Byzantine Fault Tolerant Banking (BFTB)

SEC Project Stage 1 - 2022

Alameda - G05

Larissa Tomaz ist192506 Rodrigo Gomes ist192548 Marta Brites ist192520

1 - Introduction

The goal of this project is to develop a highly dependable banking system, with Byzantine Fault Tolerant (BFT) guarantees, named BFT Banking (BFTB). The BFTB system maintains a set of bank accounts and the clients can perform operations such as transfers between accounts and transactions history requests.

2 - Assumptions

-There is a Public Key Infrastructure in place, although, for simplicity, the clients and the server use self-generated public/private keys and manual key distribution via shared directories
-There is a single non-malicious server that is only subject to crash failures from which it eventually recovers
-The client library is trusted
-The communication channels are not secure
-The attacker can drop, reject, manipulate and duplicate messages

3 - Dependability and security guarantees

-The balance of each account is always positive
-The state of the accounts cannot be modified by unauthorized users
-The system guarantees the non-repudiation of all operations issued on a bank account by both parties (sender and receiver)
-All the operations that modify the balance of the accounts are logged and can be verified later by an auditor

4 - Proposed solution

Our system consists of the following components:

- Server: a reliable server that only allows authorized users to access and manipulate bank accounts
- Database: used to store user's account and transaction information in a persistent way
- Client: the module responsible to translate applications calls into requests to the server

Each message exchanged between the client and the server is completely **hashed**, which guarantees the **authentication** of both parties involved in the communication. Furthermore, each message's hash is also **signed** with the sender's private key so that the property of

non-repudiation of those requests and responses is met. To conclude, each message exchanged contains a **nonce,** also taken into account in the message's hash calculation**,** that serves to avoid duplicated messages that could be used in replay attacks. All in all, the hash is used to fully verify if a message has been tampered with or not and the signature to validate the sender's identity and the credibility of that hash.

5 - Attacks Protection

-Replay attack: the messages exchanged between the server and the client include a nonce that ensures that duplicated messages are not accepted, thus preventing replay attacks.

-Man-in-the-middle: each message sent by the client or server is signed with their respective private key, so an attacker who tries to manipulate any of the messages exchanged (active man-in-the-middle) is easily identified because they cannot reproduce that signature.

-Sybil attack: the authenticity of the server is verified through the signature of the messages sent by it. The signature is validated on the client side through the server's public key, which we assume is previously known by the client.