Blockchain Security in 30 Minutes (not a cryptocurrency investment talk)

Parsia Hakimian

March 12th 2018

Who is this guy

- **Appsec Consulting @ Synopsys**
  - EDA software company you have not heard of
  - Software Integrity Group (SIG)
    - Cigital <-- I was here
    - Blackduck
    - Coverity
    - Defensics

What this is NOT!

- How can I get rich with Bitcoin? **Time travel.**

- How do I keep my private keys safe? **Paper wallet.**
- What should I mine? **Doge coin.**

First!

- **Do I need a blockchain? Not Really!**
- **Reinventing databases; a tragedy in two parts**

A decentralized git repositories registry on blockchain could save us all. Is anyone working on something similar?

1:10 AM - 28 Feb 2018

- GitHub goes down.
- I push my repo on another origin, even self hosted.
- I update the blockchain registry.
- The dev that included my library asks the distributed registry where to pull the code.
- The registry provides the new git repo.
- Everything still builds.

6:32 AM - 1 Mar 2018

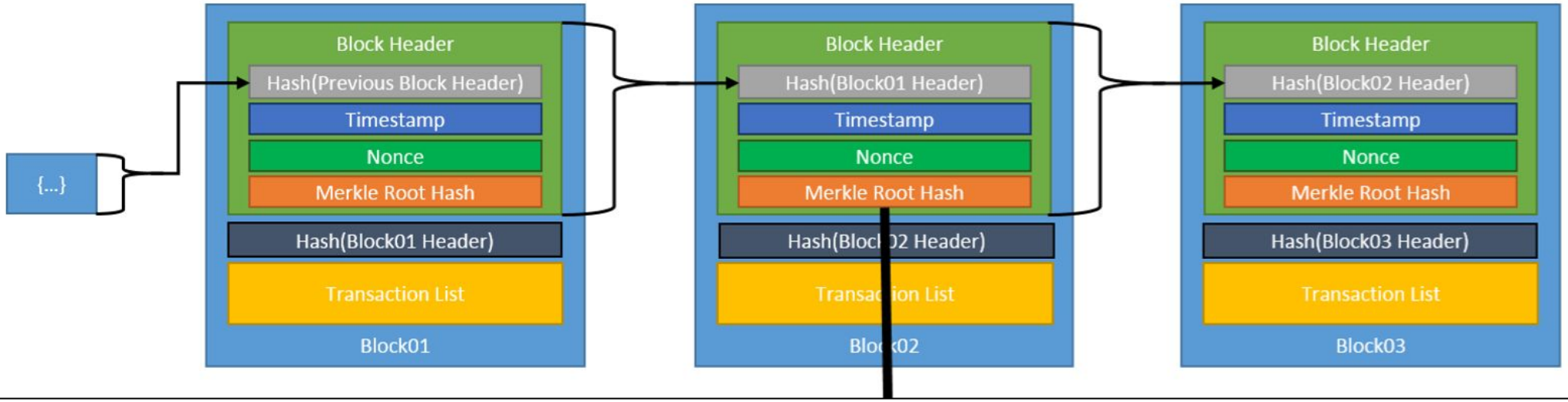
Blockchain or Glorified Distributed DB?

- **Data you want to store:**
  - Dank memes, events, transactions, network logs, etc.
- **Ledger:**
  - Event log
- **Distributed Ledger:**
  - Distributed and updated event log
- **Blockchain**
  - An implementation of a distributed ledger

Most important slide

- **Blockchain System == Distributed Network**
  - Solves some problems
  - Introduces new challenges
- **Attacks/Concerns from literature apply**
  - Sybil attack
  - Fault tolerance
  - CAP theorem

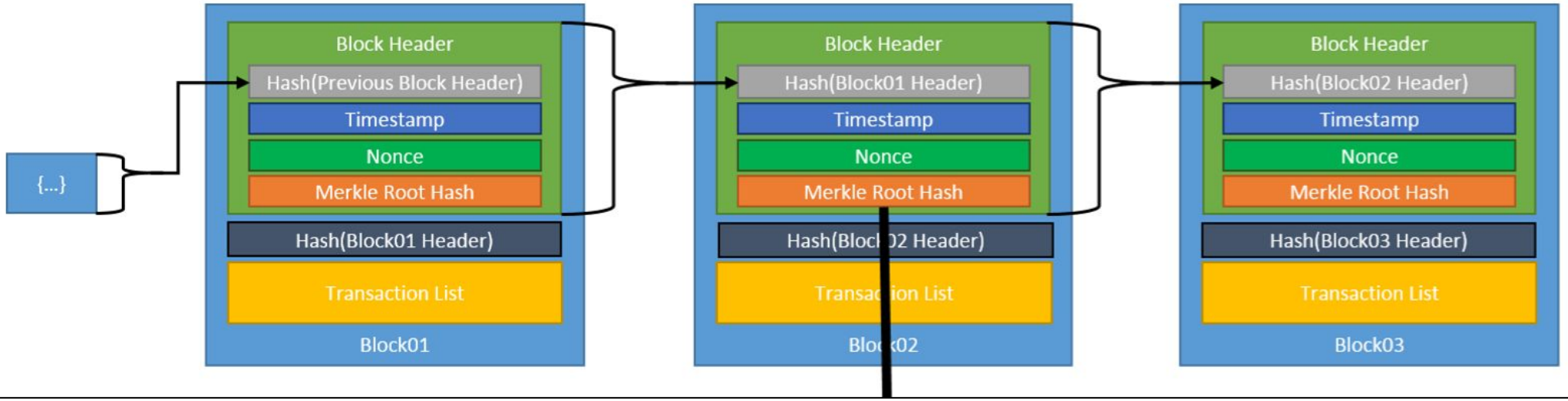
Blockchain



Source: NISTIR 8202 - Blockchain Technology Overview - January 2018 Draft

- **Write-once distributed ledger**
- **History is verifiable**
- **HashChain**

Problem 0-0: Integrity

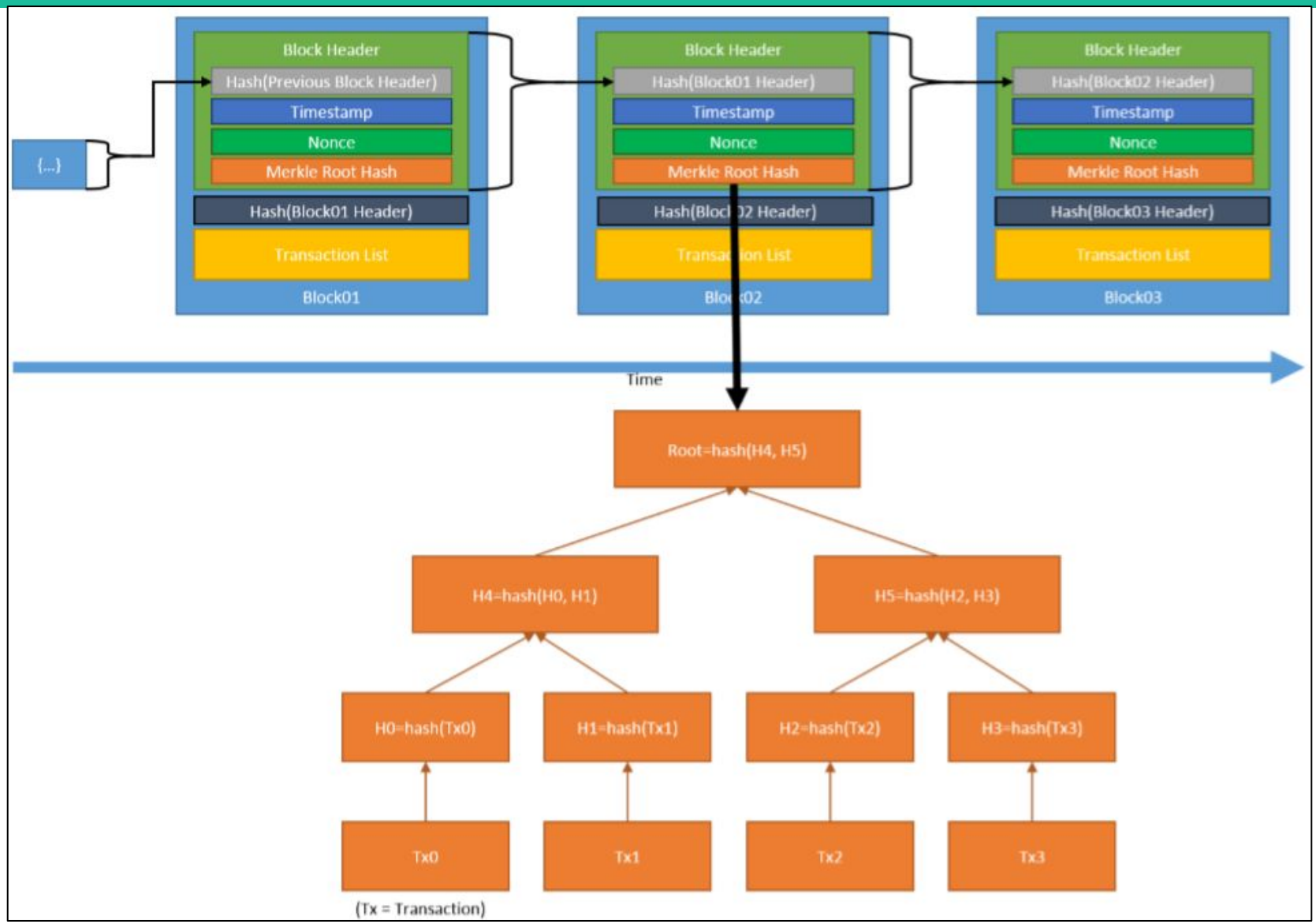


- **HashChain does the trick**
- **Good hash function:**
  - SHA/Scrypt/Ethash/Equihash/etc...

Hash Function

- **Any-length input > fixed-length output**
  - **Diffusion**
    - Smallest change in input > completely different output.
  - **Pre-image resistant**
    - Hard to predict input that results in a specific output.
  - **Collision resistant**
    - Hard to find two inputs that produce in the same output.
  - **Second pre-image resistant**
    - Having one specific input, it should be hard to find a collision.

Problem 0-1: Integrity of Transactions



Merkle Tree Root Hash

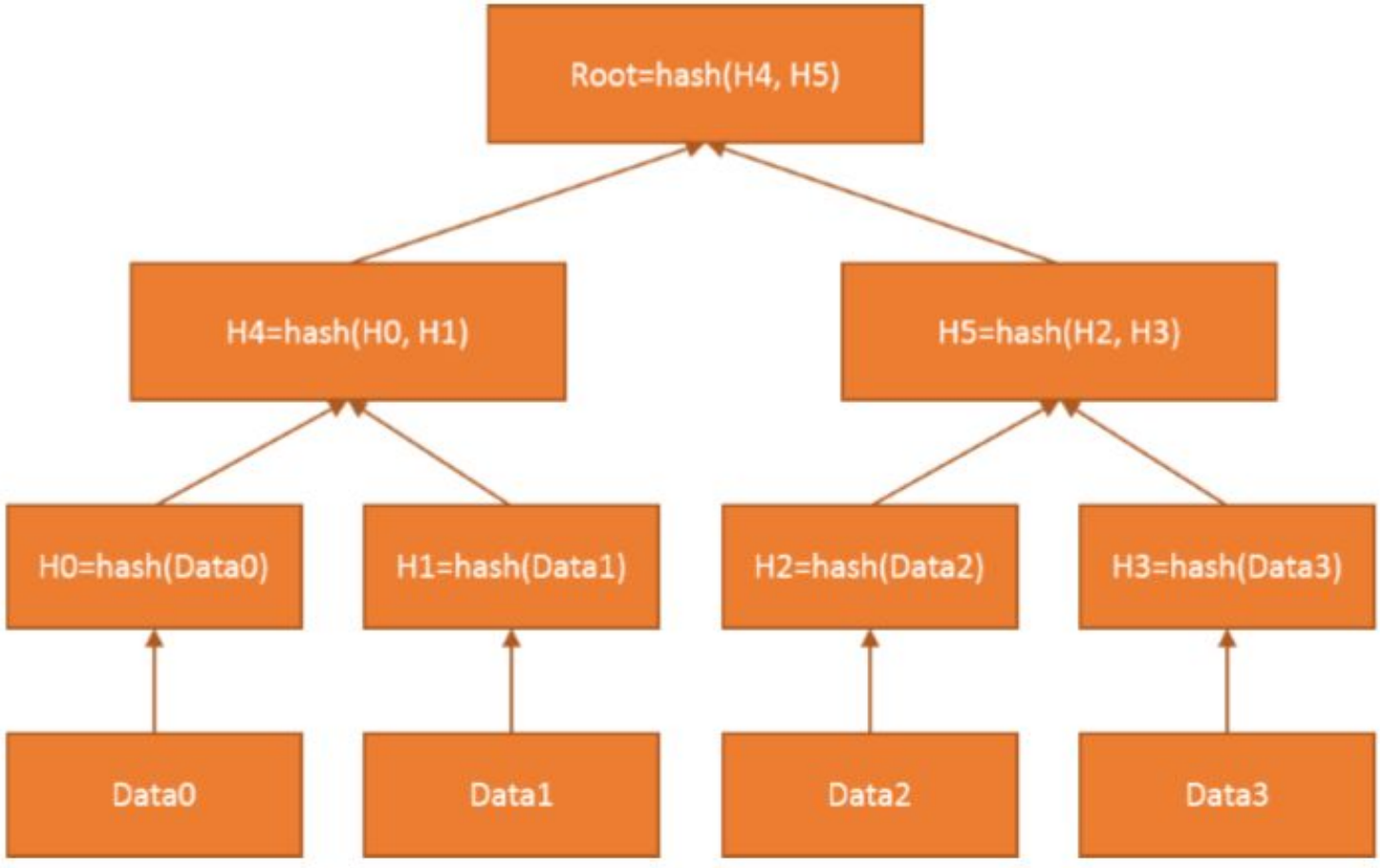


Figure 5: Example of a Merkle Tree

Source: NISTIR 8202 - Blockchain Technology Overview - January 2018 Draft

Problem 1: Who can join?

- **Permissioned**
  - Central authority vets nodes
  - Not all nodes can do everything
  - Enterprise blockchains
    - Hyperledger Fabric – IBM & Linux foundation
    - Quorum – JPMC
- **Permissionless**
  - Everyone can submit/read/write
  - Most Cyptocurrencies

Federal Register on a Blockchain

- **Federal Register**
  - Data: Rules and regulations
  - Submit transaction: Office of the Federal Register
  - Writer: Government Publishing Office or GPO
  - Reader/Verifier: Everyone
- **Public permissioned blockchain**
- **This is a horrible idea!**

Problem 2: Who's who?

- **Permissioned**
  - Central vetting authority
  - Certificates
- **Permissionless**
  - Private/Public key
  - Public key > hash > stringify > address
    - Stringify: Base56
  - Wallets/transactions can be traced

Problem 3-0: Verifying Transactions

- **Verify validity**
  - Sign transaction with private key
  - Verify with public key
  - Use public key to derive address
- **Verify balance**
  - Calculate (in – out) x coins

Problem 3-1: Theft

- **Private key access == Owner**
- **No undo - SFYL**
  - Immutable ledger
- **Exchange theft**
  - Bitconnect (straight up Ponzi scheme) - “Bitconnect Guy”
- **Solution:**
  - 2FA
  - Paper wallets/Multi-signature wallets

Nodes

- **Full Node**
  - Store complete history of blockchain
- **Mining Node**
  - Full node + maintain blockchain
    - Create new blocks
- **Lightweight Node**
  - No mining or storing
  - Submit transactions
  - Pass data around

Problem 4: Distributed Backups

- **Solution**
  - Full/Mining nodes store everything
- **Challenge**
  - Waste of bandwidth
  - Bitcoin blockchain 110+ GB

Problem 5: Maintaining the Network

- **Permissioned**
  - Out-of-band incentives
    - Law: GPO
    - Business agreements: Banks
- **Permissionless**
  - Award Cryptocurrency
    - How bitcoin is created
  - New challenge
    - Who gets to create new blocks?

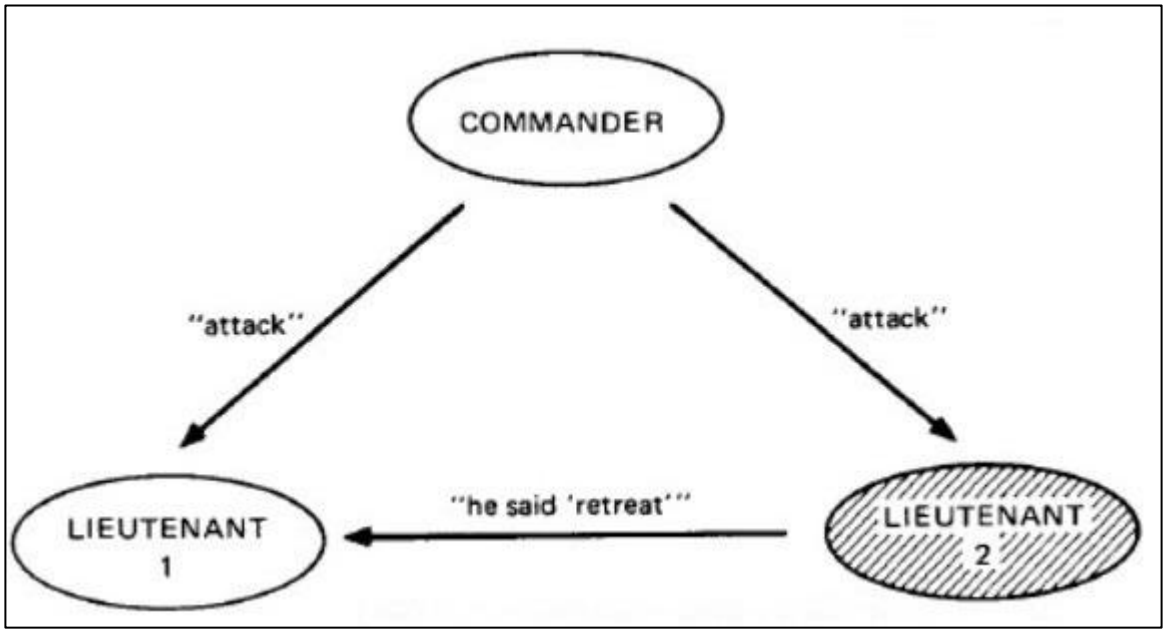
Problem 6: Malicious Nodes

- **Permissioned**
  - Nodes can go rogue/malicious
  - Nodes can be faulty
- **Permissionless**
  - Untrusted nodes
  - Everyone is malicious
- **Solutions?**
  - Distributed Networks literature

Academic Ivory Tower

Byzantine Generals

- **Byzantine Generals' Problem (1982)**
  - Leslie Lamport, Robert Shostak, and Marshall Pease
  - Reaching Consensus == Network agreeing on something



Academic Ivory Tower

Byzantine Nodes

- **Game of Telephone**
  - Malicious node: Evil me
  - Faulty node: Non-native speaker me
- **Byzantine Nodes**
  - Nodes with Byzantine Fault
  - Display different symptoms to different observers
    - Unintentional: Faulty nodes
    - Intentional: Malicious nodes

Academic Ivory Tower

Byzantine Failure

- **Byzantine Failure**
  - A network requiring consensus, failing to perform its service because of Byzantine nodes.
- **Game of Telephone – Order Pizza**
  - First person decides toppings
  - Whisper
  - Last person orders
  - Failure: Ordered pizza does not have correct toppings

Academic Ivory Tower
Byzantine Fault Tolerance

- **Reaching consensus with Byzantine Nodes**
- **Evading Byzantine Failures**
- **Impossible in our Telephone game**
  - One path from source to destination
- **Solution: Ensure message integrity**
  - Checksums (e.g. CRC32)
    - Works with non-malicious nodes (e.g. electronics)
  - Cryptographic Signing
    - Prevent tampering by malicious nodes

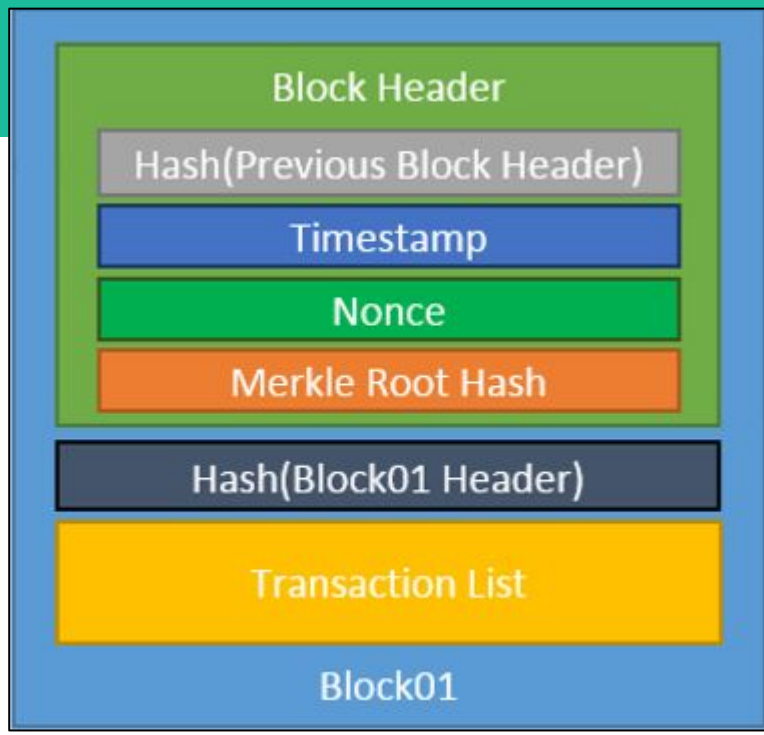
Problems 5 and 6 Revisited

- Nodes in Blockchain: **Byzantine Nodes**
- Consensus: **Who mines next block**
- Not agreeing on next miner: **Byzantine Failure**
  - Blockchain will not be maintained
- **Consensus Mode: Byzantine Fault Tolerance**
  - Proof of Work (PoW)
  - Proof of Stake
  - Round-Robin

Proof of Work (PoW) Consensus Model

- **Nodes compete to solve a puzzle**
- **Puzzle**
  - Difficult to solve but easy to verify (NP problem?)
  - Previous puzzles solves must not help with new ones
- **First node to solve makes new block**
- **Challenge**
  - Waste of time and energy

Bitcoin PoW



- **Puzzle**
  - Hash(block header) < 0000xxxxxxxx
  - Hash starting with 4 zeros or more
- **Mining**
  - Create candidate block
    - Choose & verify transactions from **unspent transaction pool**
      - Transaction fees
  - Choose nonce -> Calculate hash -> Check hash
- **Adjustable difficulty**
  - Change number of zeros

Problem 7: Conflicts

- **Decentralized networks have lag**
  - Different unspent transaction pools
- **Two valid blocks mined > Two blockchains**
  - Longer is chosen
  - If same length, wait for next block
  - Incentivizes nodes to accept valid blocks and mine on top of them

Problem 8: Sybil Attack – Double Spend

- **Also Majority Attack or 51% Attack**
- **Ignore valid blocks by others**
  - Keep cryptocurrency rewards
  - Fix: Conflict resolution strategy
- **Create longer chain with invalid transactions**
  - Double spend

Sybil Attack - Double Spend Fix

- **Computing: Proof of Work**
  - Hard to gain control of 51% hash rate
    - Hashing is a constrained resource
- **Ownership / Commercial Interest: Proof of Stake**
  - More stake in system == invested in its success
- **Hierarchical Trust: Authority**
  - Law and order - Federal Register

Current and Future Work

- **Enterprise blockchains**
  - Hyperledger: Fabric
  - Quorum
- **Smart contracts**
  - Ethereum - Quorum
    - Solidity/LLL/etc.
    - Ethereum Virtual Machine (EVM)
  - Hyperledger: Fabric
    - ChainCode (Golang)

Questions?

- **More reading:**
  - NISTIR 8202 Blockchain Technology Overview
  - “Do you need a blockchain?” by Karl Wüst, Arthur Gervais
  - https://parsiya.net/categories/blockchain/
  - Hyperledger: Fabric
    - https://github.com/hyperledger/fabric
  - Quorum
    - https://github.com/jpmorganchase/quorum