Blockchain Principles and Applications

Amir Mahdi Sadeghzadeh, PhD

Data and Network Security Lab (DNSL)
Trustworthy and Secure AI Lab (TSAIL)

Most slides have been adapted from ECE/COS 470, Pramod Viswanath, Princeton 2024 and CS251, Dan Boneh, Stanford 2023.

Recap

Basic Cryptographic Primitives

1. Cryptographic Hash Functions
2. Hash Accumulators
   Merkle trees

} Centralized Blockchain

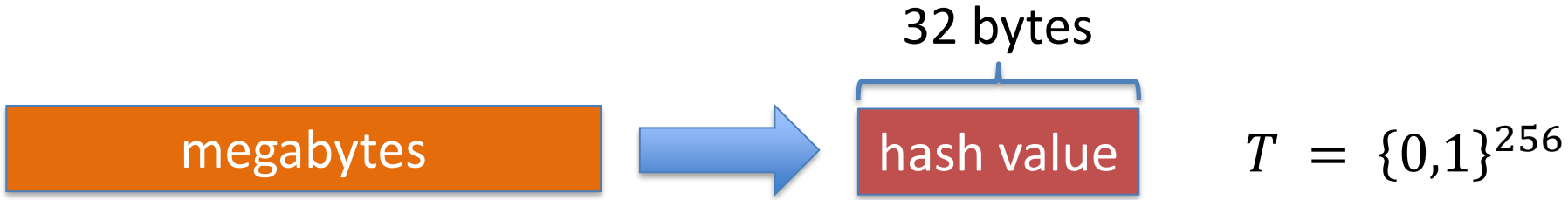
3. Digital Signatures — Decentralized Blockchain

Cryptography Background

(1) cryptographic hash functions

An efficiently computable function $H: M \rightarrow T$

where $|M| \gg |T|$



Hash Functions

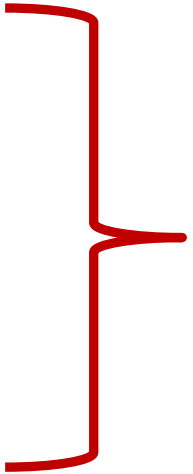
Defining Properties:

1. Arbitrary sized inputs

2. Fixed size deterministic output

3. Efficiently computable

4. Minimize collisions



Canonical application:

Hash Tables

Store and retrieve data records

Committing to a list (of transactions)

Alice has $S = (m_1, m_2, \ldots, m_n)$

32 bytes

Goal:

- Alice posts a short binding commitment to S, $h = \text{commit}(S)$
- Bob reads h. Given $(m_i,\ \text{proof}\ \pi_i)$ can check that $S[i] = m_i$

Bob runs $\text{verify}(h, i, m_i, \pi_i) \rightarrow$ accept/reject

security: adv. cannot find (S, i, m, π) s.t. $m \neq S[i]$ and

$\text{verify}(h, i, m, \pi) = \text{accept}$ where $h = \text{commit}(S)$

Cryptographic Hash Function

Extra Property:

Specialized one way function

Canonical application:

Puzzle generation
mining process

Hash(nonce, block-hash) < Threshold

Blockchain: a linked list via hash pointers

Block: Header + Data

Header: Pointer to previous block
= hash of the previous block

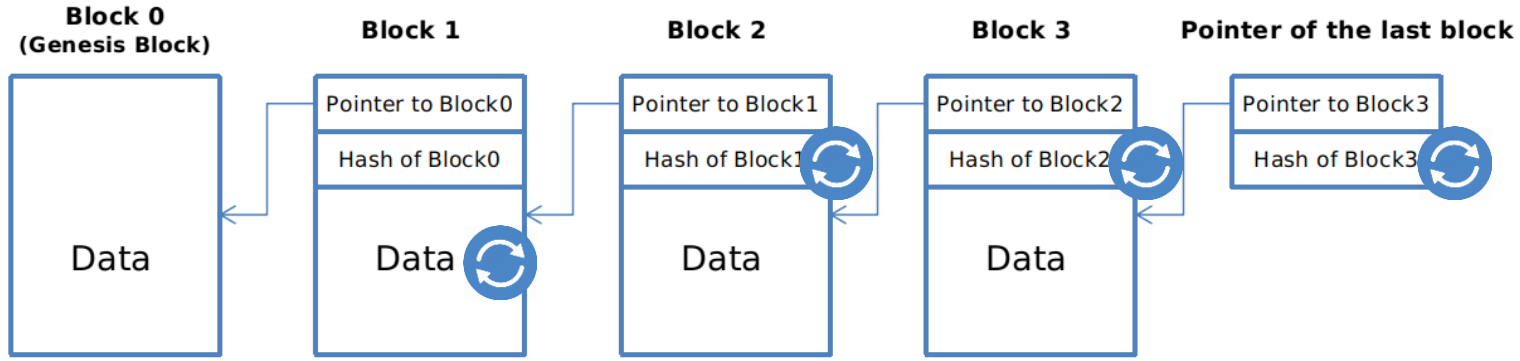
Data: information specific to the block

Application: tamper evident information log

Head of the chain being known is enough to find tamper evidence in any internal block

Hence the phrase: **block chain**
Or simply: **blockchain**

Blockchain: a linked list via hash pointers



Allows the creation of a tamper-evident information log

How about searching for specific data elements?

Merkle Tree

Binary tree of hash pointers

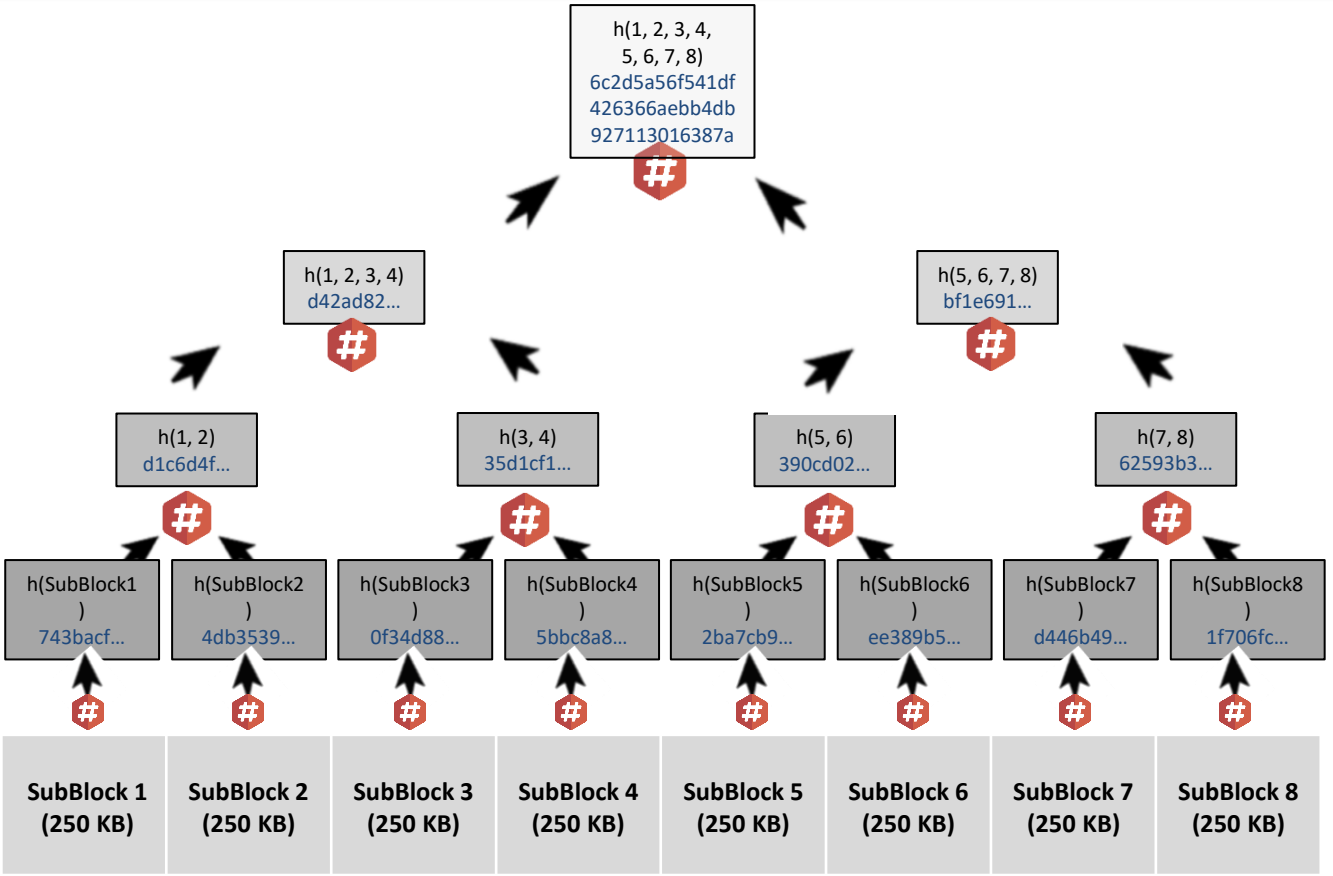
Retain only the root of the tree

Tamper of any data in the bottom of the tree is evident

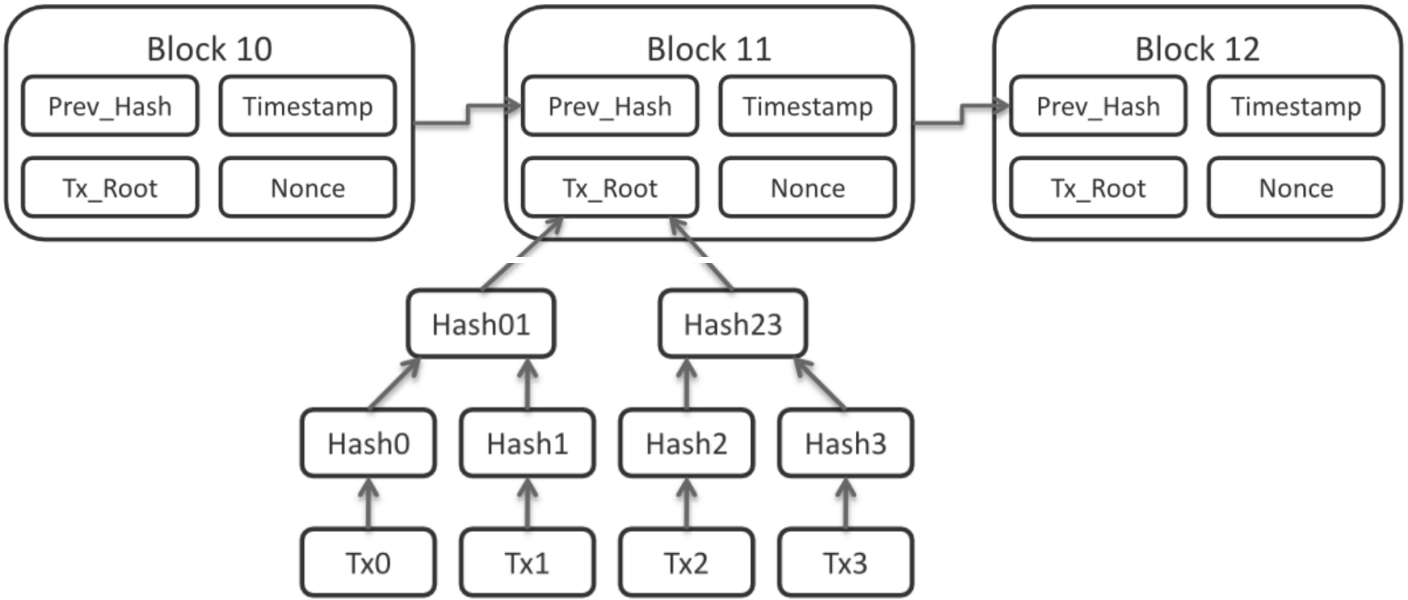
Proof of Membership

Proof of Non-membership

Merkle Tree

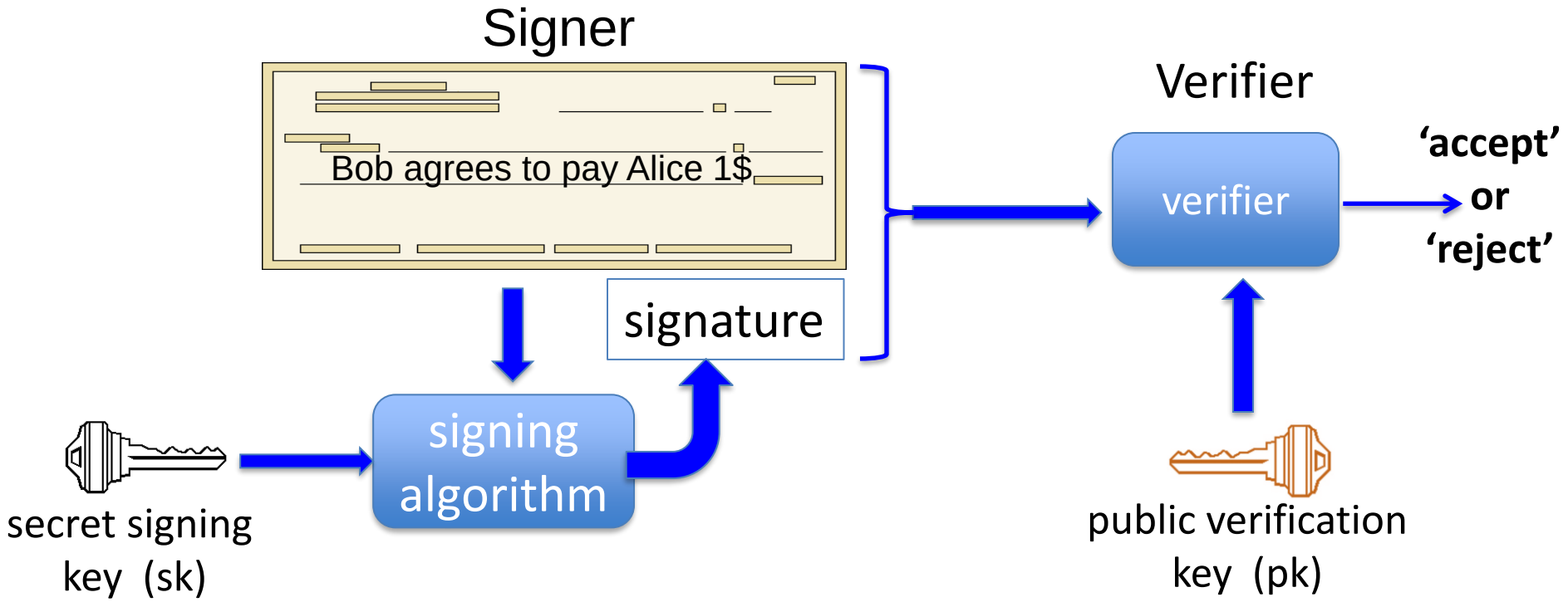

h(1, 2, 3, 4, 5, 6, 7, 8)
6c2d5a56f541df426366aebb4db927113016387a
h(1, 2, 3, 4)
d42ad82...
h(5, 6, 7, 8)
bf1e691...
h(1, 2)
d1c6d4f...
h(3, 4)
35d1cf1...
h(5, 6)
390cd02...
h(7, 8)
62593b3...
h(SubBlock1)
743bacf...
h(SubBlock2)
4db3539...
h(SubBlock3)
0f34d88...
h(SubBlock4)
5bbc8a8...
h(SubBlock5)
2ba7cb9...
h(SubBlock6)
ee389b5...
h(SubBlock7)
d446b49...
h(SubBlock8)
1f706fc...
SubBlock 1 (250 KB)
SubBlock 2 (250 KB)
SubBlock 3 (250 KB)
SubBlock 4 (250 KB)
SubBlock 5 (250 KB)
SubBlock 6 (250 KB)
SubBlock 7 (250 KB)
SubBlock 8 (250 KB)

Blockchain with Merkle Trees



Digital signatures

Solution: make signature depend on document



Signatures in Practice

Elliptic Curve Digital Signature Algorithm
(ECDSA)
Standard part of crypto libraries

Public key: 512 bits

Secret key: 256 bits

Message: 256 bits
Note: can sign hash of message

Signature: 512 bits

Decentralized Identity Management

Public keys are your identity
    address in Bitcoin terminology

Can create multiple identities
    (publickey, secretkey) pairs
    publish publickey
    sign using secretkey

Can create oneself
    verifiable by others

Proof of Work

and

Nakamoto Consensus

Blockchain with Merkle Trees

Block: Header + Data

Header: Pointer to previous block
= hash of the previous block header and Merkle root of data of previous block

Data: information specific to the block

Application: Centralized tamper evident information log with efficient proof of membership of any data entry

Head of the chain being known is enough to find tamper evidence in any internal block

Decentralized Blockchain

Block: Header + Data + Signature

Header: Pointer to previous block
= hash of the previous block header and Merkle root of data of previous block

Data: information specific to the block

Signature: one of the users signs the block (header+data)

List of signatures known ahead of time: **permissioned** blockchains

Questions:

1. How is this list known ahead of time?
2. Which user in this list gets to add which block?
3. Who polices this?

Distributed Consensus

Question: Who maintains the ledger of transactions?

Distributed Consensus

- Interactive Protocol
  - Allows distributed non-trusting nodes to come to agreement
- Traditional area of computer science (Byzantine Fault Tolerance)

Bitcoin's consensus protocol is vastly different

- decentralized identity (permissionless setting)
- less pessimistic network assumptions

Decentralized Identity

Public keys are used as **identity**

Single entity can create vast number of identities

- Sybil
- Cannot do majority or super-majority voting

Network Assumption

Any node can **broadcast** to **all** nodes into the network
fully connected network

Every broadcast message **reaches every** node
albeit with some delay
Bitcoin: ten minutes

Leader Election: Oracle

Time is organized into **slots**

Oracle selects one of the nodes (public identities)

- random
- everyone can verify the unique *winner*

The selected node is the proposer in that slot

- constitutes a block with transactions
- validates transactions
- includes hash pointer to previous block
- signs the block

Proof of Work

Practical method to simulate the Oracle

Mining

cryptographic hash function creates computational puzzle

Hash(nonce, block-hash) < Threshold

nonce is the proof of work

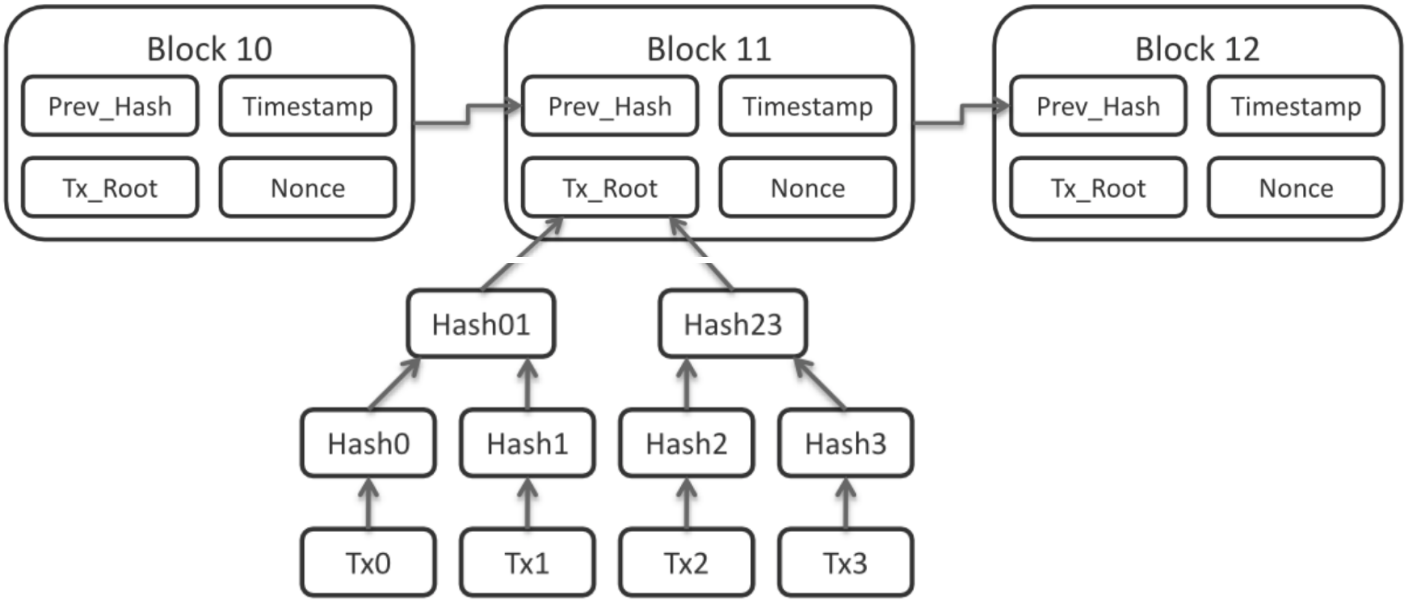
include nonce inside the block

Threshold

chosen such that a block is mined successfully on average once in 10 minutes

a successfully mined block will be broadcast to all nodes in the network

Block Constituents



Properties of Proof of Work Mining

1. Random miner selected at each time

2. Independent randomness across time and across miners

3. Probability of successful mining proportional to fraction of total hash power

4. Sybil resistance

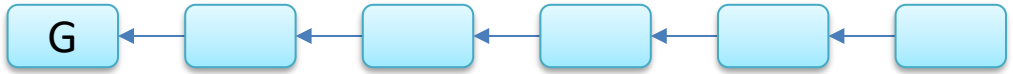
5. Spam resistance

6. Tamper proof – even by the proposer!

Longest Chain Protocol

Where should the mined block hash-point to?

The latest block



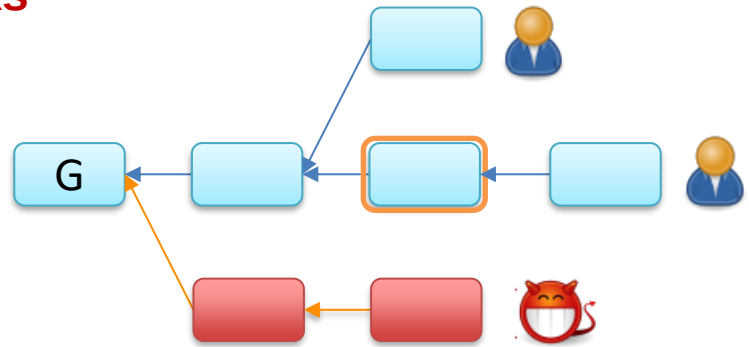
Longest Chain Protocol

Where should the mined block hash-point to?

However, blockchain may have **forks**

because of network delays

because of adversarial action



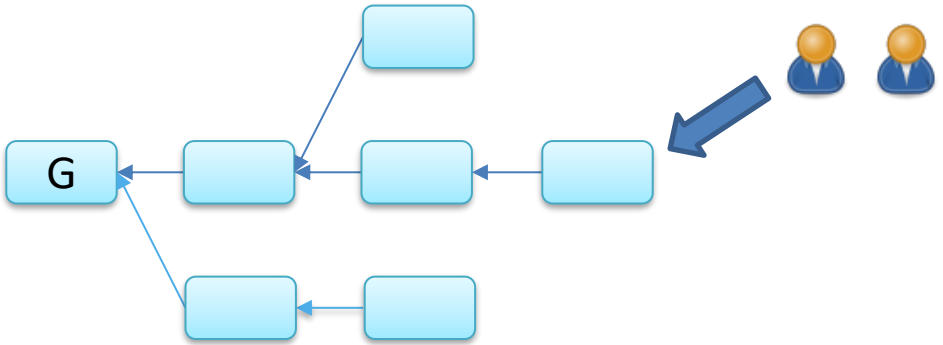
Longest Chain Protocol

Where should the mined block hash-point to?

Blockchain may have **forks**

because of network delays

because of adversarial action



Longest chain protocol

attach the block to the leaf of the longest chain in the block tree

Why Variable Difficulty

Block Difficulty

Example: in September 2022 the mining target or threshold (in hexadecimal) is:

0x 0000000000000000008c8940000000000000000000000000000000000000

19 leading zeros

The hash of any valid block must be below this value ~ 8/16 . 16^{-19} = 2^{-77}

Difficulty of a block:

Block_difficulty = 1/mining_target

Bitcoin Rule

(a) The mining difficulty changes every 2016 blocks

next_difficulty = (previous_difficulty * 2016 * 10 minutes) / (time to mine last 2016 blocks)

(b) Adopt the heaviest chain instead of the longest chain

chain_difficulty = sum of block_difficulty

(c) Allow the difficulty to be adjusted only mildly every epoch

¼ < next_difficulty/previous_difficulty < 4

Resources

- ECE/COS 470, Pramod Viswanath, Princeton 2024
- CS251, Dan Boneh, Stanford 2023