CSE 3550/5000: Blockchain Technology

Lecture 2
Blockchain Semantics and Cryptographic Primitives Review

Ghada Almashaqbeh
UConn - Spring 2026

Outline

- Semantics of blockchain-based systems.
    - Basic components.
    - Basic properties.
- Cryptographic primitives overview.
    - Hash functions.
    - Hash pointers.
    - Linked lists with hash pointers.
    - Merkle trees.
    - Digital signatures.

Semantics of Blockchain-based Systems

Mainly we will focus on public permissionless blockchains, later we will discuss other types of blockchains and how they differ from permissionless ones.

Basic Components I

- Participants.
    - Two types:
        - clients who are only interested in using the service/system,
        - miners who maintain the blockchain and implement the network protocol.
- Peer-to-peer (P2P) network.
    - Connectivity between the participants.
    - No pre-deployed infrastructure, any party searches for peers to connect with in order to join the network.
    - To bootstrap, seed servers are hardcoded in the network protocol.
        - Return a randomly selected set of miners to connect with.

Basic Components II

- **Blockchain**.
    - A log of all transactions happened so far in the system (scalability?).
    - A linked list of records or blocks (backward connections, each block points to its parent).
    - Connections between blocks are hard to alter (the older the block, the harder to alter its backward connection).
    - Starts with a genesis block; first page of the ledger.
    - Each miner stores a copy of the blockchain representing its view of the system.

Basic Components III

- **Mining protocol.**
  - The process of recording new transactions on the blockchain.
  - Main task of the miners.
  - Each round a new block is mined and appended to the blockchain.
  - Requires proving ownership of mining rights (or power).
- **Consensus protocol.**
  - Rules to agree on the current view of the system (represented by its blockchain).

Basic Components IV

- **Network protocol and transaction processing rules.**
  - Rules to validate and process transactions.
    - Transaction processing changes the system state (like balance adjustment, data posting, etc.).
    - Transactions could be simple currency transfer or more sophisticated functionality like multisignature escrows or user-defined scripts/programs.
  - Rules to handle specific actions based on the type of service provided (e.g., creating escrows, paying for file storage, etc.)
  - Other system-specific rules (e.g., connect to the system, solving disputes, etc.).

Basic Components V

- **Incentives.**
  - Several forms:
    - Explicit in the form of tokens or coins (mainly in cryptocurrency-based systems).
    - Implicit in the form of the desire to achieve a common goal (e.g., public good).
- **Community.**
  - The participants themselves (more like stakeholders).
  - Some specialized entities like foundations and code developers.
  - Outsider entities like research community (academia and industry).
  - The value of the system stems from the beliefs of its community.

Basic Properties and Assumptions I

- Open access.
  - No real identities are required.
- Decentralization.
  - Log storage: distributed among all miners.
  - Log extension: anyone can join as a miner and work on extending the blockchain.
  - Consensus: agreeing on the state of the system is done in a fully decentralized way. No centralized entity to orchestrate that.
  - Governance: automated through the network protocol implemented by the miners.
  - Code: anyone can work on the code (according to a specific procedure), and the code is usually open source.

Basic Properties and Assumptions II

- Public verifiability.
    - Anyone can verify the status of the system based on the actions logged on the blockchain.
- Log immutability.
    - Hard to alter the blockchain. Requires a large mining power in order to do so.
- Honest majority.
    - An assumption necessary for the security of the system.
    - Simply states that the majority of the mining power is in the hands of honest miners.

Basic Properties and Assumptions III

- Incentive compatibility.
  - Participants are rational and self-interested.
  - Compatibility means that the participants’ self-interest, or incentives, are aligned with the system goals.
    - For example, miners with high stake are interested in keeping the system up and running (to attract more users and reserve the value of their stake). So they act honestly since this is in their best interest.
- Probabilistic polynomial time (PPT) adversaries. How about post quantum security?
- Eclipse attacks (at a large scale for a long period) are not viable.

Cryptographic Primitives Overview

Hash Functions I

- Mathematical functions that take input of any size and produces a fixed size output.
    - Compress the input to produce a shorter output.
    - Usually the output is called a digest.
- They are also called one-way functions where computing the hash of some string is easy but reversing the hash to recover the input is hard.
    - Technically: computing the image of some input is efficient (done in polynomial time of O(n) where n is the input size), while computing the preimage is inefficient (non-polynomial or computationally hard).
        - So in some sense the output does not leak any information about the input.

Hash Functions II

- Security properties:
  - First-preimage resistance: given y where y = h(x) it is computationally hard to find x.
  - Second-preimage resistance: given x it is computationally hard to find x’ such that h(x) = h(x’).
  - Collision resistance: it is computationally hard to find any x, x’ such that h(x) = h(x’).
- Applications:
  - Message authentication codes.
  - Commitment schemes.
  - Hash tables.
  - Password hashing.
  - Merkle trees.

Hash Functions III

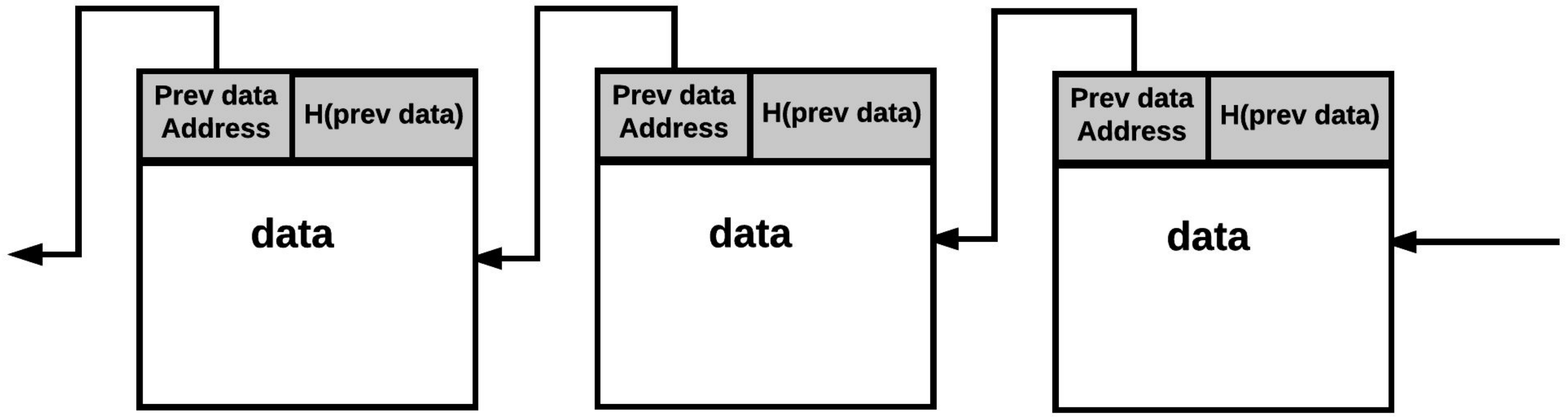
- In theory, hash functions are keyed functions.
    - Meaning that the hash function is parameterized by a key (public, known to everyone). Changing the key changes the mapping (or the output).
- In practice, hash functions are unkeyed (for the same input they always produce the same output).
    - Examples: SHA1, SHA2, SHA3, etc.
- It is so common when analyzing security to model hash functions as random oracles.
    - Called the random oracle model (ROM).
    - Basically means that a hash function acts as a random function that chooses a digest completely at random.

Hash Pointers

- A hash pointer is a pointer that provides a way to retrieve some information as well as to verify that this information has not changed.
    - It has both the address (or location) of the data and a hash of that data.
    - correctness of the data can be verified by computing its hash and compare this hash to the one stored inside the pointer.

Linked Lists with Hash Pointers

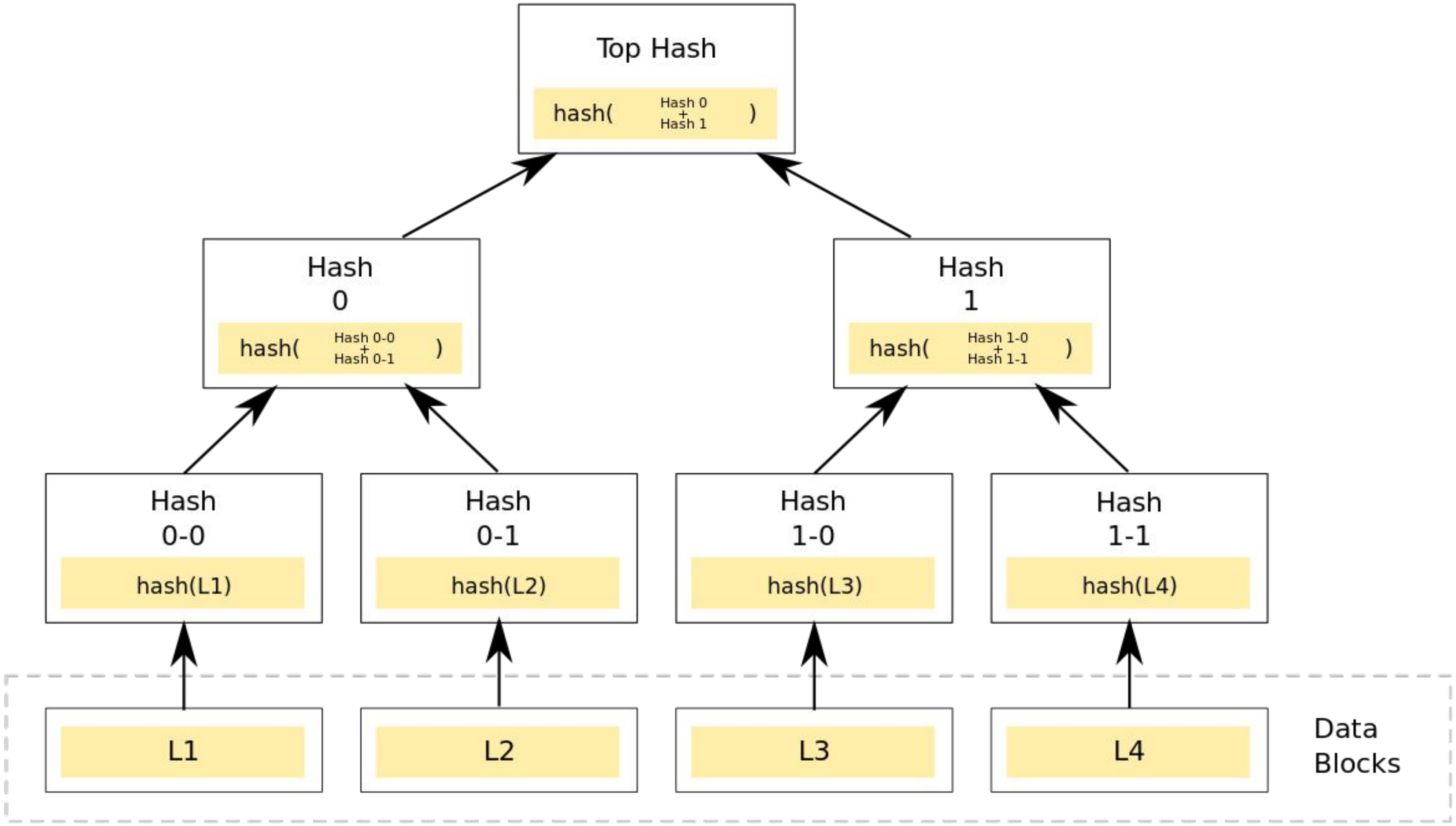
- A linked list that uses hash pointers instead of normal pointers.
- Useful for tamper detection, can you see how?
  - Editing the data of one node breaks the chain.
- Blockchains are simply hash pointer-based linked lists.



Merkle Tree

- An efficient solution for data authentication.
  - Usually used in distributed systems and file storage.
- It is a binary tree with leaf nodes L1, …, Ln (these are the inputs) and depth log n.
- Each internal node in the tree is the hash of its two children.
- The root digest represents the hash of all leaf nodes.
- Verifying the correctness of an input (aka authenticating it) is done by providing a membership path of size log n digests.
  - It requires log n digests and log n hash invocations.

Merkle Tree Pictorially



* https://en.wikipedia.org/wiki/Merkle_tree

Digital Signatures I

- Used to provide two security goals: message integrity and non-repudiation.
  - Integrity: prevent message tampering.
  - Non-repudiation: bind the sender (or message originator) to the message.
- Work in the public-key setting.
  - Each party creates a key pair; public and private key.
  - She uses the private key to sign a message, while anyone can verify the signature on that message using the public key.
  - The public key usually serves as an ID for the user in a blockchain-based system.

Digital Signatures II

- A digital signature scheme consists of three algorithms:
  - Gen: generate a key pair.
  - Sign: sign a message using the private key (usually randomized).
  - Verify: verify the signature over the signed message (deterministic).
- Informally, a secure digital signature scheme means that an attacker cannot generate a valid signature over a message without knowing the secret key even if it obtains signatures on other messages.
  - It cannot forge a signature.
- Most practical digital signatures constructions are computationally secure.
  - It is computationally hard to forge a signature unless you know the secret key.
- Usually the hash-sign paradigm is used, where the message is hashed first and then the digest is signed.