# CSE 2550/5000: Blockchain Technology

## Lecture 9
## Blockchain Types

**Ghada Almashaqbeh**
UConn - Spring 2026

# Outline

- Types of blockchains:
  - Permissionless public blockchains.
  - Permissionless private blockchains.
  - Permissioned blockchains.
  - Mutable blockchains.

# Permissionless Public Blockchains

- Most popular since it is the first blockchain type to witness huge adoption in practice. E.g., blockchains of Bitcoin, Ethereum, etc.
- Permissioless; open to anyone to participate as a miner or as a client.
  - No real identities, or any form of authentication/authorization, is required.
- Public; block content is public.
- Open source code; anyone can download, inspect, and suggest modifications.
- High overhead since all miners have to verify all transactions and blocks.

# Permissionless Private Blockchains I

- Still permissionless with open source code.
- Difference is that the blockchain content is confidential (e.g., encrypted).
  - Thus, inspecting a block content does not reveal anything unless one knows the secret information (e.g., key) used to seal the data.
  - Correctness/privacy is enforced using advanced cryptographic primitives.
    - Zero knowledge proofs, ring signatures, homomorphic commitment/encryption schemes, etc.
- Examples include blockchains of Zcash and Monero.

# Permissionless Private Blockchains II

- Usually offer three modes of operation:
  - Public transactions,
  - private with transaction confidentiality only,
  - and private with both confidentiality and anonymity.
  - Users get to choose which mode to use on per transaction basis.
- Use similar techniques of extending and maintaining the blockchain as in permissionless public blockchains.
  - But verification is now more expensive due to the involvement of advanced cryptographic primitives.

# Permissioned Blockchains I

- Sometimes called federated or enterprise blockchains.
- Joining the system is NOT open for anyone.
  - The stakeholders, or miners, are an approved set of nodes.
  - Access for clients who want to use the service may vary;
    - Usually only authorized clients are allowed to use the system, e.g., customers of a particular bank.
- In such an environment, respecting information privacy and regulation compliance is a key.
- Extending the blockchain is usually done using a BFT-based consensus protocol.

# Permissioned Blockchains II

- Write permission; only the approved set of miners/validators is allowed to extend the blockchain.
- Read permission may vary;
  - Public, anyone can access the stored records.
  - Permissioned, specific audience (e.g. all employees within one organization).
- Main use cases center around banking systems.
  - A group of financial institutions work collectively together and share data/ensure accountability through the common blockchain.
    - Main motivations are reducing cost and speeding up service processing that span multiple institutions.

# Permissioned Blockchains - Examples

- Hyperledger Fabric and Corda.
- Mostly follow the general theme of Ethereum.
  - Utilize smart contracts to allow customized processing of data, automated deals and financial agreements.
  - Such contracts could be accompanied with legal prose to enforce compliance.
    - Corda adopts this model. Each deal has a state object composed of a smart contract and legal prose.
  - Although Corda supports smart contracts, it adopts a UTXO model not an account model.
  - Hyperledger Fabric adopts the account model.

# Permissioned Blockchains - Ordering I

- Several permission blockchains separate transaction validation/execution from ordering and have these tasks implemented by different nodes.
- Corda involves notary that decides the order of transactions that consume the same state—it does not establish global ordering of transactions!
  - In other words, this is done to prevent double spending or to resolve conflicting transactions.
  - Clients send transactions to the notary, if all is good, the notary endorses (signs) the transaction.
  - If a notary pool is used, so having several notaries instead of one, BFT is used to let them all agree on the decision.
  - After being endorsed, clients proceed as usual by distributing their tarsnacions to the validators for execution and finalization.

# Permissioned Blockchains - Ordering I

- Hyperledger Fabric adopts the model of endorse, order, and execute transactions—establishes a global ordering of transactions!
  - Endorse is basically validating and simulating transaction execution locally (i.e., no ledger state changes) to determine the impact of the transaction. The validator signs the simulated output and send it back to the client who issued the transaction.
  - After collecting enough endorsements, the client sends the transaction to the ordering service.
  - Finally, transactions in a block are executed (produce ledger state updates).
  - Advantages: scalability and deterministic smart contract execution (no transaction ordering dependency).

# Permissioned Blockchains - Privacy

- Respecting data privacy can be enforced by partitioning the blockchain state into several layers.
  - In Hyperledger Fabric these are called channels.
    - A channel is a blockchain on its own.
  - Each channel has its own ordering of transactions and may have different set of members.
    - Thus, members see different data and views based on which channel they belong to.

# Mutable or Redactable Blockchains

- Developed to address criticism that immutability of the blockchain, which is a security requirement, has several disadvantages:
  - How about the right to be forgotten?
  - How to remove inappropriate content?
  - Can we compress blocks in the blockchain?
    - Cut stale history that is no longer needed to improve scalability and reduce storage footprint.
  - Can we enable editable storage for smart contracts?
    - Being able to change its logic or patch security vulnerabilities without deploying a new version of the contract.

# Mutable Blockchains - An Example

- One scheme proposed in [Ateniese et al., 2017].
- It is based on using a hash function with a trapdoor that allows finding collisions.
  - That is, this hash function has a secret key that can be used to make a modified block produce the same hash as the old block.
  - One such primitive is Chameleon hash functions.
- By doing so, blocks in the blockchain can be rewritten without breaking the chain.
- Without the trapdoor, no edits can be made.
- Edits are public and auditable by other miners.
  - since they must approve the new blockchain version and the needs/justifications of the edits made.

# A Modified Chameleon Hash Function

- Consists of a tuple of four algorithms:
  - **Key generation:** generates a public key $pk$ (used for hashing) and a trapdoor key $sk$ (used for finding collisions).
  - **Hash:** Uses the public key to hash a message $m$. Outputs the hash $h$ of $m$ and some string $w$ needed to verify correctness of the hash (usually called a witness).
  - **Verify:** takes $m, w, h$, and $pk$ as inputs and returns 1 or 0 based on whether the hash $h$ is correct or not.
  - **Find collision:** takes the trapdoor key $sk, m, h, w$, and a new message $m'$, and produces $w'$ such that $Verify(pk, m', h, w') = 1$.
    - The hash is not changed!

# Who Can Edit

- Centralized setup:
  - One entity knows the trapdoor, and hence, can edit.
  - Undesired as it conflicts with the basic concept of blockchains—decentralization.
- Distributed setup:
  - Replace the trusted party with several parties.
  - Generate the trapdoor key in a distributed way (utilize threshold secret sharing to collectively generate random shares of the trapdoor).
    - Each party will have a share of the key.
    - The key is not known to any single party.
  - At least *t* parties are needed to collectively compute a collision.
  - The assumption is that the adversary cannot corrupt more than *t-1* parties.

# References


- [Ateniese et al., 2017] Ateniese G, Magri B, Venturi D, Andrade E. Redactable blockchain–or–rewriting history in bitcoin and friends. In 2017 IEEE European Symposium on Security and Privacy (EuroS&P) 2017 Apr 26 (pp. 111-126).