Highly Dependable Systems

HDS Serenity Ledger - Stage 1

Project Report

Team nr.:	25
99265:	Luís Filipe Pedro Marques
99313:	Ramiro Marques Moldes
105458:	João Pedro Lobo Moreira

MEIC-A

IST-ALAMEDA

2023/2024

Contents

List of Figures

Chapter 1

Motivation

This project was developed to understand the Istanbul BFT (IBFT) Consensus Algorithm more specifically the round change component implementation and it's application on a simple client-server architecture. The system was developed based on a Byzantine fault-tolerant protocol that uses cryptography to achieve a higher level of security.

Chapter 2

System Build

In this section we will explain how the system works as a whole and what was implemented by our group.

2.1 Client

The client wasn't provided in the template, therefore we needed to implement it. It broadcasts a string to all the servers and the message sent is secured using asymmetric key by signing it. The string sent from the client is done by an append command that concatenates the string into the server's ledger.

2.2 Servers

The server was partly implemented in the solution case, being able to send to each other pre-prepare, prepare, commit, acknowledgement, ignore and round change messages. It also can deal with strings sent by the client and use the IBFT Consensus Algorithm [1] to guarantee a certain order received by the servers. Each server is created with the help of a config file where we can read theirs id, port, client port and behavior.

2.3 Links

The connections between servers and servers-clients are made using the UDP protocol. The supplied template had already assured Fair Loss Links and also perfect links,

only missing the authenticated perfect links that was assured by using digital signatures more precisely the RSA asymmetric cryptography, where each server knows the keys of each server, therefore there is no implementation of a key distribution.

The digital signatures works in the follwing way: the process wants to send a message to another server in this case he signs the message with his private-key and sends a message type signed message which contains the message and the signature resulted from the sign action where the digest is encoded in base64 so it can be transported to the others servers. Afterwards, the server that receives the sign message uses the sender's public-key and verifies it's signature.

Authentication is assured since the private-key used to sign, normally is only known by the sender itself. Integrity is also assured since if the message was tampered during transmission the signature's verification would fail even if the receiver was using the right public-key.

Our solution also does signing of the requests made between the client and server which was a feature not asked in the document of the project stage 1.

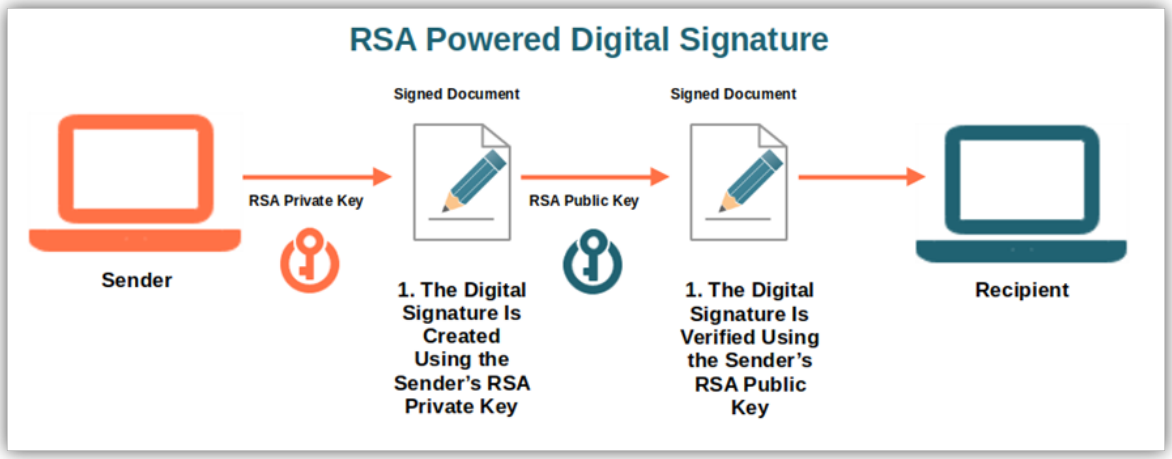

Figure 2.1: Cryptography Diagram

2.4 IBFT Completed

The given template already had the implementation of pre-prepare, prepare, and commit phases that are explained in more depth in the IBFT document [1]. The one aspect left to develop was the round change. The implementation of the round change began with a timer. When the timer expired and no decision had been made, the server will

broadcast a round change message. This message would then be received by all the other servers. Subsequently, when a certain number round change messages (f + 1) received by the servers, they would also broadcast the round change message. Once the leader server received a quorum of round change messages, it would elect a new leader using Rotating Byzantine Leader Election. This method utilizes the server configuration to change the leader by ID and in the order presented in the file.

After the round has changed, the next pre-prepare sent needs to be justified. To achieve this, our solution involves verifying that when a server receives a pre-prepare, the sender's ID is that of the new leader. Additionally, if there was a previous round change quorum, either its prepared value should be null or, if not null, it should have had a quorum of prepared messages. The value presented in those prepared messages should match the prepared value presented in the round change message that achieved quorum. Due to issues with the template, if a round before had already reached a consensus, the next round triggered by a round change would still attempt to send messages to achieve a consensus, even if one had already been decided. To address this, we mitigated the problem by ensuring the system ignores messages once a value has already been decided.

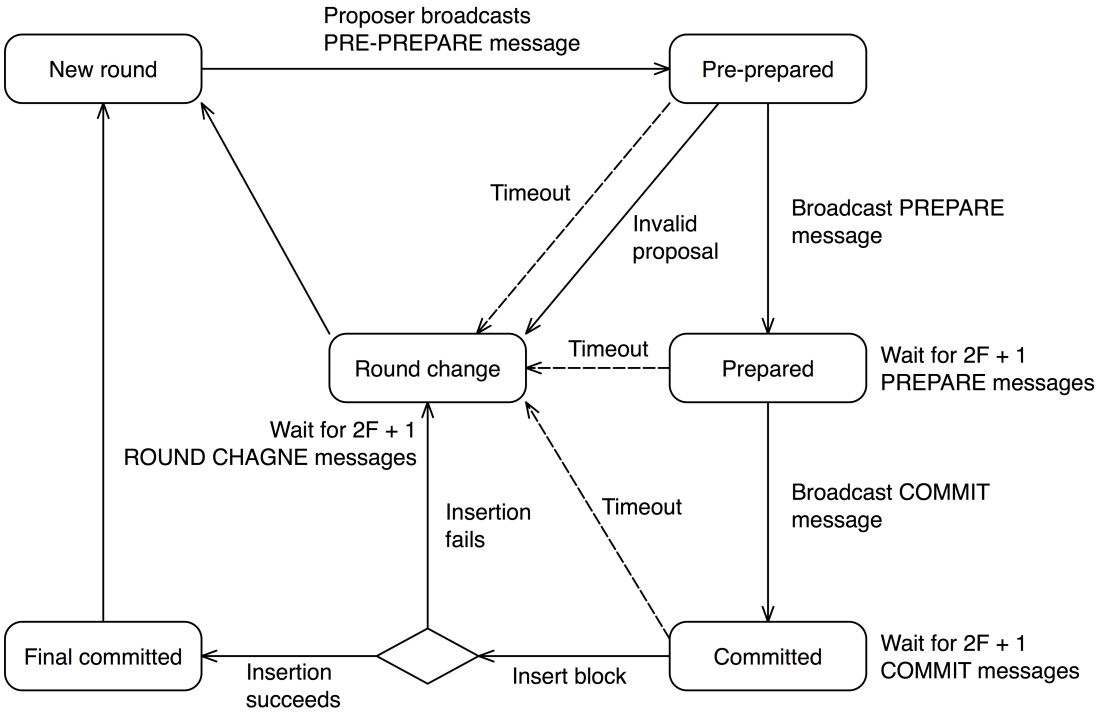

Figure 2.2: Round Change Diagram

2.5 Behaviours

To test if the system was functioning correctly, several tests were made, more specific methods that are called in the node service when a specific behaviour of the node is referred in the configuration file.

The behaviour property of each server can be the following:

- Behaves normal (`NORMAL`)
- The process ignores all the messages received (`NON_RESPONSIVE`)
- Sends PRE-PREPARE messages as a non-leader (`FAKE_PRE_PREPARE`)
- A non-leader process sends COMMIT and PREPARE messages as a leader (`FAKE_LEADER_C_P`)
- A Leader sends PRE-PREPARE message as a non-leader (`LEADER_PRETENDING`)
- A process sends a COMMIT message with a different value (`FAKE_COMMIT`)
- A process sends a PREPARE message with a different value (`FAKE_PREPARE`)
- A process does not broadcast any message sending only to some other process (`BROADCAST_FAIL`)
- A process does not send PREPARE messages if it is on round 1 upon receiving a PRE-PREPARE message (`NO_PREPARE_01`)
- A process that does not verify it's messages (`NO_VERIFICATION`)
- A Leader process starts a consensus with a predefined value (`NO_CLIENT`)

The most important behaviours are `NO_PREPARE_01` and `LEADER_PRETENDING`. `NO_PREPARE_01` it's our method to test the round change since the first round will not progress without PREPARE messages. `LEADER_PRETENDING` tests the messages signatures by making it's verification fail assuring the security, more precisely the authentication and integrity of the messages sent between servers.

Bibliography

[1] H. Moniz, “The instabul bft consensus algorithm,” *Quorum Enginnering*, 2020.