CSE 5095-007: Blockchain Technology

Lecture 10
Blockchain Types

Ghada Almashaqbeh
UConn - Fall 2021

Outline

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

Permissionless Public Blockchains

- Popular since it is the first type of blockchains to be used.
  - Examples: 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 are required.
- Public; block content is public, everyone can see the sender/receiver addresses, transaction value, etc.
- Open source code; anyone can download, inspect, and suggest modifications.
- High overhead since all miners have to verify all transactions and blocks in the system.
- Techniques used in maintaining and extending the blockchain may vary:
  - adopted mining algorithm, block generation rate, average block size, types of transactions, etc.

Permissionless Private Blockchains I

- Still permissionless with open source code.
    - Anyone can join to mine or just use the service.
- Difference is that the blockchain content is confidential (like encrypted).
    - Thus, inspecting a block content does not reveal anything unless one knows the secret information 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

- Systems may not hide everything!
  - Provide three modes of operation: public, private with transaction confidentiality only, 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.
  - Respect privacy restrictions of miners when applicable/possible.

Permissioned Blockchains I

- Sometimes called federated or enterprise blockchains.
- Joining the system is NOT open for anyone.
  - In particular the stakeholders or miners; only 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.
- In such an environment, respecting information privacy and regulation compliance is a key.
- Extending the blockchain usually done through a Byzantine fault tolerant agreement protocol.
  - Also proof-of-elapsed time was developed to be used in such permissioned setting.

Permissioned Blockchains II

- Write permission; only the approved set of nodes are allowed to extend the blockchain.
- Read permissions 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.

Permissioned Blockchains - Examples

- Several examples; Hyperledger Fabric, Corda, Quorum, etc.
- Most follow the general theme of Ethereum.
    - Utilize smart contracts to allow customized processing of data and automating deals and financial agreements.
        - Quorum is an enterprise version of Ethereum which replaces its proof-of-work with PBFT consensus.
    - 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.

Permissioned Blockchains - Ordering

- Several systems separate transaction validation/execution from ordering and have these tasks implemented by different nodes.
    - Corda involves a notary pool that decides the order of transactions that consume that same state.
        - If a notary pool is composed of several entities, PBFT is used to let them all agree on the same decision.
    - Hyperledger Fabric adopts the model of execute (or endorse), order, and validate transactions.
        - A group of nodes work together to reach a consensus of the order of transactions.
    - Validating transactions is done independently by each node through executing the smart contract code.

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, have several disadvantages.
  - How about the right to be forgotten?
  - How to remove inappropriate content?
  - Can we compress blocks in the blockchain?
    - Cut the stale history that is no longer needed.
  - Can we enable editable storage for smart contracts?
    - Being able to 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 have the same hash as the old block.
  - Recall that immutability is achieved because of the collision resistance property of the hash function used.
- By doing so, specific blocks in a 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 and have access to its old copies.

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 the correctness of the hash (usually called a witness).
  - Verify: takes m, w, h, and the public key as inputs and return 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.
- Distributed setup:
  - Replace the trusted party with several parties.
  - Utilize multiparty computation protocols (MPC) to generate the trapdoor key in a distributed way (utilize threshold secret sharing).
    - Each party will have a share of the key (some random string).
  - Another MPC protocol allows using these shares, at least t of them, to compute a collision.

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).