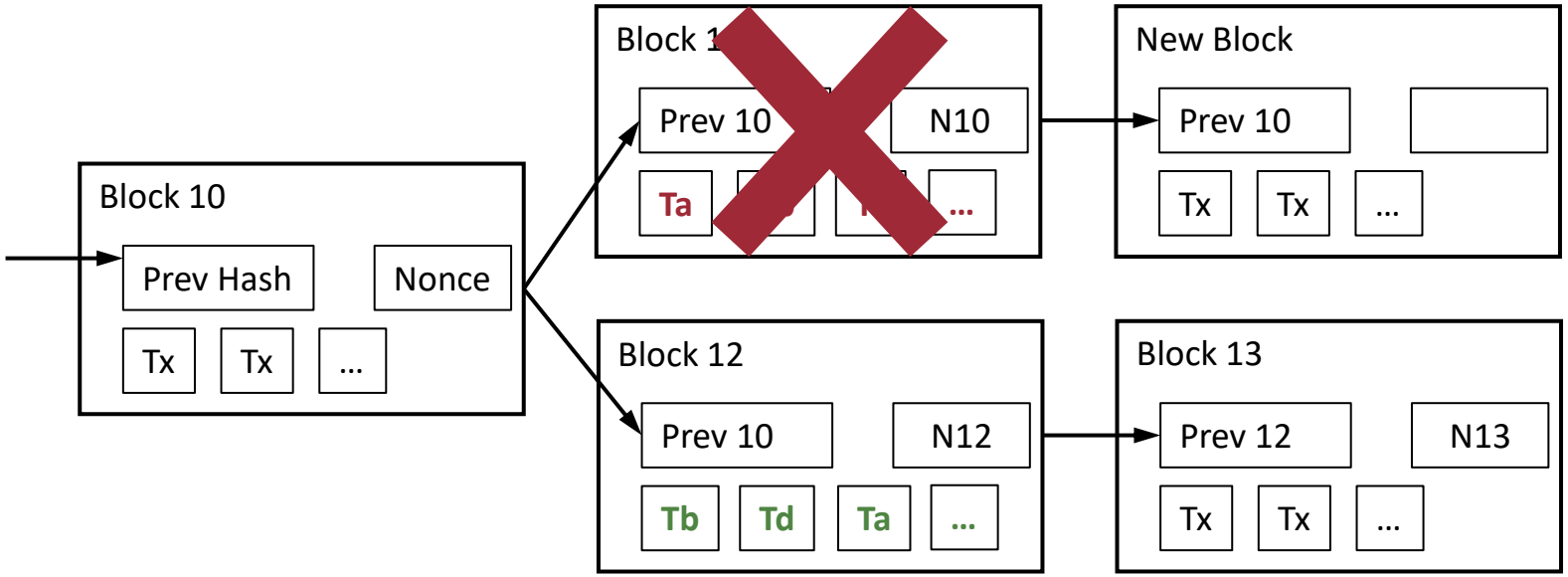
Some nodes store
B10 -> B11 blockchain

No total order, back to double spending problem …

Other nodes store
B10 -> B12 blockchain

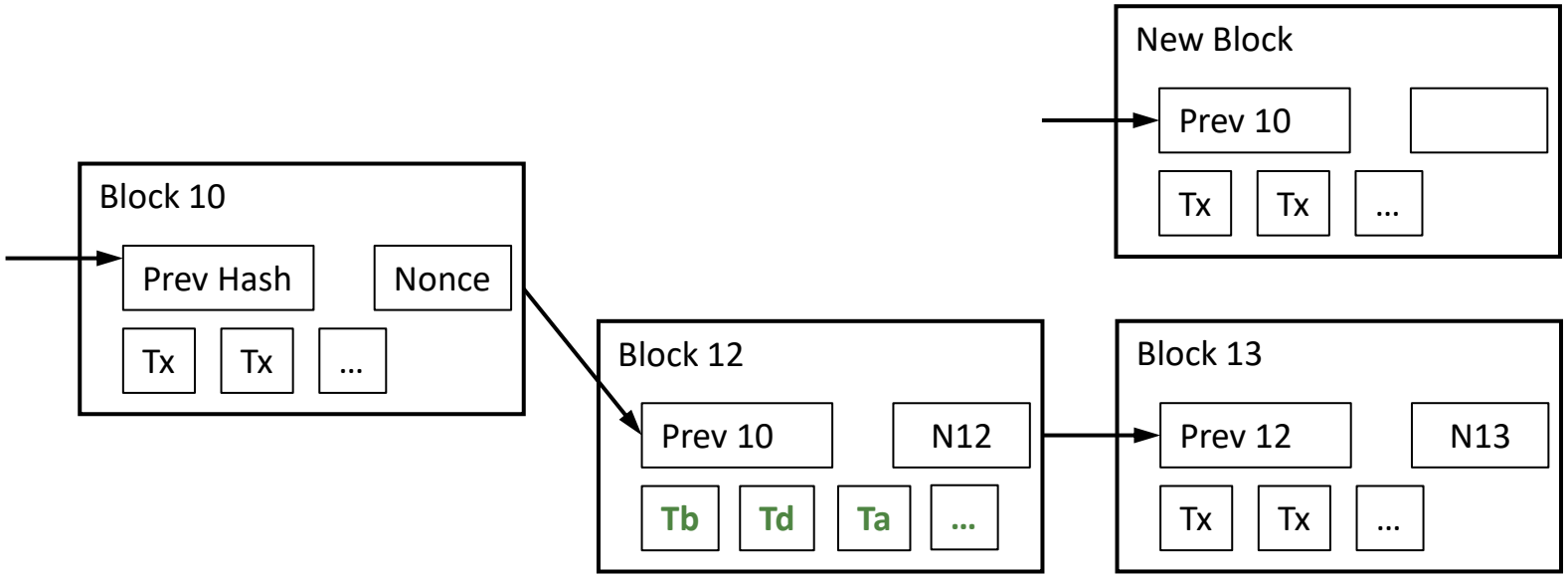
How to resolve blockchain forks?

- Nodes switch to longest fork when they see it
- Say a node is mining for a new block that extends B11
  - If it sees the longer chain B12 -> B13, it will drop B11 chain
    - Blocks in B11 chain are called stale or orphan blocks



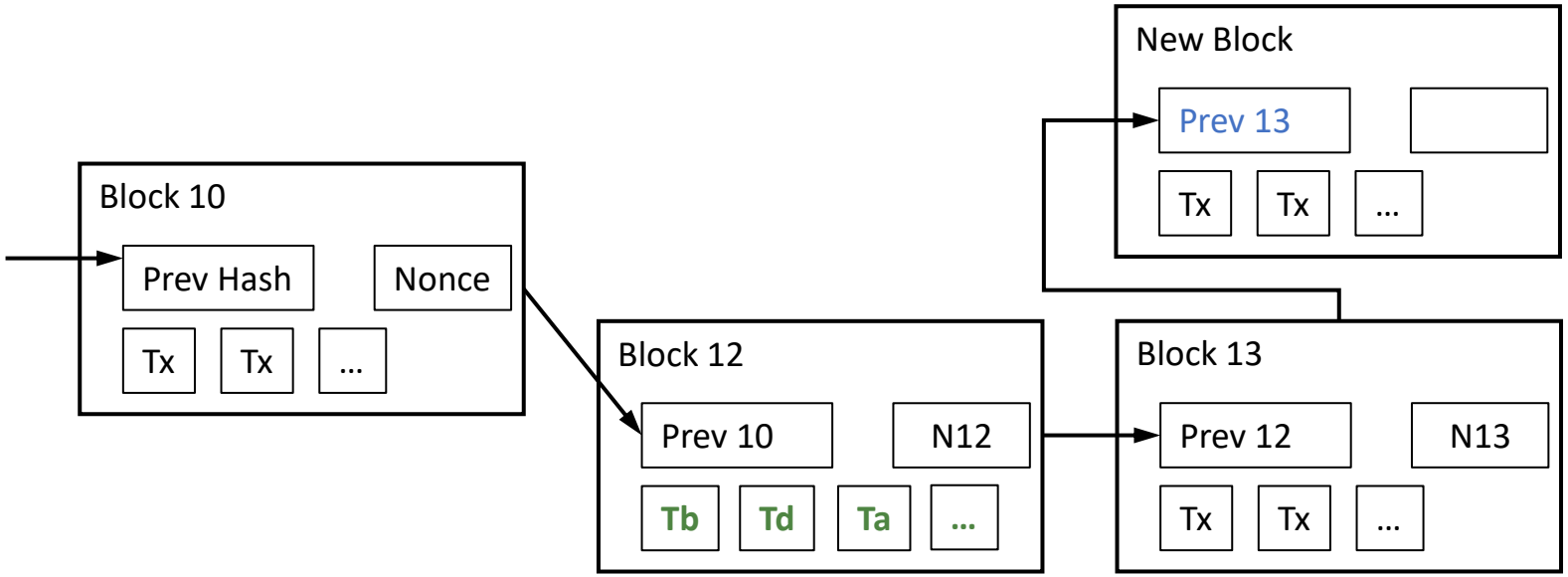
How to resolve blockchain forks?

- Nodes switch to longest fork when they see it
- Say a node is mining for a new block that extends B11
  - If it sees the longer chain B12 -> B13, it will drop B11 chain
    - Blocks in B11 chain are called stale or orphan blocks
- Node will start mining for a new block that extends B13



How to resolve blockchain forks?

- Nodes switch to longest fork when they see it
- Say a node is mining for a new block that extends B11
  - If it sees the longer chain B12 -> B13, it will drop B11 chain
    - Blocks in B11 chain are called stale or orphan blocks
- Node will start mining for a new block that extends B13

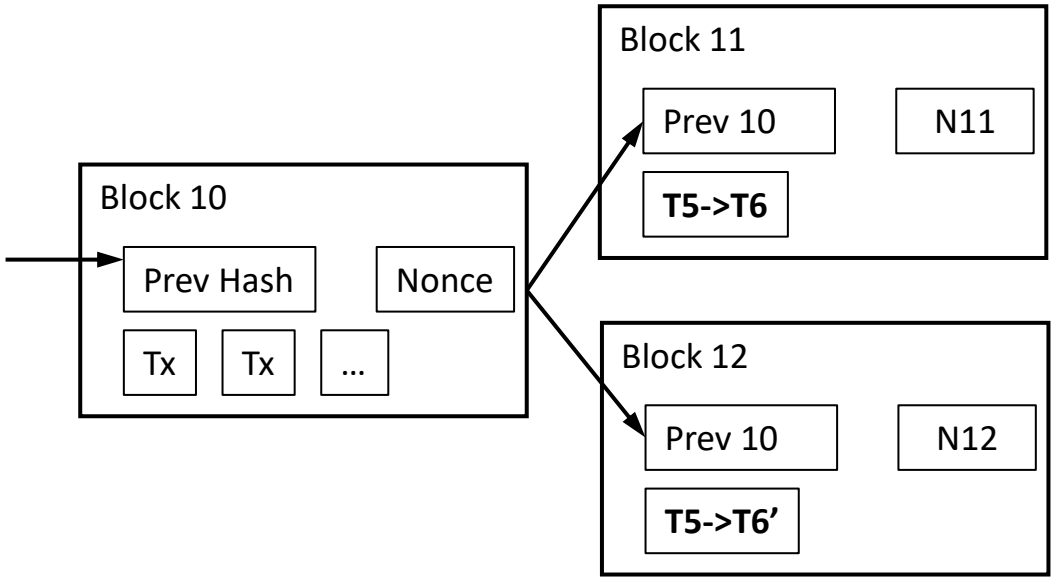


Understanding fork resolution

- If more nodes mine using Chain A than Chain B, then Chain A's successor is likely to be created earlier
  - Chain A becomes longer, Chain B is dropped
- What if same # of nodes work on Chain A and Chain B?
  - A chain is likely to be extended faster due to variance in mining
- What happens to transactions in orphan blocks?
  - Many will be in the longest chain, so okay
  - But if they are only in the orphan block, then transaction will appear and then disappear!
    - Clients may need to retry transactions in orphaned blocks

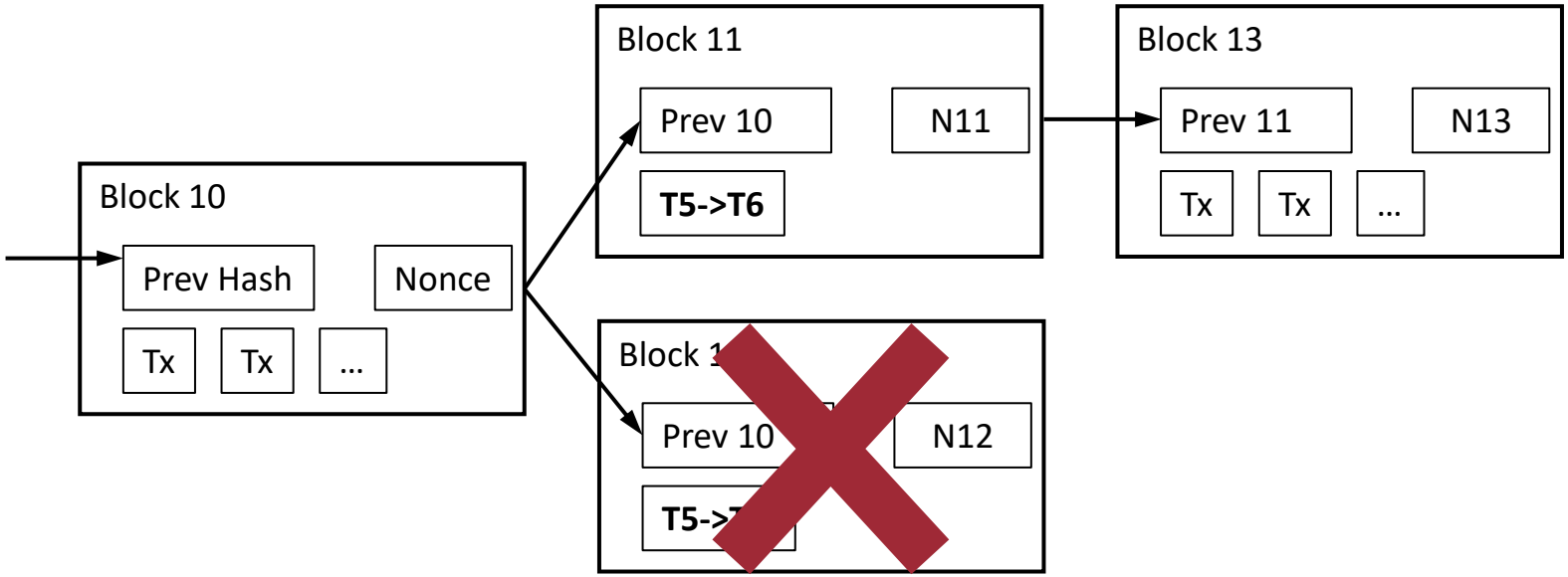
Double spending revisited

- Recall, Darth sent T5->T6 to Bob, T5->T6’ to Charlie
  - If nodes see T6 and then T6’, they reject T6’ since T5 is spent
- But what if Darth sends T5->T6 to some nodes, and T5->T6’ to other nodes and a fork occurs?
  - Both transactions would be accepted, why?



Double spending and fork

- Temporary double spending can occur due to forks!
  - But one of the forks is highly likely to disappear soon
- Bob should wait for the T5->T6 chain to be extended
  - It is highly unlikely that T5->T6' chain will overtake it



How to reduce forks?

- Forked blocks may be created while newly mined block is being flooded
- Choose mean block mining time >> block flooding time
- Mean block mining time is 10 minutes
- Tradeoff between transaction throughput and fork

Immutability of blocks in blockchain

- Why can a node not modify a block in a blockchain?
  - E.g., remove a transaction, modify recipient, etc.
- Node needs to mine a new nonce to make block valid
- New block has different hash, so not part of original chain
- Node would need to fork the chain,
  starting from the modified block, and hope to win
- Potential threat: node colludes with other malicious nodes
  - Grows the forked chain and makes it longest chain
  - But this requires > 50% of total CPU power of the network

Proof-of-work consensus

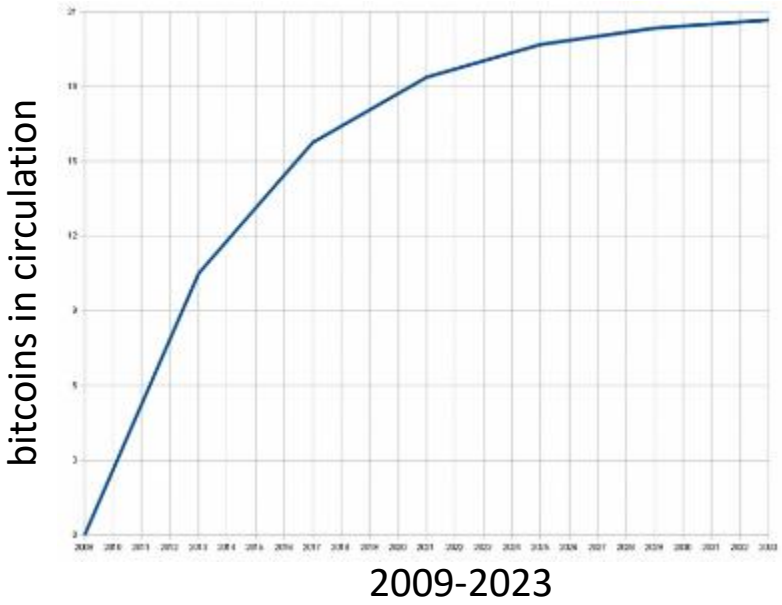
- Bitcoin mining is also called proof-of-work
- Any random node can extend its fork of the blockchain, but nodes with more CPU power will succeed more
  - If most of the CPU power is controlled by honest nodes, their chain will grow fastest, outcompeting other chains
  - If most nodes are honest, they will agree on this longest chain
- So, consensus is based on good nodes having more CPU power than bad nodes
  - No need to know the identity or worry about number of nodes!
- Bitcoin mining and fork resolution ensures safety
  - What about progress, i.e., who does Bitcoin mining?

Q4: How to create bitcoins?

- Mining burns CPU power, energy, and is costly
- Need incentives for mining
- How can we incentivize mining?
- Give miners new bitcoins for each new mined block
  - First transaction in a block (coinbase txn) is a reward to the miner
- Solves the Bitcoin creation problem as well!
  - Bitcoins are only legitimately created when a new block is mined

Q5: How to avoid too many bitcoins?

- There is a total pool of 21M bitcoins to reward miners
- Miners get paid out of that pool
- Miner rewards halve every ~4 years
  - Started at 50 BTC
  - In Apr 2024, 3.125 BTC per block
  - In 2140, no more block reward, i.e., no more bitcoins, miners will get transaction fees
- Other stats (2026):
  - Bitcoins in circulation: 20M
  - New bitcoins per day: 450
  - Bitcoin database: 730GB



Bitcoin security challenges

- Bitcoin is so successful that miners are in an arms race
  - Miners today build dedicated machines with ASICs for mining
    - High barrier to entry
  - Mining pools: miners team up and split the rewards
    - Pools have become large, e.g., top 3 pools control > 50% mining power
  - Both lead to centralization of mining power
    - Goes against the principle of a decentralized currency system
    - Potential threat to the honest node assumptions in the security model
- Privacy through pseudonymity
  - Blockchain is public, contains public keys, not real names
  - But if any public key can be linked to a real name, then the transaction graph can leak information

Bitcoin performance challenges

- Average txns/block is 2-4K, 1 block mined every 10 min
  - Throughput is 7 txns/sec!
    - Several order of magnitude lower than needed worldwide
  - Time to confirm transaction: 10-90 minutes
    - With longer wait, more assurance transaction will not be orphaned
- Mining wastes energy
  - 0.6-0.7% of world's total energy consumption
  - Roughly 25th in energy consumption by country!
- Energy efficiency
  - One Bitcoin transaction can consume 850 kWh ~= month of electricity for the average US household ~= almost 500K VISA transactions!

Bitcoin versus PBFT

	Bitcoin	PBFT
	Permissionless	Permissioned
Approach	Competitive	Cooperative
Basic technique	Proof-of-resource	Byzantine consensus
Trust requirements	Crypto (+ peer …)	Peers (+ crypto …)
Membership	Open	Closed
Energy-efficiency	Often terrible	Excellent
Transaction rate	At best 100s/sec	Many 1000s/sec
Txn latency	Min-hours	Less than a sec

Impact

- Huge interest due to high valuation
- Mother of zillions of cryptocurrencies
- Governments also interested in blockchain technologies
- But heavy price fluctuation, money laundering, environmental impact, collective delusion …

Conclusions

- Bitcoin is a peer-to-peer digital currency system for a permissionless environment
- Uses public key cryptography to prevent forgery of transactions and stealing of bitcoins
- Uses a replicated, public ledger to record transactions
- Uses a novel consensus mechanism based on proof-of-work to prevent double spending

Where to go from here

- Take other systems courses, e.g., systems programming, data networking, security, databases, parallel computing …
  - Look for course FAQ on website
- Join companies, startups, build real distributed systems
- Join graduate program, take advanced systems courses