# Introduction to Blockchain

Muhammad Moinur Rahman <moin@bofh.im>

# INTRODUCTION

# Blockchain != Bitcoin/Cryptocurrency

# Overview

- Prologue: A chess-by-mail analogy
- What problem does a blockchain solve?
- How do they work?
  - Hash chains
  - Deciding what blocks are valid on the chain
  - Deciding whether we have the current chain
- Permissioned blockchains, proof of work, etc.
- Wrapup
- Handson on Blockstack

# Warm-up: A chess game by mail

- Alice sends Bob “1 e4”
- Bob sends back “1 ... e5”
- Alice sends Bob “2 Nf3”
- ...
- Each of these messages is one move in the game
- What’s necessary for them to be able to play the game?

# Agree on the state of the board

- If they don’t agree on the state of the board, they can’t play a game!
  - Both know the starting positions of the board.
  - Both know the sequence of messages so far.
  - Those messages make up a transcript of the game.
  - Thus, they can reconstruct the state of the board.
- If we agree on history, we agree on the present state of the world!

# How blockchain plays?

- We have some distributed system
- We need to all agree on the state of some system
- We all agree on the initial state of the system
- A blockchain contains a history of individual transactions
- Thus: We can all agree on the current state of the system
- A blockchain lets mutually-distrusting entities agree on history… which lets them agree on the state of the system now.

# Why is this important?

- Example: Bitcoin
- Suppose I want to transfer 10 BTC to you.
- You need to know whether my account has 10 BTC in it.
- For that, you need to know the current state of the system.
- Note: You need to know the current state. If you're looking at an old state of the system, I might be paying you with  money I've already spent!

# What problem are addressed?

- A blockchain lets us agree on the state of the system, even if we don’t  all trust each other!
- Ultimate goal: We all need to agree on the state of some system.
  - How much BTC in each account?
  - Who owns which property?
  - What’s the current state of my program?
- We can all agree on that if we agree on history.
  - Starting state + history => current state
- We don’t want a single trusted arbiter of the state of the world.
  - We want some level of decentralization—not a single point of failure or compromise.

# Trusted Arbiter

- If we had a completely trusted arbiter, we wouldn’t need a  blockchain!
- We could just define reality as whatever TA said it was.
- For a payment system, imagine TA as the bank
  - Bank provides the official sequence of transactions and account balances
  - When you want to spend your money, you send a message to bank
  - Bank permits transaction if you have money, and updates account balances.

# One trusted arbiter, then?

- Single point of failure
  - If the TA goes down for a week, the system stops working!
- Concentration of power
  - “He who controls the past, controls the future”
  - TA can censor transactions, impose new conditions to get transactions  included in history, etc.
- Maybe there’s nobody we all trust

# What does blockchain gives us?

- Distributed system
- We don’t all trust each other or any single entity
- We want to agree on history
- ...so we can agree on the state of our system...
- ...so we can do something.
- We get the functionality of a trusted arbiter … without needing a trusted arbiter

# How blockchain works?

- A blockchain is a sequence of hash-chained records
  - Once you've seen record N, you can't change anything in the past.
- Some procedure for adding blocks to blockchain
  - Who gets to add blocks? How is it done?
- Validity conditions for new blocks
  - Are transactions valid? Are digital signatures correct? Etc.
  - Enforced by consensus-–chains with invalid blocks won't be accepted.
- Some procedure for deciding between alternative candidate blockchains.
  - When Alice and Bob have different pictures of history, there's some way for them to eventually come to agreement about who is right.

# Building Block: Cryptographic hash functions

A cryptographic hash function:

- Takes any bitstring as an input* (Like a 10 MB file)
- Produces a fixed-length output  (Typically 256 or 512 bits)
- Nobody can find collisions.
- Examples: SHA256, SHA512, SHA3-256, RIPEMD-160, X11,  Ethash, Cryptonite, Nist5

- * Sometimes there’s a (really huge) maximum input length.

# Collision

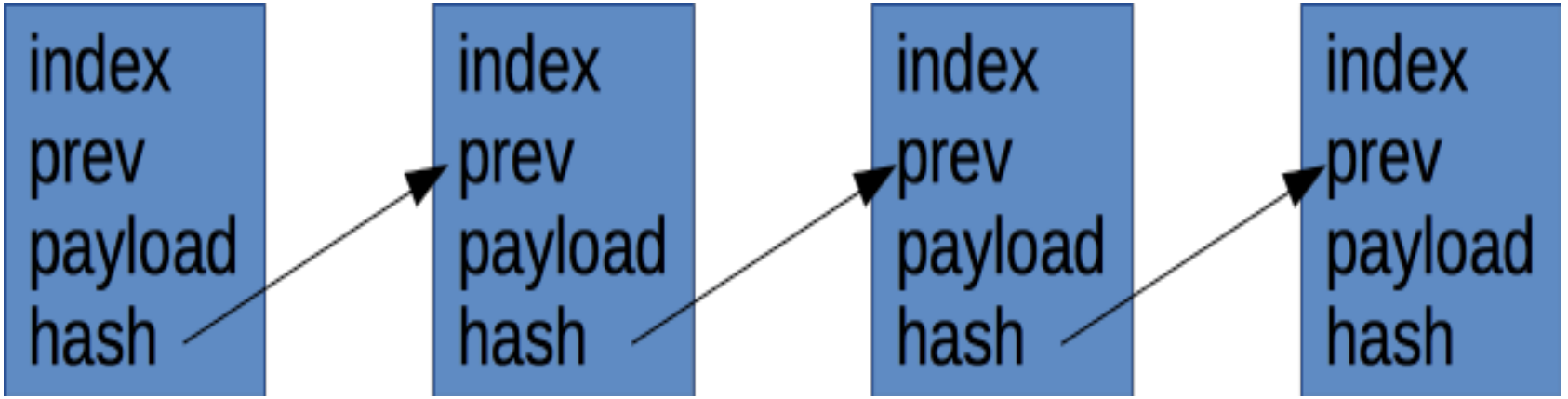
- Suppose I can find two different inputs X and Y so that  Hash(X) = Hash(Y)
- That’s a collision.
- For a cryptographic hash function to be any good, it needs to be collision-resistant.
- That just means it’s impossible in practice to find colliding inputs.

# Why is collision resistance useful?

- If nobody can find X != Y such that Hash(X) == Hash(Y),
- ...then we can use hash(X) as a kind of message digest of X.
  - Digital signatures actually sign hash(message) instead of message.
- Nobody can change X without changing hash(X)
  - If they could do that, they can find collisions for hash()
- hash(X) also commits to X.
  - Once I've seen hash(X), later, you can show me X, and I'll know it's the value you committed to
  - ...you can't show me some other X*, because it won't have the same hash.

# Building block: Hash chains

A hash chain is a sequence of records in which each record contains  the hash of the previous record in the chain, and the hash of all the  current record's contents.




A sequence of records linked together; each record contains the hash of the previous record.

# What does Hash Chain gives us?

- We’re using a cryptographic hash function like SHA256.
- That means nobody can find two inputs with the same hash value.
- ...and that means that record N contains a commitment to record N-1
- ...which contains a commitment to record N-2, which contains a  commitment to record N-3, and so on.

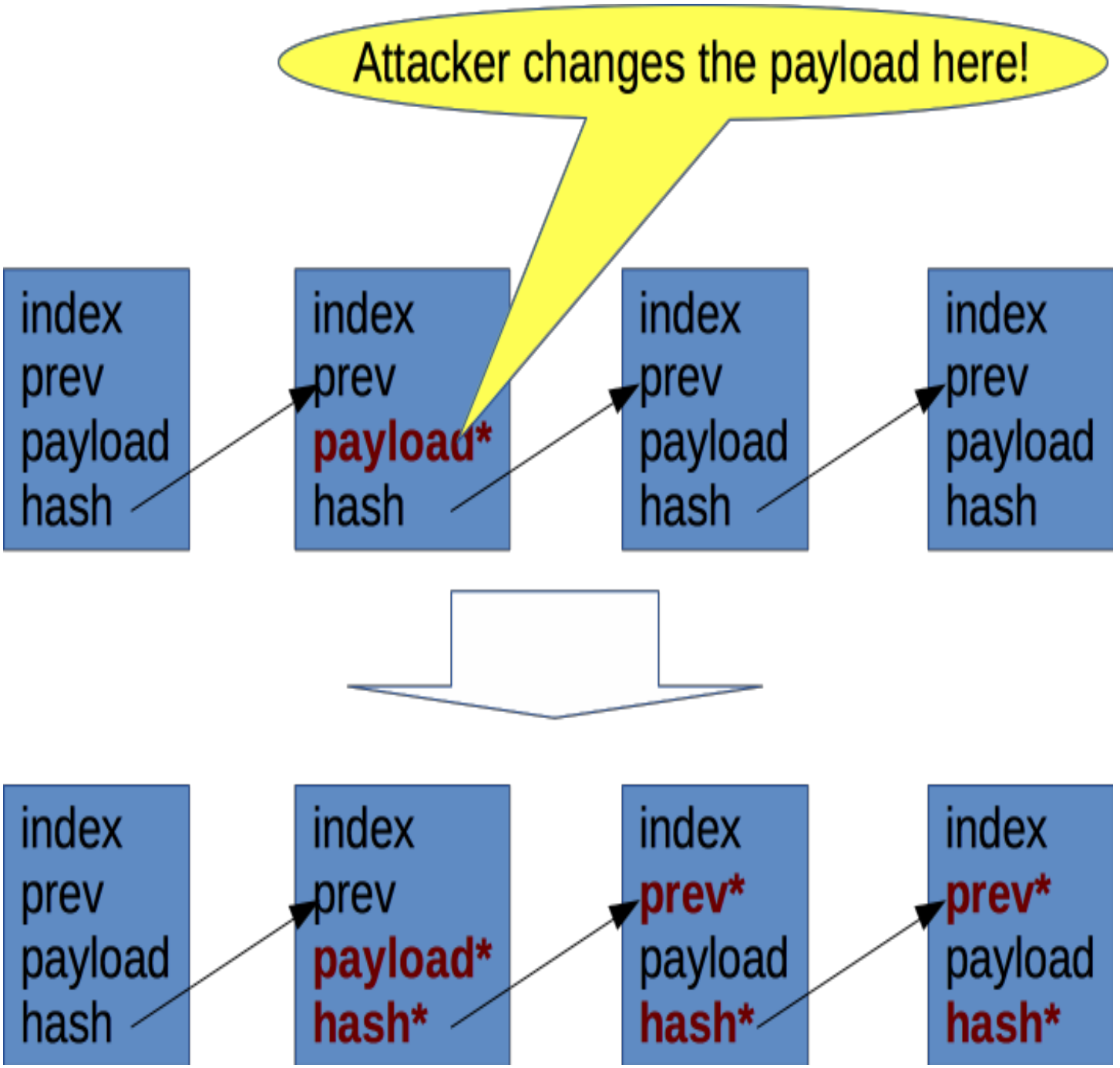



Figure: A change in one record in the hash chain propagates forward to change the hashes in all future records.

# Hash Chain vs Blockchain

- Hash chains have the property that every record contains a commitment to all previous records.
  - If you change record N, this changes the final hashes of records N+1, N+2, ...
- Result: Once we all accept record N, we have locked in the contents of record 1, 2, 3, ..., N-1 as well.
- Blockchains use hash chains as a component
- Hash chains are also useful in a lot of other contexts
  - For example, a system with a trusted arbiter can use a hash chain to limit the arbiter's power—even the arbiter can't change history.

# The Blockchain

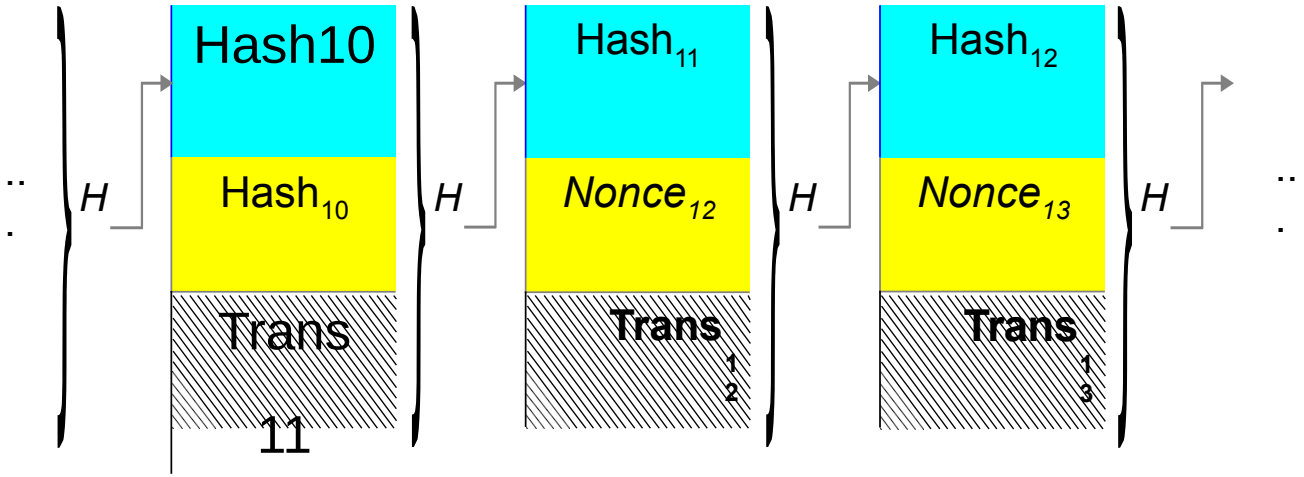



Figure: A block chain containing three blocks, each containing the hash of the previous block, and each containing a sequence of transactions and a nonce.

- Each block in the chain *commits* to all previous blocks and transactions

# Building Block: Validity conditions

What will the world accept as the next block?

- We don’t have some trusted entity to decide what may be added to block chain
- ...so we have to decide what blocks are valid.
- Example: Bitcoin
  - Signatures needed for moving BTC from an account
  - Not allowed to leave a negative balance in an account
  - Block must contain correct proof-of-work
- A proposed additional block that doesn’t meet these conditions won’t be accepted by the rest of the network.
- Enforced by consensus

# Forked Chains

- Blockchains are used in distributed systems
- It’s possible for different parties in the system to have a disagreement
  - There’s no trusted arbiter to decide who is right!
- Any blockchain must deal with this issue somehow!
- Suppose Alice and Bob no longer agree on the state of the world
  - (because they don’t agree on history)
- How do they come to an agreement on who’s right?
  - Several different techniques to resolve disagreements

# Adding new blocks to the chain

- Any blockchain system has to determine who can add new blocks to  the chain, and how it’s done.
- Two main ideas we’ll discuss below
  - Proof of work
  - Permissioned blockchain
- Also more ideas I’m not going to talk about
  - Proof of stake
  - Proof of storage
  - Probably several more I’ve never heard of

# Building Block: Proof of Work

- I want you to do a big computation.
  - I want you to prove you did it.
  - I don't want to do much work checking the proof.
- Why is this useful?
  - Limits the rate of new blocks
  - Makes attempts to add invalid blocks to the chain expensive
  - Provides a clear way to decide between competing chains when there is a disagreement—the one with the most work wins.
- Note: Not all blockchains use proof of work

# Hash-based proof of work

- I give you challenge C and limit L = 2^220.
- Ask you to find N such that
- SHA256(C||N) < L
- Expected work = 2^36
- Each new N has prob 2^-36 of success
- When you succeed, it only takes me one hash to check.
- This is more-or-less Adam Back's hashcash scheme

# Proofs of work in every block

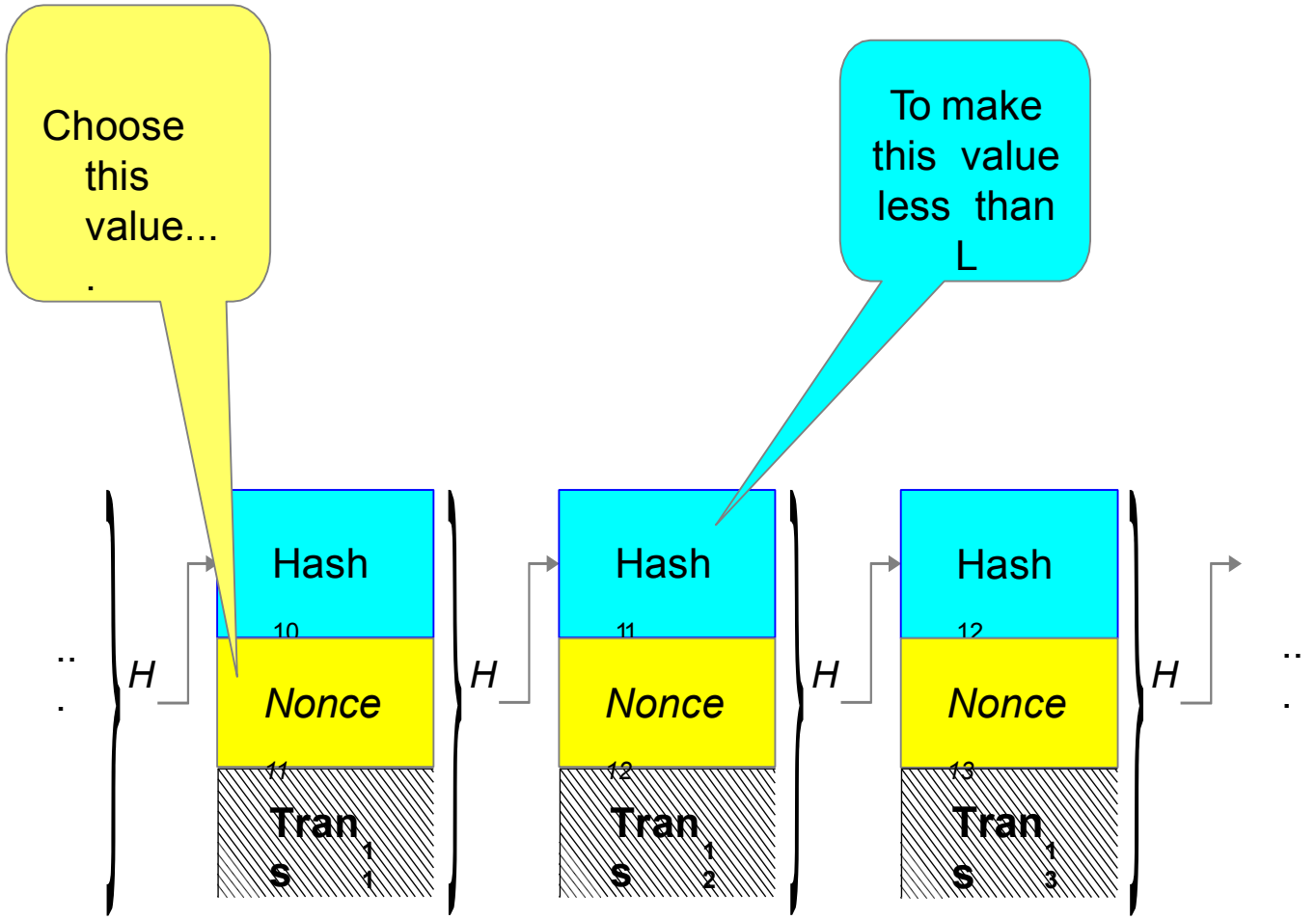



Figure: Three blocks in a block chain. The nonce in the first block is chosen to force its hash value (which appears in the second block) to be less than L.

# Proofs of work solve some problems...

- We can resolve disagreements.
  - When chain forks, take fork with most work.
  - When there's a tie, keep working till one of the chains has the most work.
- Discourage people trying to add invalid blocks to chain.
  - You spend money adding a block to chain...
  - ...but if it's not valid, nobody accepts it.
- Part of how Bitcoin's very clever design of incentives works.

# ...but introduce others

- Expensive—lots of energy used do generate proofs
  - Done by “miners” in Bitcoin
  - Use special-purpose mining rigs optimized for doing proofs of work.
  - Environmental impact—uses lots of power, accomplishing no useful goal except keeping blockchain working
- Slow—proof of work seems to put a limit on transaction speed
  - Even more when you consider need to resolve potential disagreements
  - Bitcoin rule of thumb is wait 6 blocks (about an hour) to be sure of transaction

# Permissioned blockchains

- An alternative to proof-of-work
- We have set of somewhat-trusted entities who can work together to add records to the blockchain.
- For example, we could have five trustees, and if any 3/5 vote in favor of accepting a block on the chain, then the block is added.
- Validity condition for adding a block = 3/5 signatures
- Resolution for conflicting chains = look for longest chain (aka most votes)
  - With 3/5 there shouldn't be any forked chains—someone would have to vote for two competing blocks!

# Incentive design

- The real genius in Bitcoin's design is the way incentives are aligned
  - Untrusted, self-interested miners keep the system working
  - They have a big incentive to follow the protocol
  - They have substantial capital invested in Bitcoin, so they also have an
  - incentive to avoid any attack that would undermine their investment
  - This all works because Bitcoin is all about moving money around, so it's easy to build payoffs into the protocol.
- Other blockchains (especially permissioned ones) have to find alternatives to incentives
  - Not so obvious how to build a payoff into a protocol to store medical records

# Why do we trust trustees?

- Existing business or legal arrangements?
- Incentives for playing fair?
- Reputation?
- We have the “And then you go to jail” problem....

# “And then you go to jail”

- It’s usually a bad idea to build crypto protocols that rely on outside enforcement mechanisms
  - Sending misbehaving users to jail
  - Suing people who don’t follow the protocol
  - Assuming that damage to someone’s reputation will convince them to behave properly.
- Lots of examples of these things not working
  - Even if they do work, they tend to be slow
- This is something any permissioned blockchain has to solve

# Wrapup 1: Blockchains let us agree on history

- We don’t have to trust each other
- We don’t have to have a trusted third party
- System is distributed
- Agreeing on history => agreeing on state of system

# Wrapup 2: Blockchains and hash chains

- The Nth record in the hash chain commits to all previous records.
- Can’t change any previous record without making hash chain invalid.
- A blockchain is a hash chain with some other stuff added
  - Validity conditions
  - Way to resolve disagreements

# Wrapup 3: Permissioned vs Proof-of-work

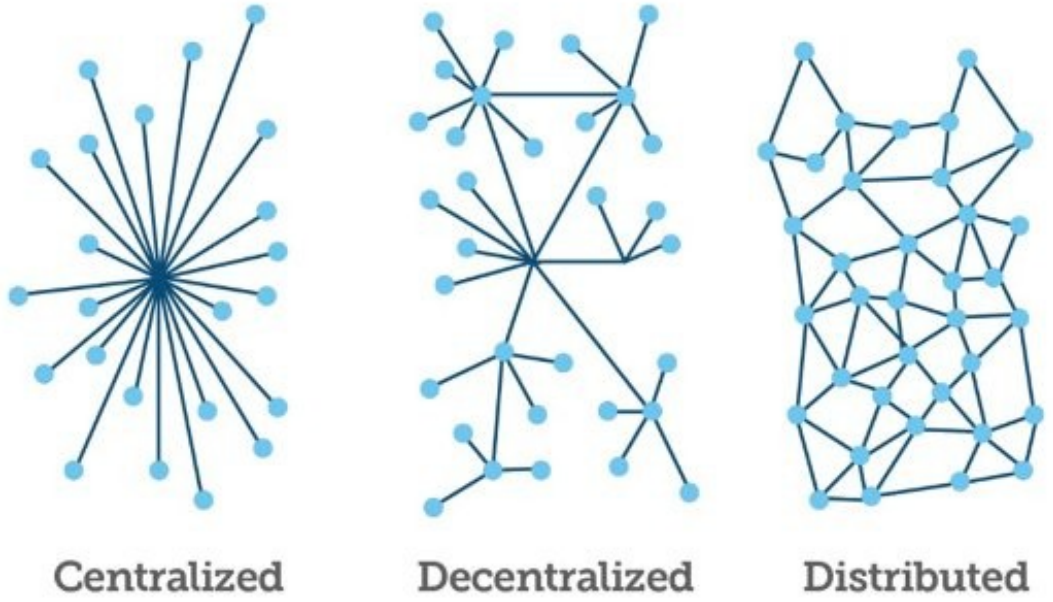
- Most blockchains in use now use proof-of-work
- Many new proposals use permissioned blockchains
  - Some set of somewhat-trusted entities
- There are other ways to do it
  - Proof of storage
  - Proof of stake
  - Probably more I don’t know about

# Questions

- Reference
- Introduction to Blockchain, John Kelsey

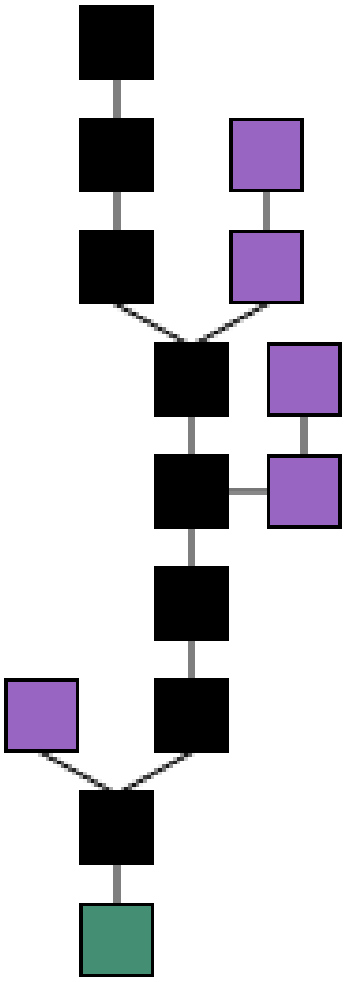
# What is Blockchain?

- A distributed Database
- Decentralized



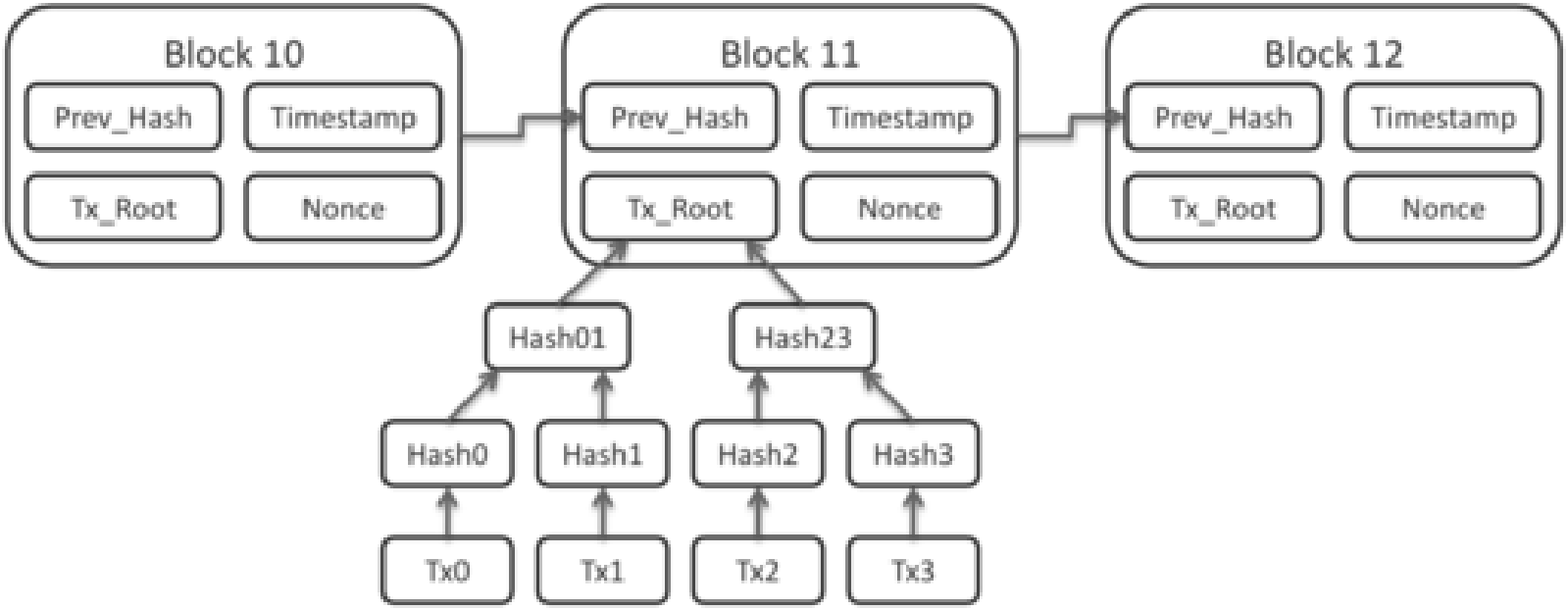
# What is Blockchain? - Continued

- Mainly list of records
- Always growing



# What is Blockchain? - Continued

- Tamper proof and Revision proof
- Open, permission less and public
- Byzantine Fault Tolerant



# How Blockchain Works?

- A growing list of records called blocks
- Each block contains
  - A timestamp
  - A link to previous block
- Can be managed
  - Publicly or autonomously
    - Peer-to-peer network
    - Distributed Time-stamping
  - Privately

# Uses of Blockchain

- Recording of events
- Medical Records
- Identity Management
- Transaction processing
- Documenting provenance
- Financial Transactions
- Marketplace
- Smart Contracts
- Digital Products Marketing
- Property Records
- Voting
- Cloud Storage

# History of Blockchain

- 1991 – Stuart Haber & Scott Stornetta
- 1996 – Ross J. Anderson
- 1997 – Michael Doyle
- 1998 – Bruce Schneier
- 1998 – Nick Szabo – Bit Gold
- 2008 – Satoshi Nakamoto
- 2014 – Blockchain 2.0

# Internet

- A 30 years old technology
- Based on IP routing and DNS
- Primary purpose was to share abundance of Data
- Now under mass surveillance
- Now censored by different governments to capitalize political benefits
- Always requires DNS information to make full communication


# Internet

- Centralized by the so called ROOT DNS servers
- Secured by DNSSEC
- Web trust system is broken
- Under the control of around 1200+ CA
- Most of the https are weakly configured
- Prone to catastrophe, zombie apocalypse, alien invasion, Government shutdown

# Words from ..

- “We didn’t focus on how you could wreck this system intentionally,” - Vinton G. Cerf.
- "I invented the web. Here are three things we need to change to save it
  - We’ve lost control of our personal data
  - It’s too easy for misinformation to spread on the web
  - Political advertising online needs transparency and understanding” - Sir Tim Berners Lee

# DNS

- A decentralized database maintained by root servers
- Secured implementation by DNSSEC(Hardly implemented by Domain owners)
- Web anchor of trust is based on CA(Run by large Companies, Controlled by Governments)

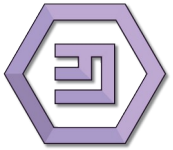
# DNS in a Distributed Blockchain

- Censor-free
- Distributed, hard to knock down by a single attack or Government
- Supported by TOR or I2P
- Private

# DNSchain - Implementation

- Namecoin – Bit DNS
- OKTurtle
- Emercoin – EMCDNS
- And more ..





# Data storage in Blockchain

- Privacy at stake for major Cloud Storage providers
- Personal information stored in clouds
- Accessible by Providers/Governments
  - With Warrant
  - With Subpoena
- Encrypted Storage
- Ransomware

# Data storage in Blockchain - Implementation

- storj
- sia.tech

# Traffic Routing

- Always goes through an ISP
- Running Deep Packet Inspection
- National Firewall
- DNS Cache Poisoning
- Plain traffic is intercept-able
- Caching Data poisoning
- Comes with Privacy Demolished

# Blockchain in routing(BGP) – What if

- Voting capability to avoid bad routes
- Overlay Network with Path Performance Computation
- A DHT maintaining RemyCC (Remy Congestion Control)
- Prefixes are added/removed/modified by DPKI

# SDN in Blockchain

- SDN – Cutting Edge Technology
- Uneducated wrong configured Infrastructures
- API/Programmability is vulnerable
- Flow Table can be modified remotely

# SDN in Blockchain – What if ..

- Flowtable is maintained in a blockchain
- Modification of flowtable is authenticated against KSI(Keyless Signature Infrastructure or DPKI)
- Saving the events in a blockchain to track it down to its root
- Easier Log readability
- Authenticate agents, messages, control interfaces, devices, state of a service

# IoT in Blockchain

- A trillion dollar Industry
- Billions of devices will be connected
- Interact in between them
- Going to be the most vulnerable systems from the prying eyes
- Decentralization is required, considering centralized infrastructure

# IoT in Blockchain – What if

- Billions of ongoing transactions will be stored in blockchain
- Rather than centralizing the Data storage
- A single fail-proof network
- No MITM attack
- A single Tamper proof DHT(Distributed Hash Table)
- Blockchain is already a proven technology with billions of Dollar market for Cryptocurrencies

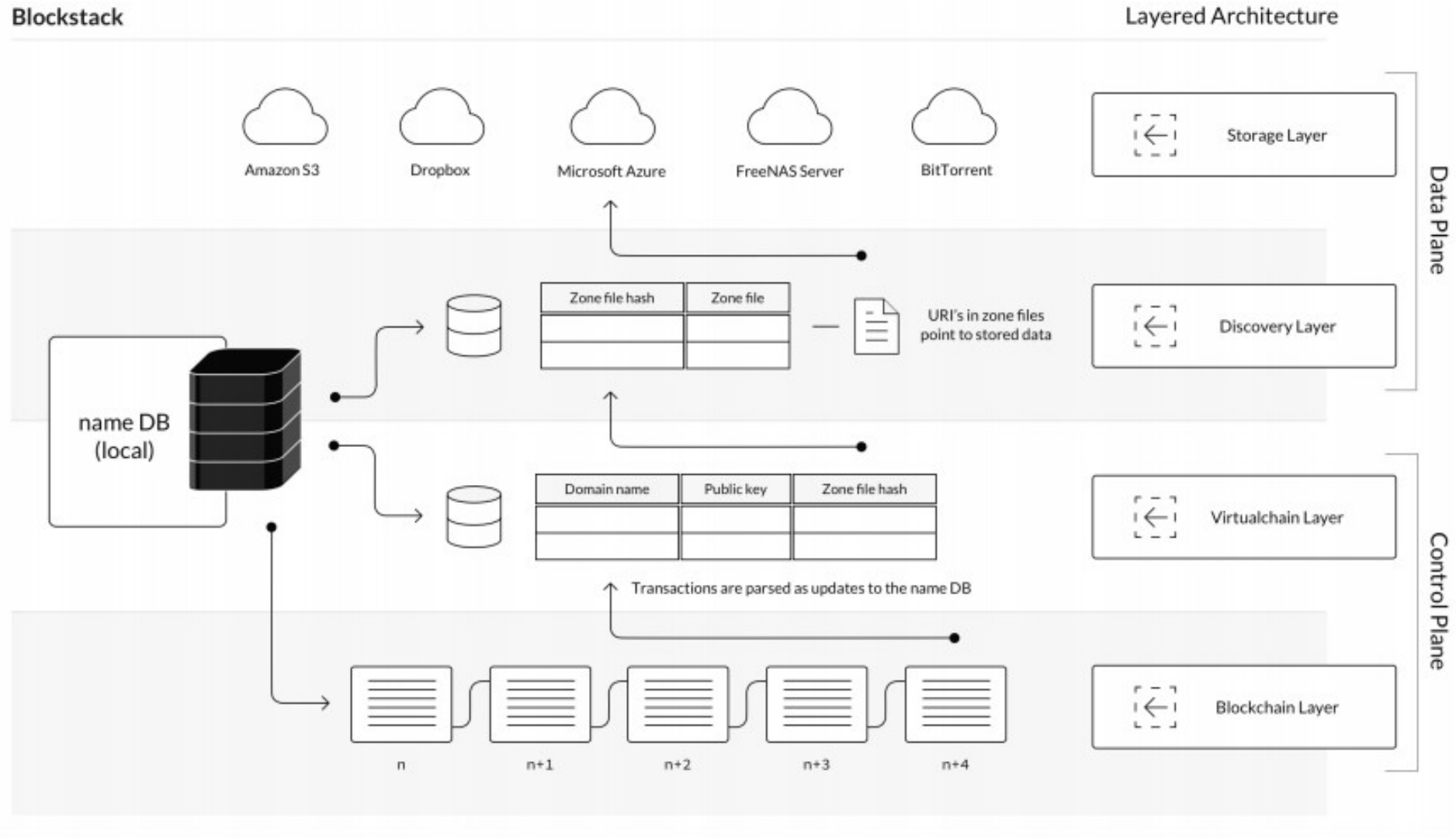
# BlockStack

- Decentralized Internet
- A Full Stack of Apps
  - Identity
  - Storage
  - Payments
- With the possibility of
  - Decentralized Social Network
  - P2P Marketplace
  - Community Run Voting
- Blockchain

# BlockStack - Details

- Blockchain Layer
- Virtual chain Layer
- Discovery Layer
- Storage Layer
  - Any Cloud Storage Provider
  - Personal Storage at your bunker
- BNS
- ATLAS Network

# BlockStack - Architecture



# End Goal

- A Censorship free Internet
- Freedom of Speech
- Privacy in the Digital Age

# Questions ..

Thank you ..