TÉCNICO LISBOA

Highly Dependable Systems

2022/2023

Grupo 12:

Guilherme Santos, nº 96740

José Santos, nº 96750

Tomás Oficial, nº101259

1. Used Technologies

In our project, we used Java 17 as our programming language and as the source for our cryptographic primitives. We used maven for build automation.

As we were supposed to simulate the abstraction of fair-loss links, we used UDP sockets for the communication between the components of our project.

2. System Components and Abstractions

Our system is composed of two main parts, the servers who run and maintain the blockchain, and the users who send requests to the servers' leader in order to add a string to the blockchain. The set of servers is previously known by all the participants in the system and we assume that the leader is always correct and never changes.

We implemented perfect links in the communication between servers, in order to assure that requests from users and messages exchanged in the consensus algorithm, are always delivered and delivered just once.

We implemented a simplified version of the Istanbul BFT Consensus algorithm to be used by the set of servers to choose the next string to append to the blockchain. Since in this stage we assume that the set of servers is previously known by all the participants in the system and we assume that the leader is always correct and never changes, we didn't implement the rounds mechanism described in the algorithm, but it was our concern to develop a simplified version of the algorithm that was easy to change and to add the rounds mechanism to it in a later phase of this project. Our algorithm supports concurrent consensus instances; whenever the leader receives a request from a user, it initiates a consensus instance with some consensus id and starts sending "PREPARES" messages to the other servers. To prevent that some servers could decide to append some strings in a different order than other servers, we added a mechanism to the algorithm that forces a given server to not append a string to the blockchain unless all the strings of consensus instances with lower ids were already decided and appended to the blockchain, or aborted. When a string is added to the blockchain, the leader informs the user who made that request whether his request was appended or not.

In the communication between servers, we send the messages with a specific format we defined, concatenated with a digital signature, in order to assure integrity, authentication, and non-repudiation. The message goes in plain text.

In the communication between the servers' leader and the users, we send the messages with a specific format we defined, concatenated with an HMAC, in order to

assure integrity and authentication. In order to build the HMAC, a symmetric key is exchanged between the servers' leader and some user when that user is initiated, the leader sends a symmetric key and an iv, encrypted with the user's public key, so only he can access it. .

We did not care about the confidentiality of the messages since no sensible information is being exchanged and the ledgers are often public.

3.Guarantees

- Dropping Messages Prevention:
    - The messages exchanged between servers are always delivered due to the perfect links.
- Replay Attacks Resistant:
    - o The message ID is always appended to every message. The Signature or HMAC contains this ID so if the message is changed the new HMAC/Signature will not be verified.
- Authenticity:
    - o Signatures (RSA-4096) for server communication or HMACs for user-server communication (SHA-256) are generated and appended in every message over its payload.
- Confidentiality:
    - The HMAC of messages is encrypted using the session key generated for the user, so in this way we keep the HMAC for integrity and authentication confidencial.
- Client Concurrency:
    - The Server is implemented using synchronized methods that access the same data.