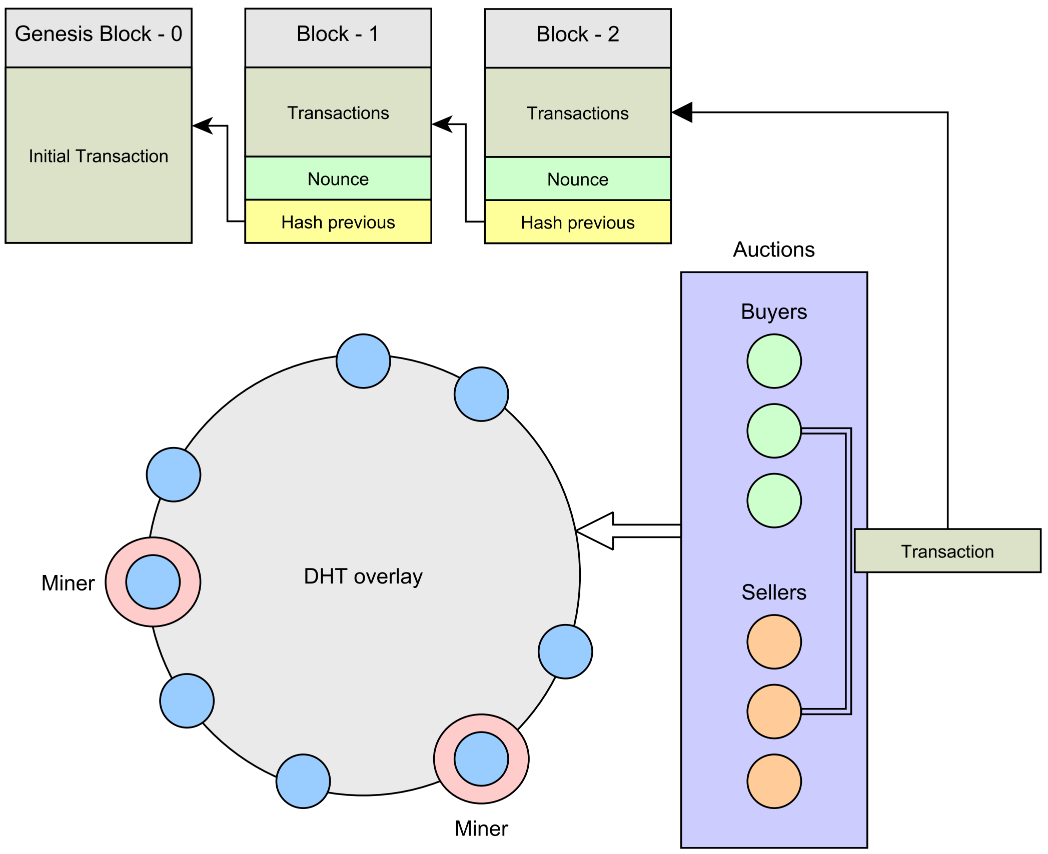
Assignment: Public Ledger for Auctions

System and Data Security 20/21

February 15, 2021

This assignment requires the implementation of a public blockchain (non-permissioned) [1, 2], but opposite to Bitcoin and Etherium, the purpose here is to have a decentralized public ledger capable of storing auction transactions. All code must be written in Rust or Java.



The work is divided into 3 parts, **distributed ledger**, **secure P2P** and **auction mechanisms**:

- The secure ledger should be modular, and it must support PoW as the core consensus:
  - using proof-of-work [1] (grade 4/20) (4pt) and,
  - Bonus: proof-of-stake [2] (2pt).

  - Bonus: Present an architecture (no implementation is required!) for implementing a permissioned blockchain that uses Byzantine Fault-Tolerance (BFT) [3] as its core distributed consensus (hint: start by looking at BFT-SMaRt [4]) (extra grade 1pt).

- A P2P layer to gossip the necessary data to support the blockchain that has to include:
  - must implement Kademlia [5, 6] (grade 9/20) (5pt).
  - Resistance to Sybil and Eclipse attacks (grade 13/20) (4pt).
  - Implement trust mechanisms [7] (grade 14/20) (1pt).

- An auction system capable of supporting sellers and buyers using a single attribute auction following the English auction (18/20)(4pt):
  - Transactions should be saved in the blockchain (using public key crypto)
  - A publisher/subscriber should be built on top of Kademlia to suport auctions [8].

- Report with a maximum of 10 pages (with unlimited pages for references), A4, font size 11, using latex (grade 20/20) (2pt).

- The architecture must be clearly presented, with the design choices being driven by the functional requirements(presented above). It also must present the assumptions made, namely mandated by theoretical and practical limitations.

Limitations - Warnings! You can only use low level libraries to help you, such as Netty for communications, or BouncyCastle for crypto. You can not reuse existing open-source projects to do this assignment, namely for proof-of-* and P2P. In doubt, ask me.

The expected outcomes are the following:

- Design and implementation of the aforementioned system, with security as a first class citizen. This means that identity, authorization, authentication, access control and trust/authoritative domains must be addressed. Beware of the communications channels and crypto you use; be sure they fit your needs. The codebase must be hosted in a private repository in Github or Bitbucket.

Hint: Use the IntelliJ IDEs for development (https://www.jetbrains.com/idea/). They are free for students. Scholar is your friend. Use it for exploring related work https://scholar.google.com

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[2] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[3] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[4] João Sousa, Eduardo Alchieri, and Alysson Bessani. State machine replication for the masses with bft-smart. 2013.

[5] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

[6] Ingmar Baumgart and Sebastian Mies. S/kademlia: A practicable approach towards secure key-based routing. In *Parallel and Distributed Systems, 2007 International Conference on*, pages 1–8. IEEE, 2007.

[7] Francis N. Nwebonyi, Rolando Martins, and Manuel E. Correia. Reputation based approach for improved fairness and robustness in p2p protocols. *Peer-to-Peer Networking and Applications*, Dec 2018.

[8] Bittorrent publish/subscribe protocol. `http://bittorrent.org/beps/bep_0050.html`. Accessed: 2021-02-15.