# Highly Dependable Location Tracker

MEIC-T | Group 25

Stage 1 Report


André Augusto 90704

Lucas Vicente 90744

José Oliveira 87678


April 2021

# 1 Solution

## 1.1 Primitives

- **Location Proof**: *(proverId, ep, proverLat, proverLng)* signed by a nearby witness.
- **Location Report**: *(provers location + the collection of correct distinct proofs)* signed by the prover.

## 1.2 Proximity Service

The proximity service handles all of the interactions client-to-client, such as requesting and responding to location proof operations. A user which intends to proof their location for a certain epoch, called a prover, must collect $f$ distinct correct proofs, from nearby users or witnesses. Only then, with *f witnesses + 1 prover* users vouching for the same location, is the report considered legitimate.

## 1.3 Location Service

The location service handles all of the interactions client-to-server, being submitting and obtaining location reports, and obtaining lists of users that were in a specific location at an epoch.

In both services, all of the communication guarantees integrity, authenticity and non-repudiation with the use of digital signatures. On top of that, in the location service it also guaranteed confidentiality with the use of hybrid cryptography AES-256 + RSA-2048.

# 2 Addressing Design Requirements

## 2.1 Properties

1. *"A correct user should be able to eventually generate a report of its own location that is deemed valid by the system."*: Since a user needs $f$ correct proofs, we guarantee that if a user has $f$ correct nearby users it will eventually generate a legitimate report of its own location.
2. *"A correct user should not be able to repudiate any previously submitted location report."*: A correct user as a prover generates a report as specified in 1.1, we also store this in the server and send it whenever any query is made to the location server.
3. *"A byzantine user should not be able to create a (false) proof of location on behalf of a correct user."*: Accepted proofs follow the structure specified in 1.1, therefore without the proper signature the impersonation results in the proof being considered illegitimate.
4. *"A byzantine user should not be able to create a false proof of its own location."*: Only a proof following the structure specified in 1.1 is deemed legitimate, and a proof with the prover as witness is not considered legitimate.

## 2.2 Dependability and Security Attributes

### 2.2.1 Integrity

Both the client and server make verifications on a report's proofs. On the client side, a correct user would only submit a report with at least $f$ correct distinct proofs, with the proper signature and location. On the server side a report is also only accepted with $f$ correct distinct proofs. With this in mind, we guarantee the absence of improper system modifications.

### 2.2.2 Safety

Whenever a user submits a new correct report the server saves persistently its current state as two JSON files, a main one and a backup one. With two files, we assure that in corruption of the main one, the server is able to restore its state through the backup file, allowing recovery from possible crashes.

## 2.3 Possible Threats and Corresponding Protection Mechanisms

### 2.3.1 Drop & Reject Messages

When any request is made, with no response in 1 second, the same message is sent over, until a response is received.

### 2.3.2 Manipulate Messages

Since all requests and responses, client-client and client-server, are digitally signed, and verified when received, it is assured data integrity.

### 2.3.3 Duplicate Messages

A prover verifies duplicate proofs, when requesting them, and if any are found they will be discarded from the report. The server, when receiving a report, will also check the existence of a report for the same user at the same epoch, if found, it will be discarded and the user is properly notified.

### 2.3.4 Eavesdropping

It is guaranteed confidentiality on all client-server communications. The contents of the messages exchanged are encrypted using an AES-256 key, generated for that specific message. This key, is also encrypted using the public key of the other end, ensuring only the intended receiver is able to decipher the key, and consequently of reading the content itself.