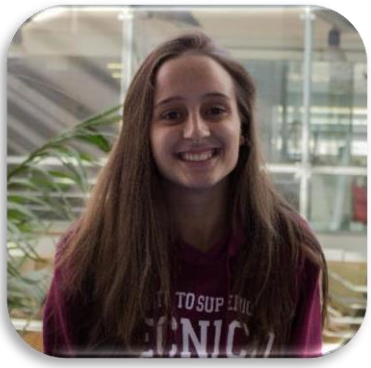
Instituto Superior Técnico

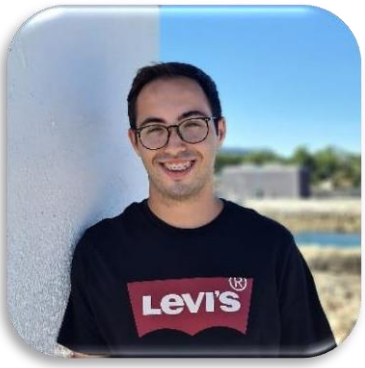
Highly Dependable Systems

Group 13 - Taguspark

Project - Stage 1
Highly Dependable Location Tracker



Sara Machado
86923



Rafael Figueiredo
90770

Ricardo Grade
90774

1. Design

We used the following Technologies:

- Programming Language: Java 15
- Database: PostgreSQL 12
- Build Automation: Maven
- Cryptographic Library: Java Crypto
- Communication Technology: gRPC
- Testing Framework: JUnit (Using Shell Script to run Server)

We decided to use PostgreSQL as the Server Database to assure the data persistence and prevent data corruption, due to crashes.

The communication technology adopted was gRPC because it is easy to use and provides the system a good performance, handling multiple requests in different Threads.

We used JUnit to build automated tests since it allows to write flexible tests calling the system methods directly.

Our system is divided in 5 Modules:

- A Server which is responsible for answering the Clients.
- A User and a HA which are the Clients of the Server.
- A Library which contains Cryptographic functions used by all Modules.
- A Byzantine which contains misbehaves of the Clients.

2. System Processes

2.1. Base Considerations

- Our system is asynchronous. Therefore, a User can request its Location at any epoch.
- Communication with the Server uses HMAC for authenticity and AES-CBC for confidentiality after a Session is established.
- Communication between Users uses Signatures.
- All Exceptions thrown across Modules are signed along with the received Nonce, with the emitters Private Key, to prevent spoofed message rejections.

2.2. Diffie-Hellman

Our system ensures Perfect Forward Secrecy. Therefore, whenever the Client wants to request the Server, they establish a Session Key, to do so they execute the Diffie-Hellman Protocol. This Key has a short-term period validity, when it expires the Client establishes a new Session Key.

2.3. Location Proofs Submission

When a User needs to Proof its Location at a given epoch to the Server it requests Location Proofs to the nearby Users. This request contains two additional fields, a Nonce and the emitter's Signature. The respondents will assert the closeness of the User and respond accordingly.

When the requester is indeed nearby, the respondent will generate a Proof containing the requester Username, Location and Epoch. This Proof is then signed by the respondent using its Private Key. Along with the signed Proof it is also sent a Signature of the Proof and the Nonce, which is validated by the requester, to prevent spoofed messages.

When the respondent either detects that the request is not properly signed by its emitter or the requester is not in its proximity, it throws an Exception.

To assure the Server can verify the validity of its Location in a scenario where it can be tolerated $f' < f$ byzantine Users in a proximity of each User, the requester gathers f validated Proofs to submit to the Server.

The Server then verifies the authenticity of the submission request, as well as its Proofs, and if there are enough valid, it keeps the Location submitted.

2.4. Queries

- The User can only request its Location at a given epoch.
- The HA can request the Location of any User at a given epoch.
- The HA can request a list of Users at a given Location at a given epoch.

The Server is able to verify whoever issued the request, because a Session has already been established between them.

3. Guarantees

- **Dropping Messages Prevention**:
  - If the request times out, the requester retries it.
- **Rejecting Messages Prevention**:
  - If the Exception is not correctly signed by its emitter, the requester retries the request.
- **Replay Attacks Resistant**:
  - A Nonce is always appended to every request. The Signature or HMAC of the reply must contain this Nonce. Doing so the requester can verify the freshness of the reply. Regarding the Server, it also verifies the freshness of the request by keeping the Nonces.
- **Authenticity**:
  - Signatures (RSA-4096) or HMACs (SHA-512) are generated and appended in every message over its payload and Nonce.

- **Perfect Forward Secrecy**:
  - The Diffie-Hellman (DH-3072) is used to establish Session Keys between the Server and its Clients with a short-term period validity.
- **Confidentiality**:
  - Messages along with IVs are encrypted by Session Keys (AES-128) using the CBC Mode between the Server and its Clients.
- **Byzantine Users Safeness**:
  - The Server only accepts a Location submission when N Location Proofs issued by different Users are deemed valid, being $N \geq f$. Where $f' < f$ byzantine Users in a proximity of each User.
- **Data Persistency and Non-Corruption**:
  - The Server keeps its data in a Database, which survives to System Crashes, keeping it consistent even if it happens in a middle of an operation.
- **Client Concurrency**:
  - The Server is implemented using properly synchronized regions for accesses at the same data.