Preliminary Notes

This class provides background about blockchain technology. This laboratory is based on previous work elaborated by the contributors [1–3]. We recommend students have attended Lab1 and Lab2.

1 A Primer on Blockchain

We now introduce Blockchain, whereby its building blocks are rooted in Distributed Systems and Cryptography (see Lab01 and Lab02).

1.1 A Technical Viewpoint

The term blockchain is generally referred to as a replicated data structure: a persistent, auditable, immutable, append-only distributed ledger. The Blockchain is maintained by a group of physical machines, nodes (or peers or participants). Nodes are not trusted individually to maintain the distributed ledger; they are trusted as a group due to their number and diversity. As long as nodes have conflicting interests, collusion between them will probably not happen: and thus, the network can be considered safe. Blockchain, in fact, implements many concepts discussed in Lab01. For instance, a byzantine fault-tolerant (or just crash fault-tolerant) replicated state machine agrees on a global state via consensus. The Blockchain is secure due to a combination of decentralization and cryptography: transactions are aggregated in blocks, linked by the cryptographic hash of the previous block. If a participant changes a previous block, its hash will change and the posterior hashes.

Most blockchains serve as a runtime for programs called smart contracts. The execution of smart contracts is originated by transactions. Smart contracts are executed in a virtual machine, e.g., in the Ethereum Virtual Machine (EVM) in Ethereum, one of the most used blockchains. Smart contracts can abstract the representation of currency, resources, assets, access, equity, identity, collectibles via tokens.

There are different types of blockchains, which can typically be categorized into permissionless blockchains and permissioned blockchains.

1.1.1 Permissionless blockchains

Public blockchains can be accessed and utilized by anyone with Internet access. Permissionless blockchains do not require participants to be identified to participate. We use public and permissionless as synonyms. Typically, in such networks, the participants are rewarded for supporting the ecosystem, making it self-sustainable. The first entirely decentralized digital currency, Bitcoin, is a permissionless peer-to-peer version of electronic cash, which would allow transactions to go directly from one party to the other, excluding the need for any intermediary. Bitcoin quickly gained popularity because it solved the famous double-spending problem without the need of any central authority.

Examples: Bitcoin, Ethereum, EOS, Ripple, IOTA.

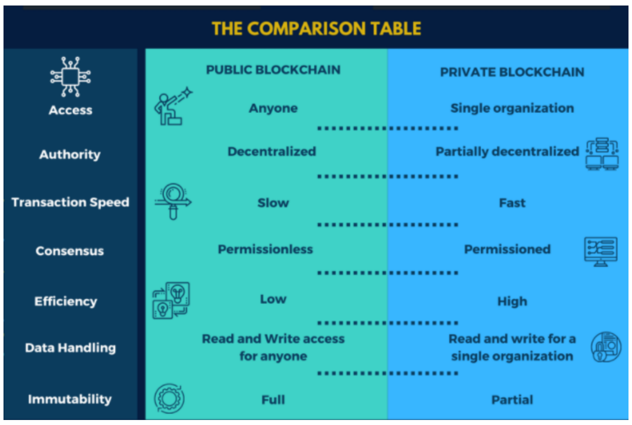
1.1.2 Permissioned blockchains

Permissioned blockchains identify and associate participation right to participants. Private blockchains require credentials for participants to vizualize it. We use private and permissioned as synonyms. Private blockchains are the property of an organization or consortium of organizations. Suppose more than one organization is included in the management of the Blockchain. In that case, it is called a community or federated Blockchain. A permissioned solution seems suitable for companies aiming for competitiveness that blockchain technology can offer while protecting sensitive information. Write permissions are kept centralized to specific nodes within the organization. Therefore, a trade-off between control and decentralization exists. From a technological point of view, there is an aggravated risk of subversion compared to public blockchains. There are fewer participants in the network that validates transactions. Nevertheless, in practice, as the network only allows identified participants, it is easier to identify a culprit.

Examples: Hyperledger Fabric, Hyperledger Indy, Hyperledger Burrow, Hyperledger Sawtooth, R3 Corda, and Multichain.

1.2 Comparison

Public and private blockchains are essentially different in nature. Inside each category, blockchains are also very different: their consensus, networking, data, and application layers can differ substantially, leading to a wide range of choices. This leads to fragmentation, where some blockchains are immature (not stable). While this can be a big problem, research in blockchain interoperability attenuates this [3].



THE COMPARISON TABLE

	PUBLIC BLOCKCHAIN	PRIVATE BLOCKCHAIN
Access	Anyone	Single organization
Authority	Decentralized	Partially decentralized
Transaction Speed	Slow	Fast
Consensus	Permissionless	Permissioned
Efficiency	Low	High
Data Handling	Read and Write access for anyone	Read and write for a single organization
Immutability	Full	Partial

Figure 1: Comparison between public and private blockchain

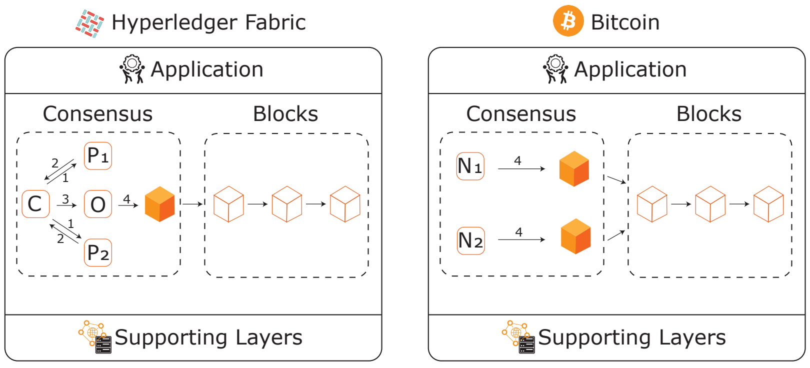


Figure 2: High-level comparison of Hyperledger Fabric and Bitcoin, from [**?**]

Figure 2 depicts two blockchains: Hyperledger Fabric, a permissioned blockchain; and Bitcoin, a permissionless

In Hyperledger Fabric, the consensus is based on endorsement policies. In Fabric, a client (C) sends a transaction proposal to the peer nodes (P), and obtains a signed transaction, called an endorsement (steps 1 and 2). An orderer validates the endorsements and builds a block with valid transactions, appending it to the ledger (steps 3 and 4). In Bitcoin, the consensus is based on the notion of Proof-of-Work (PoW), a cryptographic puzzle that mining nodes need to solve to build a valid block. This corresponds roughly to Fabric's steps 1-3. After a node finds a solution to PoW, it then can propose a block of transactions to be appended to the ledger (step 4).

2 Hands on Blockchain

A very reliable way to learn Blockchain is to build one. Blockchain4Students is a blockchain[1] written in the Python programming language specifically tailored to be easily understandable and thus be a reliable basis for blockchain teaching. Let's dive deeper into the blockchain world by exploring Blockchain4Students, and then answering to the following exercises.

[1]https://github.com/hyperledger-labs/university-course/tree/master/support/Lab03%20-%20Blockchain4Students

2.1 Theory & Business

Exercise 1: What is the advantage of the Blockchain being a decentralized, Peer-to-Peer system instead of a centralized one?

Exercise 2: Consider a private blockchain running three nodes. Given that it is managed from a single organization, how do we assure decentralization?

Exercise 3: What stops a malicious entity from being able to alter blocks in a permissionless blockchain?

Exercise 4: What about permissioned blockchains? Can a malicious entity change in these as well?

Exercise 5: How does a client insert a transaction in the system?

Exercise 6: How do we know if a transaction was created by a client and not by a malicious node?

Exercise 7: Why can't permissionless blockchains use the classical Consensus algorithms and must instead rely on other forms of Consensus like Proof-Of-Work?

Exercise 8: Why can't a node fake a result from Proof-of-Work?

Exercise 9: Why do we want to add difficulty to the Proof-of-Work?

Exercise 10: How does the difficulty value work in Proof-of-Work?

Exercise 11: How does a node share its new block?

Exercise 12: What are the advantages and disadvantages of sharing the chain by broadcasting the block or by verifying other chains from peer nodes?

Exercise 13: What happens when multiple nodes mine a new block at the same time?

Exercise 14: Why do the miners choose the longest chain to work and not a forked, smaller chain?

Exercise 15: What happens to the other block and transactions that were created in the forked chain that was discontinued?

Exercise 16: What if the service was already made and the payment (in the form of a transaction) disappears?

Exercise 17: Being a Peer-to-Peer service, how does a node discover which nodes to connect to in order to join the system?

2.2 Blockchain4Students

Consider the source code of Blockchain4Students. Start by exploring “blockchain-data-structure” and “‘consensus”, inside the blockchain folder.

Exercise 1: Consider the next code fragment. What type is the Blockchain4Students blockchain?

```
app.route('/register/node', methods=['POST'])
f register_peer_node():
de_address = request.get_json()["node_address"]

sponse = {
essage': 'Node added',
otal_nodes': list(blockchain.peer_nodes),

ockchain.register_node(node_address)
turn json.dumps(response), 200

```

Exercise 2: We give rewards right after a block is mined. What is the problem with this approach?

Exercise 3: Each Transaction includes the ID from the node that created it. Why is this information necessary?

Exercise 4: When a transaction request arrives at the node, it signs it with its own private key. What is the possible attack with this approach?

Exercise 5: When a node first connects, it contacts a discovery node to find other peers in the Blockchain. Although this method is possible, why is it not realistic to use on broader scale blockchains?

Exercise 6: When the node obtains a valid chain bigger than its own, it discards its own chain for the bigger one. Why?

Exercise 7: With our current implementation, when a node joins, he is not aware of any state of its peers. Add the functionality of when a node joins the system to ask and update its own Blockchain.

References

[1] R. Belchior. *JusticeChain: Using Blockchain To Protect Justice Data*. PhD thesis, Instituto Superior Técnico, dec 2019.

[2] R. Belchior, A. Vasconcelos, and M. Correia. Towards Secure, Decentralized, and Automatic Audits with Blockchain. In *European Conference on Information Systems*, 2020.

[3] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *arXiv*, 1(1):58, may 2020.