

# Distributed Algorithms

## Blockchains - Part 1

Alberto Montresor

Università di Trento

2021/12/04

Acknowledgments: Joseph Bonneau, Ed Felten, Arvind Narayanan,  
Christian Cachin

This work is licensed under a Creative Commons  
Attribution-ShareAlike 4.0 International License.



# Table of contents

- 1 Naive coins
  - Goofy coin
  - ScroogeCoin
- 2 Introduction
- 3 Preliminaries
  - Hash function properties
  - Hash-based data structures
  - Signatures
- 4 Bitcoin
  - Ledger
  - Decentralization in Bitcoin
  - BitCoin P2P Network
  - Consensus basics
  - Mining
- 5 History

# GoofyCoin



GoofyCoin

# GoofyCoin

Goofy can create new coins

signed by  $pk_{\text{Goofy}}$

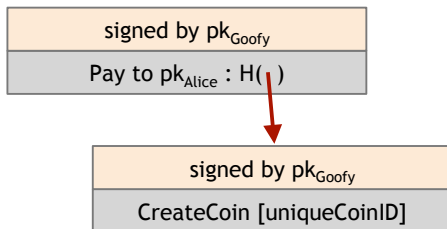
CreateCoin [uniqueCoinID]

New coins belong to me.



# GoofyCoin

A coin's owner can spend it.



Alice owns it now.



# GoofyCoin

The recipient can pass on the coin again.

signed by  $pk_{Alice}$   
Pay to  $pk_{Bob} : H( )$

signed by  $pk_{Goofy}$   
Pay to  $pk_{Alice} : H( )$

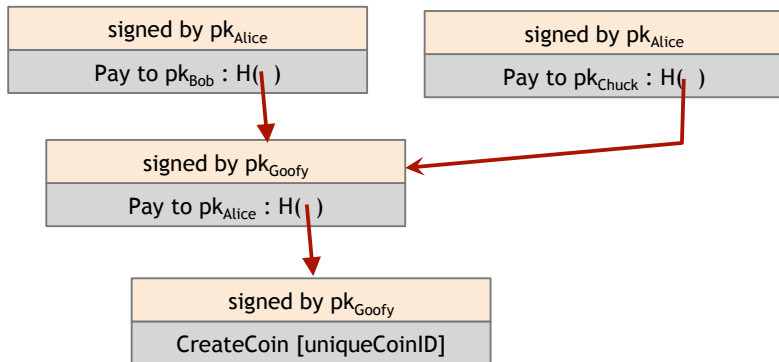
signed by  $pk_{Goofy}$   
CreateCoin [uniqueCoinID]

Bob owns it now.



## GoofyCoin

## double-spending attack



# ScroogeCoin



ScroogeCoin

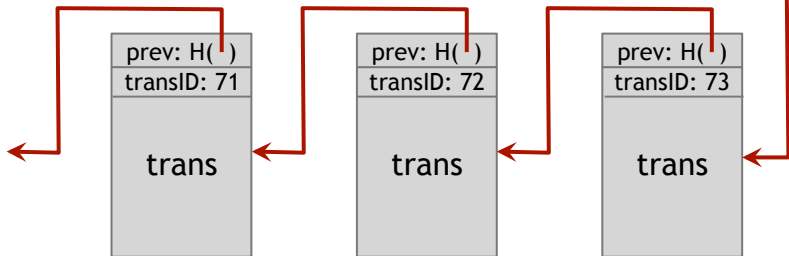


# ScroogeCoin

Scrooge publishes a history of all transactions  
(a block chain, signed by Scrooge)



$H( )$



optimization: put multiple transactions in the same block

# ScroogeCoin

CreateCoins transaction creates new coins

transID: 73		type:CreateCoins	
coins created			
<i>num</i>	<i>value</i>	<i>recipient</i>	
0	3.2	0x...	
1	1.4	0x...	
2	7.1	0x...	

← coinID 73(0)

← coinID 73(1)

← coinID 73(2)

Valid, because I said so.



# ScroogeCoin

PayCoins transaction consumes (and destroys) some coins, and creates new coins of the same total value

transID: 73    type:PayCoins		
consumed coinIDs: 68(1), 42(0), 72(3)		
coins created		
<i>num</i>	<i>value</i>	<i>recipient</i>
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...
signatures		

Valid if:

- consumed coins valid,
- not already consumed,
- total value out = total value in, and
- signed by owners of all consumed coins

# ScroogeCoin

Don't worry, I'm honest.



Crucial question:

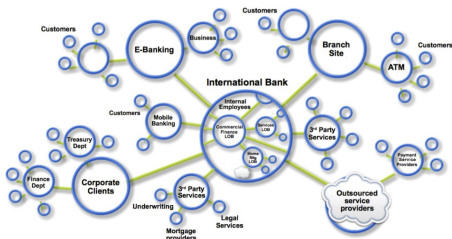
Can we descroogify the currency,  
and operate without any central,  
trusted party?

# Table of contents

- 1 Naive coins
  - Goofy coin
  - ScroogeCoin
- 2 **Introduction**
- 3 Preliminaries
  - Hash function properties
  - Hash-based data structures
  - Signatures
- 4 Bitcoin
  - Ledger
  - Decentralization in Bitcoin
  - BitCoin P2P Network
  - Consensus basics
  - Mining
- 5 History

# Connected markets

- **Networks** connect participants
  - Customers, suppliers, banks, consumers
- Public and private markets organize **trades**
- **Transactions** exchange assets
- Wealth generated by flow of assets and services among participants



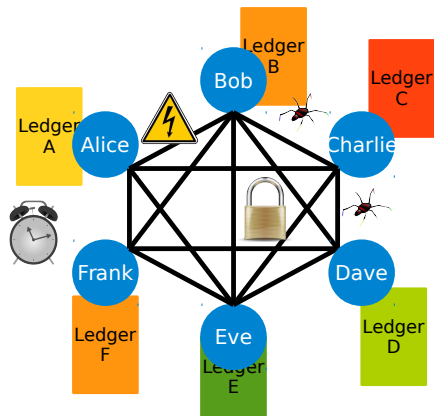
# Ledger

- Ledger records all business activity as transactions
  - Databases
- Ledger records asset transfers between participants
- Problem — (Too) many ledgers
  - Every market has its ledger
  - Every organization has its own ledger

Datum		Entnahme aus Abrechnungs	Einzahlungen, Einzahlungen, Einzahlungen	Bestand der Schuld	Bestand der Guthabens
1992					
	Übersicht:				
12. Dez.	2.500,-	✓	59,-	✓	1.409,50
23.	2.400,-	✓	102,70	✓	1.153,87
01. Jan.	2.200,-	✓	93,57	✓	1.066,94
9.	10.000,-	✓	6,50	✓	1.070,44
19.	4.500,-	✓	46,85	✓	10.572,51
24.	500,-	✓	32,50	✓	10.249,51
28.	1.800,-	✓		47,50	10.122,50
28.	1.500,-	✓		47,50	8.625
01. Jan.	1.518,-	✓	46,50	✓	9.118,5
10.	2.500,-	✓	157,50	✓	11.125
31.	Zinsen gg. per 31.12.92	✓	90,95	✓	9.176,5
1993					
Jan.	37,5	✓			
9.	59,-	✓	9,80	✓	
9.	1.200,-	✓	192,-	✓	1.057,95
21.	505,-	✓			
30.	50,-	✓			
31.	50,-	✓	105,12	✓	1.172,97

# Multiple ledgers

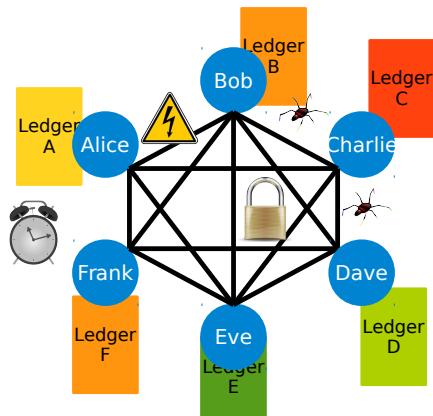
- Every party keeps its own ledger
- Problems, incidents, faults
- Diverging ledgers





# Blockchain provides one single ledger

- One common trusted ledger
  - Today often implemented by a centralized intermediary
- Blockchain creates one single ledger for all parties
- Replicated and produced collaboratively
- Trust in ledger from
  - Cryptographic protection
  - Distributed validation



# Four elements characterize Blockchain

## Replicated ledger

- History of all transactions
- Append-only with immutable past
- Distributed and replicated

## Cryptography

- Ledger integrity
- Transaction authenticity
- Transaction privacy
- Participant identity

## Consensus

- Decentralized protocol
- Shared control tolerating disruption
- Transactions validated

## Business logic

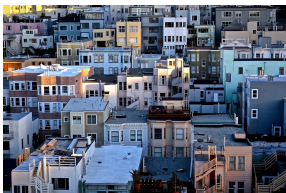
- Logic embedded in ledger
- Executed by transactions
- From simple "coins" to self-enforcing "smart contracts"

# Blockchain simplifies complex transactions



## Financial assets

Faster settlement times  
Increased credit availability  
Transparency & verifiability  
No reconciliation cost



## Property records

Digital but unforgeable  
Fewer disputes  
Transparency & verifiability  
Lower transfer fees



## Logistics

Real-time visibility  
Improved efficiency  
Transparency & verifiability  
Reduced cost

# Blockchain scenario features

## Problem

A given task or problem, but no (central) trusted party available

## Solution

Protocol among multiple nodes, solving a distributed task and reaching consensus

## Key aspects of the distributed task

- Stores data
- Multiple nodes write
- Not all writing nodes are trusted
- Operations are (somewhat) verifiable

# Blockchain scenario features

 Search

[European Commission](#) > [Strategy](#) > [Shaping Europe's digital future](#) >

Shaping Europe's digital future

TEAM RESPONSIBLE

## Digital Innovation and Blockchain (Unit F.3)

# Permissionless vs Permissioned

## Permissionless or Public blockchain

*If writing nodes are not known*

- Bitcoin, Ethereum – Proof-of-Work (PoW)
- Peercoin – Proof-of-Stake (PoS)
- Sawtooth Lake – Proof-of-Elapsed-Time (PoET)

## Permissioned or Consortium blockchain

*If all writing nodes are known*

### CFT

- Hyperledger Fabric/Kafka
- R3 Corda/Raft
- Chai, Quorum, MultiChain

### BFT

- Hyperledger Fabric/PBFT
- Symbiont/BFT-SMaRT
- Tendermint
- Kadena/ScalableBFT (?)

# History

- 31/10/08: Satoshi Nakamoto sends a link to a paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System" to a cryptography mailing list
- Released as an open-source project in January 2009
- Nakamoto worked on the source until mid-2010, when he passed the control to prominent members of the Bitcoin community.
- Several incidents:
  - 2010, exploit, large number of Bitcoins created and later reverted
  - 2013, bug, fork in the chain - two independent chains were operating
  - 2014, Mt.Gox repository filed for bankruptcy, 744KB stolen

# The original paper

## Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto  
satoshin@gmx.com  
www.bitcoin.org

**Abstract.** A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.



## A clever combination of existing techniques

- 2001: SHA-256 finalized
- 1999-present: Byzantine fault tolerance
- 1999-present: P2P networks
- 1998: Wei Dai, B-money
- 1998: Nick Szabo, Bit Gold
- 1997: HashCash
- 1992-1993: Proof-of-work for spam
- 1991: cryptographic timestamp
- 1980: public key crypto algorithm

## Many features in a single protocol

- Anonymity/privacy
- Protection against sybil attacks
- Distributed ledger (notarization)
- Leader election
- Consensus
- Currency, digital payments
- Smart contracts
- An incentive framework

# Table of contents

- 1 Naive coins
  - Goofy coin
  - ScroogeCoin
- 2 Introduction
- 3 Preliminaries**
  - Hash function properties
  - Hash-based data structures
  - Signatures
- 4 Bitcoin
  - Ledger
  - Decentralization in Bitcoin
  - BitCoin P2P Network
  - Consensus basics
  - Mining
- 5 History

# Properties of Hash functions

## Collision-free

- Difficult to find  $x$  and  $y$  such that:

$$x \neq y \text{ and } H(x) = H(y)$$

## Application: Message digest

- If we know that  $H(x) = H(y)$ , it is safe to assume that  $x = y$
- To recognize a message that we saw before, just remember its hash.

# Properties of Hash functions

## Hiding

- Given  $H(x)$ , it is infeasible to find  $x$ .
- Corollary: If  $r$  is chosen from a probability distribution that has high **min-entropy**, then given  $H(r|x)$ , it is infeasible to find  $x$ .
- A random variable  $X$  has **min-entropy**  $k$  if  $\max_x Pr[X = x] = 2^{-k}$ 
  - High min-entropy means that the distribution is “very spread out”
  - No particular value is chosen with more than negligible probability.

## Application: **Commitment**

- Want to commit to a value, reveal it later
  - “seal a value in an envelope” now, and
  - “open the envelope” later.

# Properties of Hash functions

## Commitment API

- $(com, key) \leftarrow \text{commit}(msg)$
- $match \leftarrow \text{verify}(com, key, msg)$

## Sealing the envelope

- $(com, key) \leftarrow \text{commit}(msg)$
- Publish  $com$

## Opening the envelope

- Publish  $key, msg$ ;  $com$  already published
- Anyone can use  $\text{verify}()$  to check validity

# Properties of Hash functions

## Commitment API

- $(com, key) \leftarrow \text{commit}(msg)$
- $match \leftarrow \text{verify}(com, key, msg)$

## Implementation

- $\text{commit}(msg) \equiv (H(msg|key), key)$   
where  $key$  is a random 256-bit value
- $\text{verify}(com, key, msg) \equiv (H(msg|key) = com)$

# Properties of Hash functions

## Commitment API

- $(com, key) \leftarrow \text{commit}(msg)$
- $match \leftarrow \text{verify}(com, key, msg)$

## Security properties

- **Hiding**: Given  $com$ , infeasible to find  $msg$
- **Binding**: Infeasible to find  $msg \neq msg'$  such that  $\text{verify}(\text{commit}(msg), msg') = \mathbf{true}$



# Properties of Hash functions

## Puzzle-friendly

- For every possible output value  $y$ , if  $r$  is chosen from a distribution with high min-entropy, then it is infeasible to find  $x$  such that  $H(r|x) = y$

## Application: Search puzzle

- Given a puzzle  $k$  (from high min-entropy distribution) and a target set  $Y$ , try to find a solution  $x$  such that:

$$H(k|x) \in Y$$

- Puzzle-friendly property implies that no solving strategy is much better than trying random values of  $x$

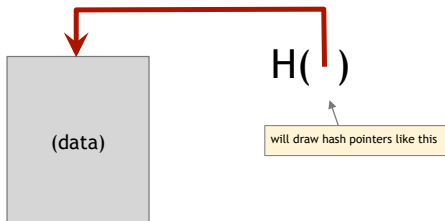
# Hash pointer

## Hash pointer

- A pointer to where some info is stored, and
- a (cryptographic) hash of the info

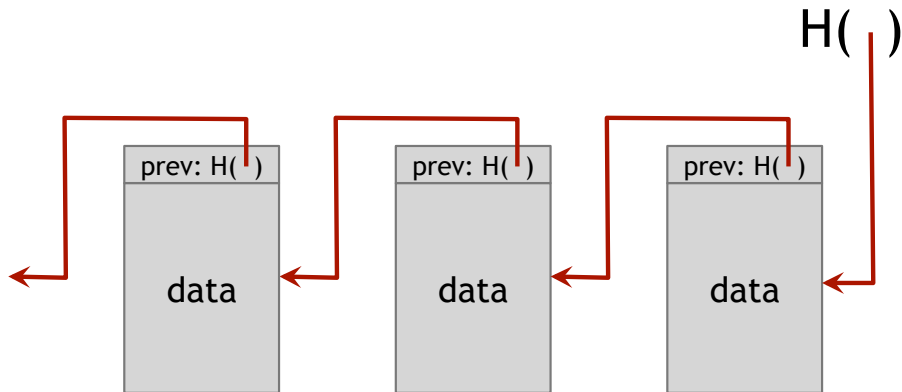
If we have a hash pointer, we can:

- ask to get the info back
- verify that it hasn't changed



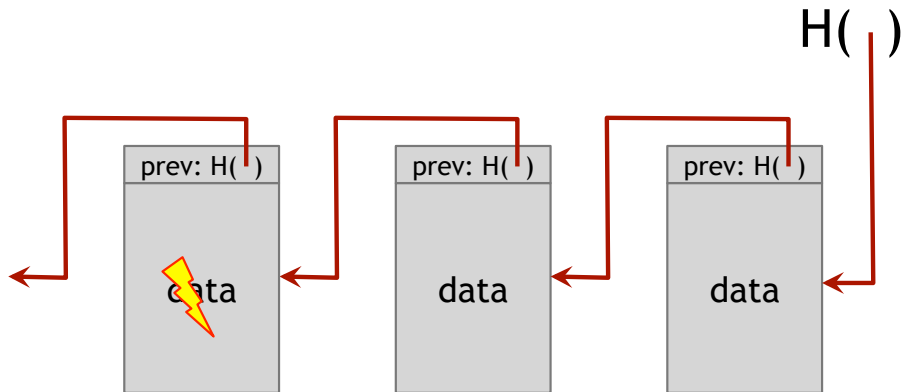
# Blockchain

Linked list with hash pointers (**Blockchain**)



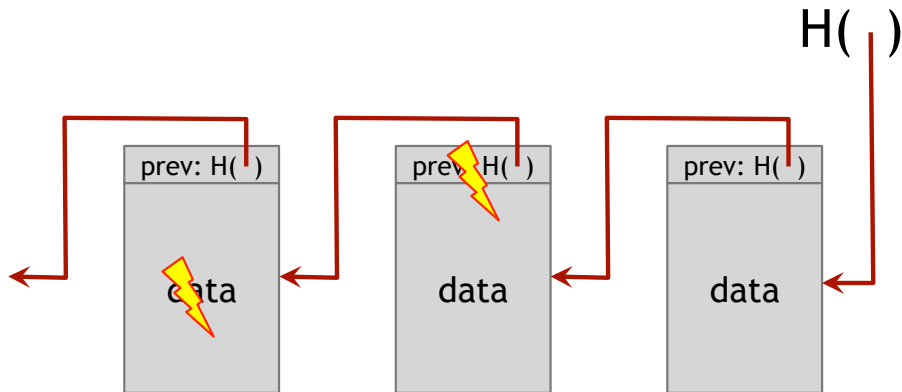
# Blockchain

Linked list with hash pointers (**Blockchain**)



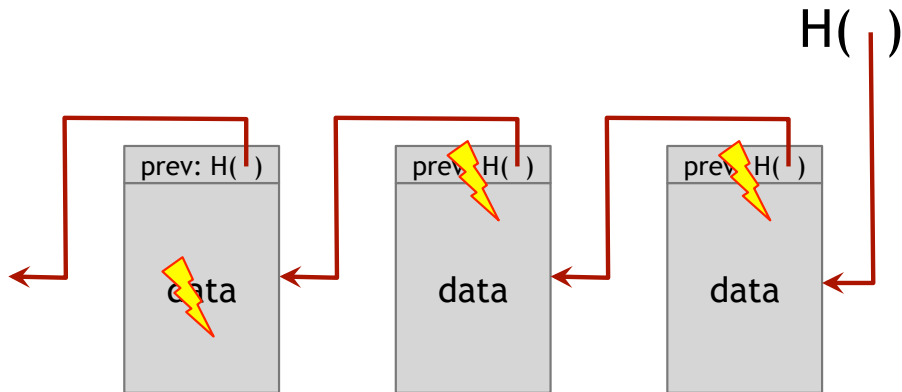
# Blockchain

Linked list with hash pointers (**Blockchain**)



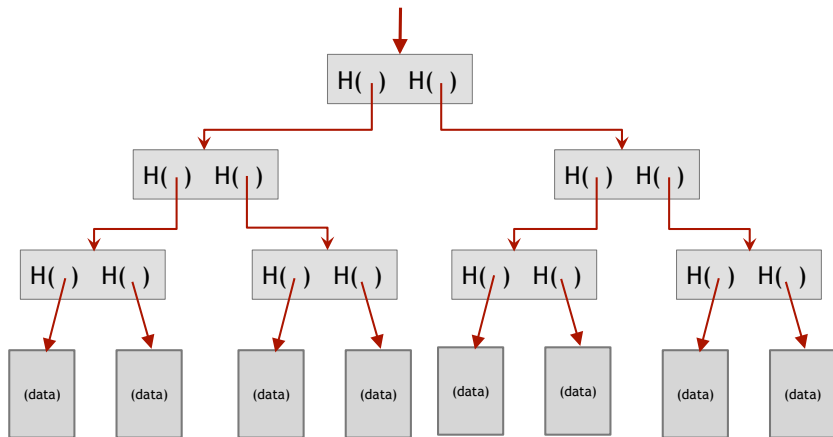
# Blockchain

Linked list with hash pointers (**Blockchain**)



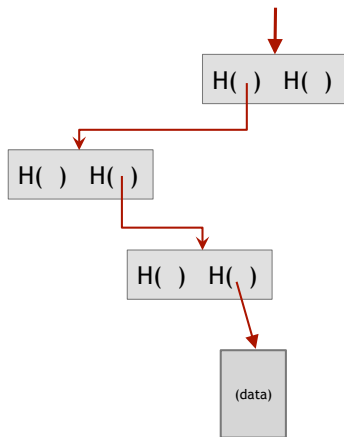
# Merkle tree

Binary tree with hash pointers (Merkle tree)



# Merkle tree

Binary tree with hash pointers (Merkle tree)



show  $O(\log n)$  items



# Merkle tree

## 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

# Public/private signatures

## Objectives

- Only you can sign, but anyone can verify
- Signature is tied to a particular document  
Cannot be cut-and-pasted to another doc

## API for digital signatures

- $(sk, pk) \leftarrow \text{generateKeys}(keysize)$ 
  - $sk$ : secret signing key
  - $pk$ : public verification key
- $sig \leftarrow \text{sign}(sk, message)$
- $isValid \leftarrow \text{verify}(pk, message, sig)$

# Public/private signatures

## Requirements

- Valid signatures verify:
  - $\text{verify}(pk, message, \text{sign}(sk, message)) = \mathbf{true}$
- An adversary who:
  - knows  $pk$
  - gets to see signatures on messages of his choicecan't produce a verifiable signature on another message

## Bitcoin

- Uses ECDSA (Elliptic Curve Digital Signature Algorithm)

# Public/private signatures

Public keys  $\equiv$  identity

- If you see  $sig$  such that  $verify(pk, msg, sig) = \mathbf{true}$ , think of it as “ $pk$  says [ $msg$ ]”

Decentralized identity management

- Anybody can make a new identity at any time
- No central point of coordination
- These identities are called “addresses” in Bitcoin

Privacy

- Addresses not directly connected to real-world identity.
- But observer can link together an address’s activity over time, make inferences

# Table of contents

- 1 Naive coins
  - Goofy coin
  - ScroogeCoin
- 2 Introduction
- 3 Preliminaries
  - Hash function properties
  - Hash-based data structures
  - Signatures
- 4 Bitcoin**
  - Ledger
  - Decentralization in Bitcoin
  - BitCoin P2P Network
  - Consensus basics
  - Mining
- 5 History

# Definitions

## Transaction

A transaction is a transfer of Bitcoin value that is broadcast to the network and collected into blocks.

- A transaction typically references previous transaction outputs as new transaction inputs and dedicates all input Bitcoin values to new outputs.
- Transactions are not encrypted, so it is possible to browse and view every transaction ever collected into a block.

# Blocks

Transaction data is bundled together and recorded in files called blocks

- Single unit of work for miners
- Limit length of hash-chain of blocks
- Faster to verify history

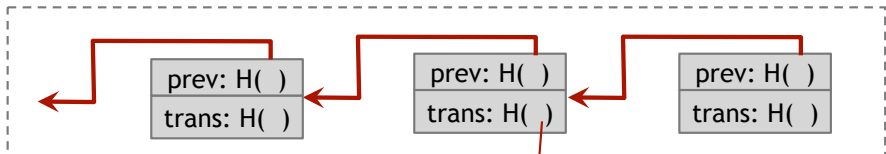
block header

transaction data

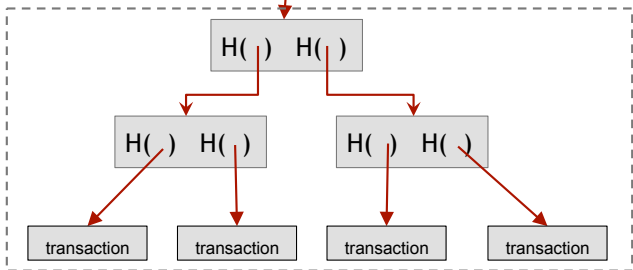
```
{
  "hash": "0000000000000001aad2...",
  "ver": 2,
  "prev_block": "0000000000000003043...",
  "time": 1391279636,
  "bits": 419558700,
  "nonce": 459459841,
  "mrkl_root": "89776...",
  "n_tx": 354,
  "size": 181520,
  "tx": [
    ...
  ],
  "mrkl_tree": [
    "6bd5eb25...",
    ...
    "89776cdb..."
  ]
}
```

# Structure of the Blockchain

Hash chain of blocks

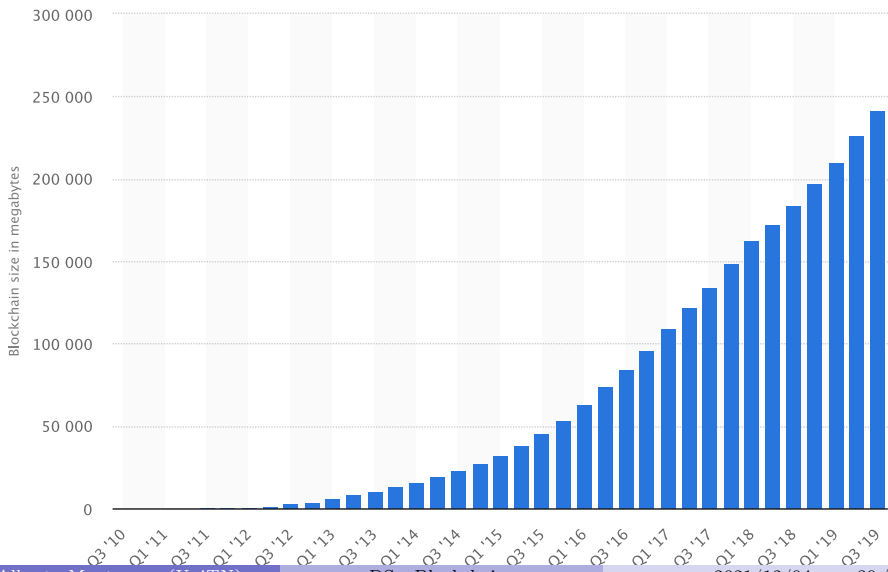


Hash tree (Merkle tree) of transactions in each block

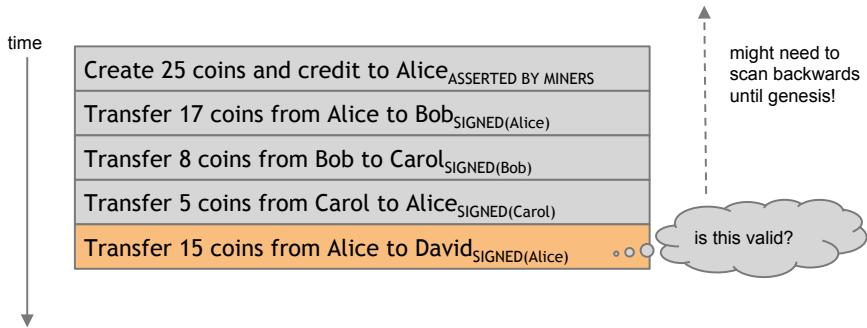




# Blockchain size

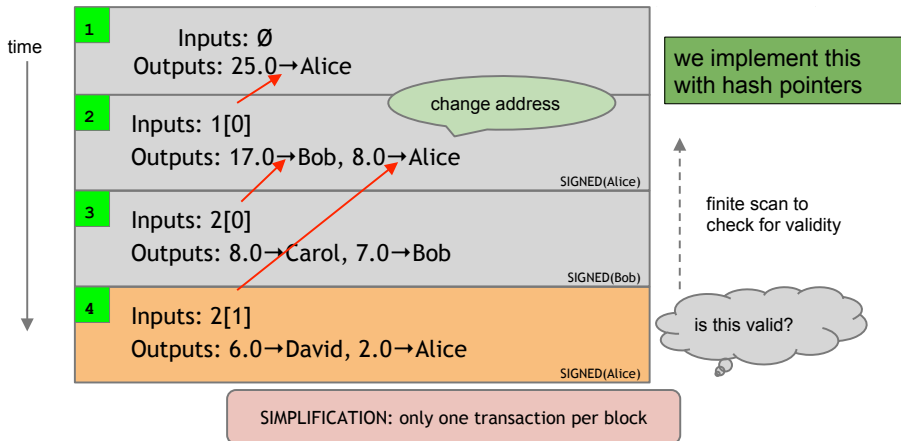


# An account-based ledger (not Bitcoin)

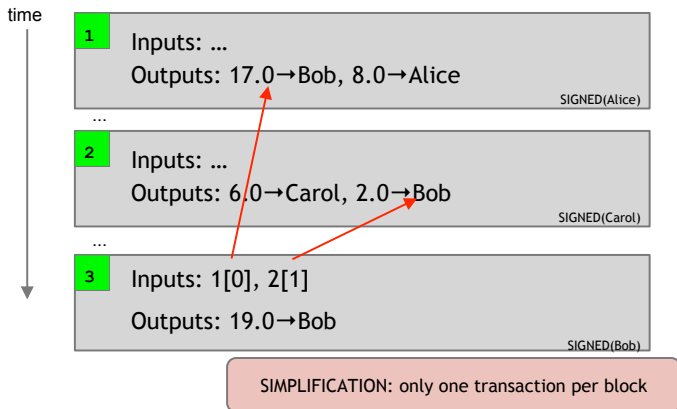


SIMPLIFICATION: only one transaction per block

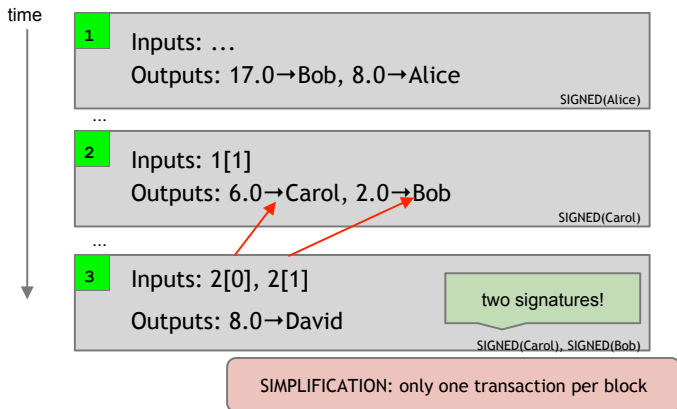
# A transaction-based ledger (Bitcoin)



# Merging value



# Joint payments



# Aspects of decentralization in Bitcoin

## Technological aspects of decentralization in Bitcoin

- Who maintains the ledger?
- Who has authority over which transactions are valid?
- Who creates new Bitcoins?

## Socio-economical aspects of decentralization in Bitcoin

- Who determines how the rules of the system change?
- Who maintains the software
- How do Bitcoins acquire exchange value?

## Beyond the protocol

- exchanges, wallet software, service providers...

# Decentralization in Bitcoin

## Peer-to-peer network

- Open to anyone, low barrier to entry

## Consensus

- Consensus is reached by waiting long enough to observe confirmations from most of the network

## Mining

- Open to anyone, but inevitable concentration of power often seen as undesirable

# Bitcoin is a peer-to-peer system

When Alice wants to pay Bob,  
she broadcasts the transaction to all Bitcoin nodes



signed by Alice  
Pay to  $pk_{\text{Bob}} : H( )$



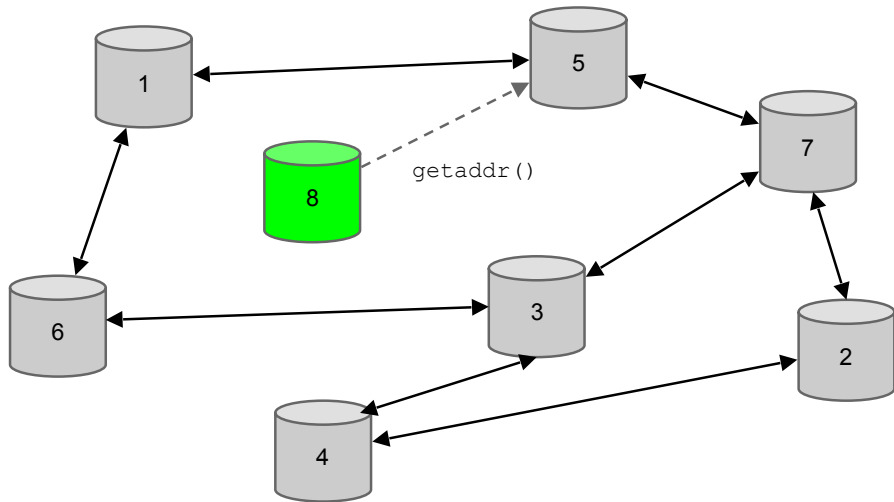
Note: Bob's computer is not in the picture



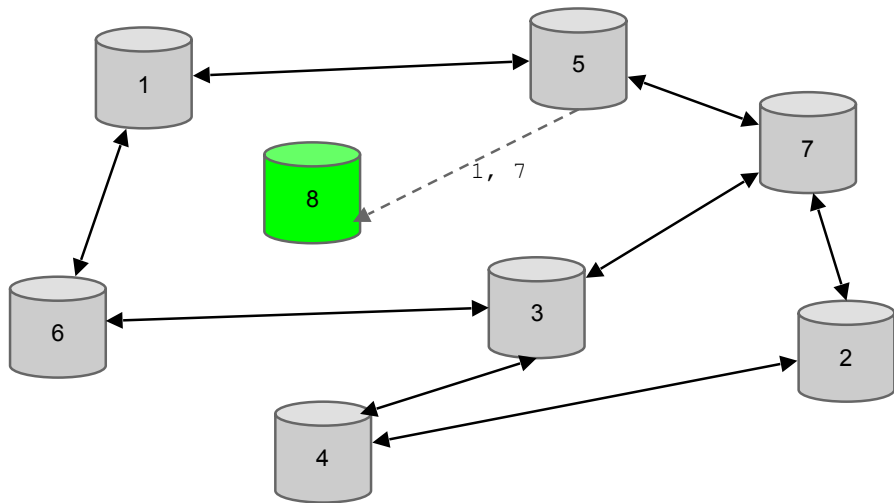
# Bitcoin P2P network

- Ad-hoc protocol (runs on TCP port 8333)
- Ad-hoc network with random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hr

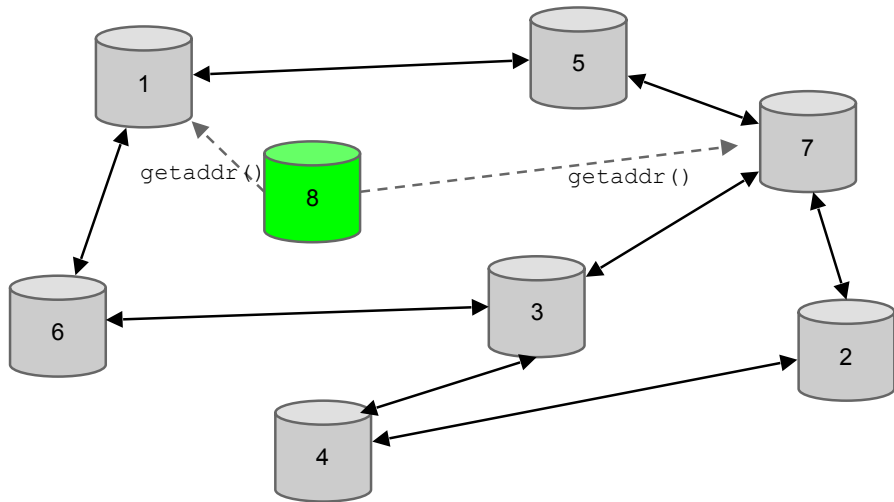
# Bitcoin P2P network



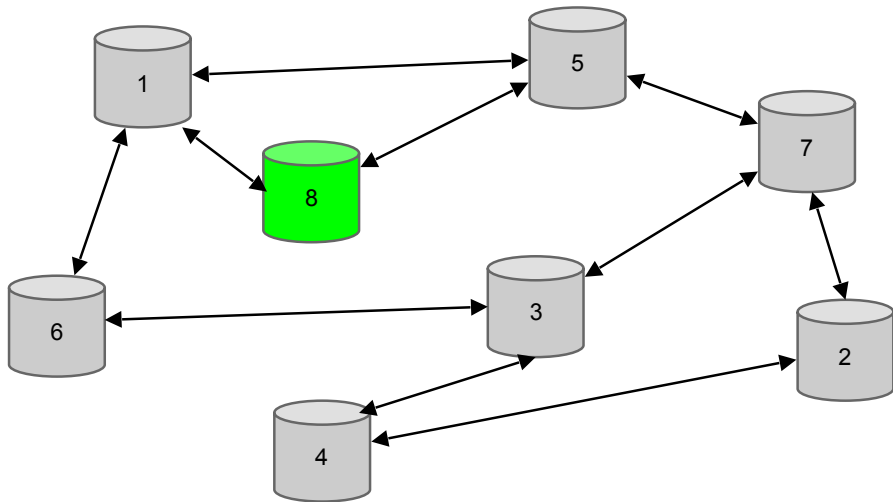
# Bitcoin P2P network



# Bitcoin P2P network



# Bitcoin P2P network



# Network characteristics

## How big is the network?

- Impossible to measure exactly
- Estimates-up to 1M IP addresses/month
- Only about 5-10k “full nodes”

## Full nodes

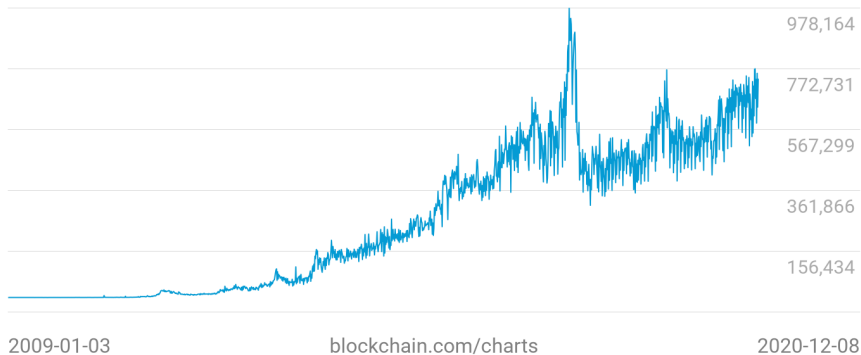
- Permanently connected
- Store entire block chain
- Forward every tx/block
- Tracking UTXO (Unspent Transaction Output)
  - 2018  $\approx$  51 M UTXOs
  - Can easily fit into RAM

## Thin clients

- Connected on-demand
- Store block headers only
- Request transactions as needed, to verify payment
- Trust full nodes
- 1000x cost savings!

# Network size

Number Of Unique Addresses Used  
**737,193**



## Software diversity

- About 90% of nodes run “Core Bitcoin” (C++)
  - Some are out of date versions
- Other implementations running successfully
  - BitcoinJ (Java)
  - Libbitcoin (C++)
  - btcd (Go)
- “Original Satoshi client”



# Bitcoin consensus

## Bitcoin's key challenge

- Key technical challenge of decentralized e-cash: **distributed consensus**
- or, how to decentralize ScroogeCoin

## Theory vs practice

- Remember the theoretical impossibility results
- Bitcoin consensus works better in practice than in theory
- Theory is still catching up

# Validation

## Ordinary Byzantine Consensus

- **Weak Validity:** Suppose all nodes are correct: if all propose  $v$ , then a node may only decide  $v$ ; if a node decides  $v$ , then  $v$  was proposed by some node
- **Agreement:** No two correct nodes decide differently
- **Termination:** Every correct node eventually decides.

## Problems

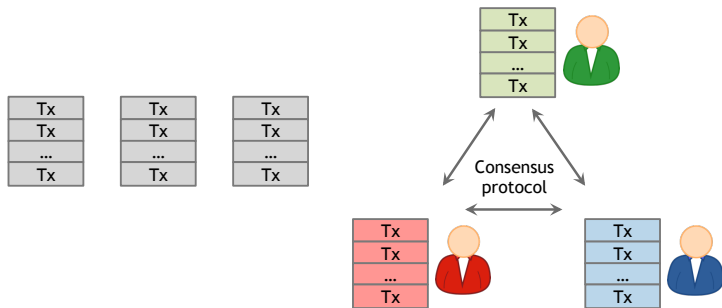
Standard validity notions do not connect to the application! Need validity anchored at external predicate.

- **External validity:** Given predicate  $P$ , known to every node, if a correct node decides  $v$ , then  $P(v)$ ; additionally,  $v$  was proposed by some node.

# How consensus could work in BitCoin

At any given time:

- All nodes have a sequence of blocks of transactions they've reached consensus on
- Each node has a set of outstanding transactions it has heard about



# Consensus without identity

## Why identity?

- Pragmatic: some protocols need node IDs
- Security: assume less than 50% are malicious

## Why don't Bitcoin nodes have identities?

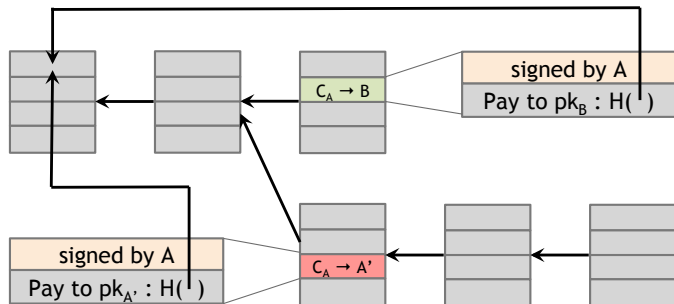
- Identity is hard in a P2P system — Sybil attack
- Pseudo-anonymity is a goal of Bitcoin

## Key idea: implicit consensus

- New transactions are sent to all nodes
- Each node collects new transactions into a block
- In each round a random node gets to broadcast its block
- This node proposes the next block in the chain
- Other nodes implicitly accept/reject this block
  - by either extending it
  - or ignoring it and extending chain from earlier block
- Every block contains hash of the block it extends

## What can a malicious node do?

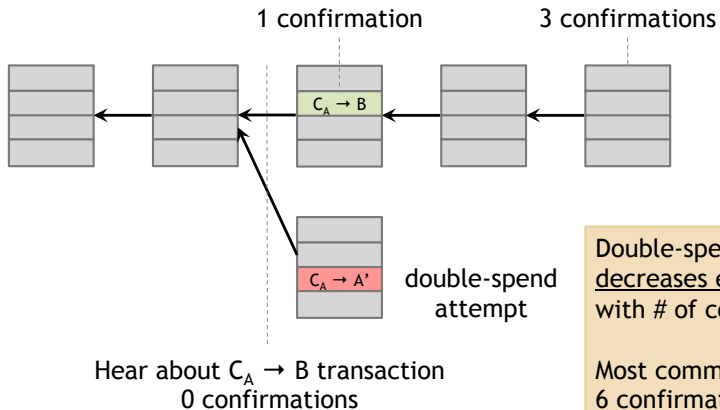
Honest nodes will extend the longest valid branch



Double-spending  
attack

# What can a malicious node do?

From Bob the merchant's point of view



# Recap

- 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



## Some things Bitcoin does differently

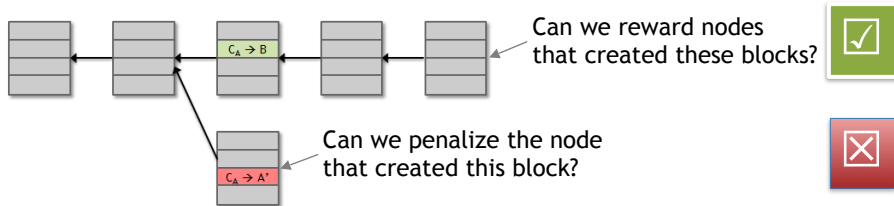
### Introduces incentives

- Possible only because it's a currency!

### Embraces randomness

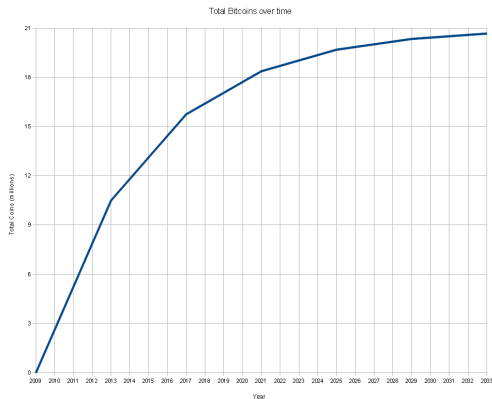
- Does away with the notion of a specific end-point
- Consensus happens over long time scales — about 1 hour

## Incentives for behaving honestly – Block reward



- Creator of block gets to
  - include special coin-creation transaction in the block
  - choose recipient address of this transaction
- Value is fixed: currently 12.5 BTC, halves every 4 years (next: 2024)
- Block creator gets to “collect” the reward only if the block ends up on long-term consensus branch!

# 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

# Selecting the random node

## Problems

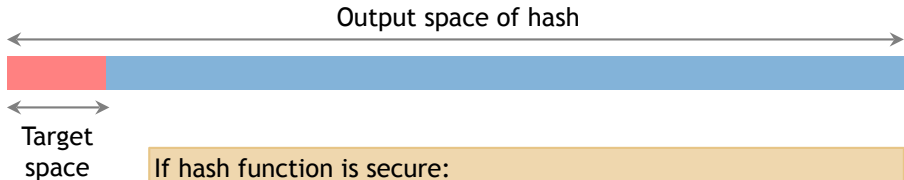
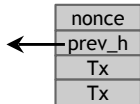
- How to pick a random node?
- How to avoid a free-for-all due to rewards?
- How to prevent Sybil attacks?

## Proof-of-work

- To approximate selecting a random node:
  - select nodes in proportion to a resource that no one can monopolize (we hope)
- Equivalent views of proof of work
  - Select nodes in proportion to computing power
  - Let nodes compete for right to create block
  - Make it moderately hard to create new identities

# Hash puzzles

To create block, find nonce s.t.  
 $H(\text{nonce} \parallel \text{prev\_hash} \parallel \text{tx} \parallel \dots \parallel \text{tx})$  is very small



If hash function is secure:  
only way to succeed is to try enough nonces until you get lucky

# Proof-of-work properties

## Property 1: Difficult to compute

- $122.8 \times 10^{18}$  hash/s (EHash/s) (10/12/2020)
- 312k transactions per day (10/12/2020)
- 77.78 TWh per year (2020, estimate)

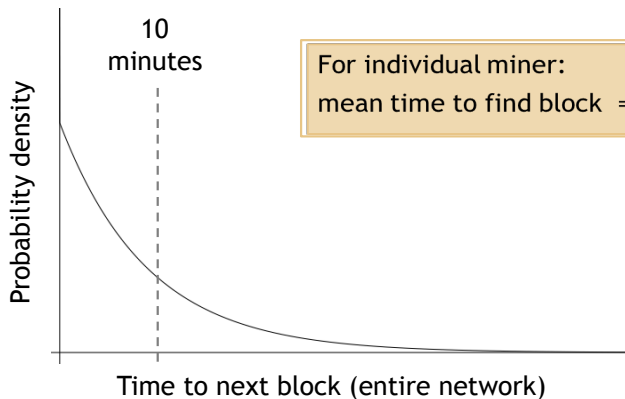
## Property 2: Parameterizable cost

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

## Property 3: Trivial to verify

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

## Solving hash puzzles is probabilistic



For individual miner:

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

# 51% Attacks

## Key security assumption

- Attacks infeasible if majority of miners weighted by hash power follow the protocol

## What can a “51% attacker” do?

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



# 51% Attacks

## Key security assumption

- Attacks infeasible if majority of miners weighted by hash power follow the protocol

## What can a “51% attacker” do?

- Steal coins from existing address? NO
- Suppress some transactions
  - From the blockchain?
  - From the P2P network
- Change the block reward?
- Destroy confidence in Bitcoin?

# 51% Attacks

## Key security assumption

- Attacks infeasible if majority of miners weighted by hash power follow the protocol

## What can a “51% attacker” do?

- Steal coins from existing address? NO
- Suppress some transactions
  - From the blockchain? YES
  - From the P2P network
- Change the block reward?
- Destroy confidence in Bitcoin?

## 51% Attacks

### Key security assumption

- Attacks infeasible if majority of miners weighted by hash power follow the protocol

### What can a “51% attacker” do?

- Steal coins from existing address? NO
- Suppress some transactions
  - From the blockchain? YES
  - From the P2P network NO
- Change the block reward?
- Destroy confidence in Bitcoin?

# 51% Attacks

## Key security assumption

- Attacks infeasible if majority of miners weighted by hash power follow the protocol

## What can a “51% attacker” do?

- Steal coins from existing address? NO
- Suppress some transactions
  - From the blockchain? YES
  - From the P2P network NO
- Change the block reward? NO
- Destroy confidence in Bitcoin?

# 51% Attacks

## Key security assumption

- Attacks infeasible if majority of miners weighted by hash power follow the protocol

## What can a “51% attacker” do?

- Steal coins from existing address? NO
- Suppress some transactions
  - From the blockchain? YES
  - From the P2P network NO
- Change the block reward? NO
- Destroy confidence in Bitcoin? YES!!!

# Limits for Bitcoin

## Hard-coded limits for BitCoin

- 10 min. average creation time per block
- 1M bytes in a block
- 20,000 signature operations per block
- 21M total Bitcoins maximum
- 50,25,12.5... Bitcoin mining reward

## Throughput limits in Bitcoin

- 1 Mbytes/block every 10 minutes
- > 250 bytes/transaction
- 7 transactions/sec
- Compare to:
  - VISA: 2,000-10,000 transactions/sec
  - PayPal: 50-100 transaction/sec

# Table of contents

- 1 Naive coins
  - Goofy coin
  - ScroogeCoin
- 2 Introduction
- 3 Preliminaries
  - Hash function properties
  - Hash-based data structures
  - Signatures
- 4 Bitcoin
  - Ledger
  - Decentralization in Bitcoin
  - BitCoin P2P Network
  - Consensus basics
  - Mining
- 5 History

# History, reloaded

## Blockchain 1.0 – Bitcoin (2009)

- Hard-coded cryptocurrency application
- Limited stack-based scripting language
- Proof-of-work-consensus (hardcoded)
- Native cryptocurrency (BTC)
- Permissionless blockchain system



# History, reloaded

## Blockchain 2.0 – Ethereum (2014)

- Distributed applications (smart contracts) in a domain-specific language (Solidity)
- Proof-of-work Consensus (hardcoded)
- Native cryptocurrency (ETH)
- Permissionless blockchain system

# History, reloaded

## Blockchain 3.0 – Hyperledger Fabric (2017)

- Distributed applications (chaincodes) in different general-purpose languages (e.g., go, Java)
- Modular/pluggable consensus
- No native cryptocurrency
- Multiple instances/deployments
- Permissioned blockchain system

# There and back again: from Bitcoin to BFT

- Designated set of homogeneous validator nodes
- BFT/Byzantine agreement
  - Tolerates  $f$ -out-of- $n$  faulty/adversarial nodes
  - Generalized quorums
- Tx sent to consensus nodes
- Consensus validates tx, decides, and disseminates
- Central entity controls group membership
  - Dynamic membership changes in protocol
  - Membership may be decided inline, by protocol itself

# Consensus in permissioned blockchains

- Trust model
  - $n$  nodes, some fraction  $f$  are faulty
- Protocols are well-understood
  - Paxos/Viewstamped Replication (VSR), ZooKeeper, Raft (many more!) tolerate crashes
  - PBFT, Byzantine randomized consensus (many more!) tolerate malicious nodes
- Despite that, many platforms have invented their own protocols
  - Often without sufficient public analysis
- Many platforms came out
  - Hyperledger Fabric, Tendermint, Symbiont, R3-Corda, Iroha, Kadena, Chain, Quorum, MultiChain

# Reading Material

- A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies*. Princeton University Press, Feb. 2016.  
<http://www.disi.unitn.it/~montreso/ds/papers/BitcoinBook.pdf>
  - In this presentation: Chapters 1-3 (77 pages)
  - Warning: 300+ pages
- F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys and Tutorials*, 18(3):2084–2123, 2016.  
<https://eprint.iacr.org/2015/464.pdf> (Suggested)
  - 37 pages
- C. Cachin and M. Vukolic. Blockchain consensus protocols in the wild. *CoRR*, abs/1707.01873, 2017.  
<http://arxiv.org/abs/1707.01873>