David Gonçalves 73891
João Portugal 84731
Nuno Pinhão 84748

Report

Introduction

For this assignment, it was asked to develop a trading system, where users can trade goods with a notary certifying it and do it ensuring some guarantees regarding integrity of information and dependability of information.

Problem

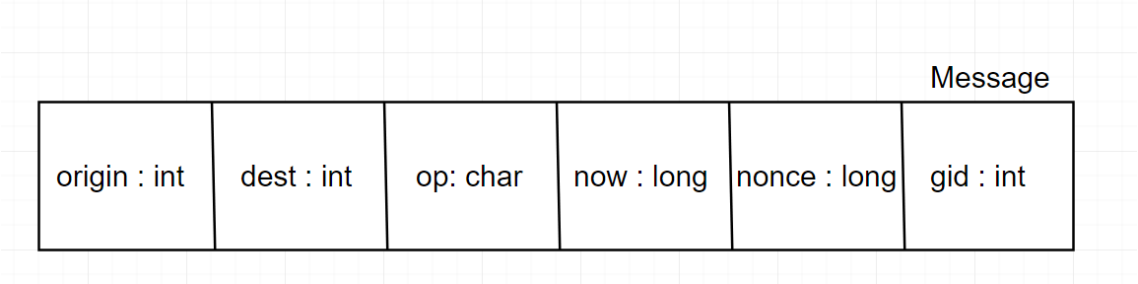
In order to ensure that information between users isn't corrupt or subject to attacks, some measures had to be taken. The possible attacks considered for this assignment and resolved within the proposed architecture are the following:

- Replay Attacks
- Man in the middle attacks
- Sybill Attacks
- Trading items owned by another user

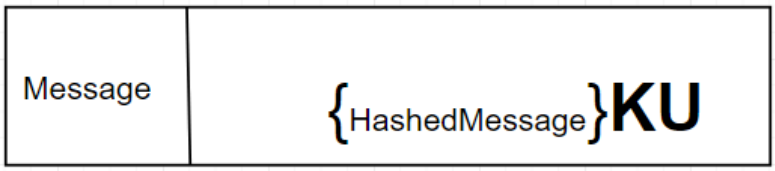
Solution

The solution proposed was composed by three modules. A security module, where all of the security classes were created and implemented, a user module that creates a connection with either another user or the notary, and a notary module, where the notary received requests from users and answered them. All of the users have each one pair of cryptographic keys, a public key known by everyone and a private key only known by him. This key's were used using Ellyptical Curves. The private key is protected using PBKDF2 algorithm using a random salt generated using SHA1PRNG, and can only be accessed using a password. For each session initiated a user ID was asked and a password to access the private key. The private key would then be loaded to a private field in the user class. The notary has a Citizen Card which he uses to certificate messages sent by him. Regarding trade of information lets' start by describing a simple message exchange between a user and the notary.

An User starts a TCP connection sending a message to the notary. The message has the following format:



David Gonçalves 73891
João Portugal 84731
Nuno Pinhão 84748



Message has 6 fields:

An origin: who sends the message

Destination: For whom is the message

Op: what operation does the sender want to perform

Now: A timestamp generated just before the creation of the message packet

Nonce: A random number generated.

GID: the good ID the sender wants to perform the operation on.

The message is then hashed using SHA2 and signed using the sender's private key with Elliptic Curve Digital Signature Algorithm.

Upon arrival the notary starts by checking if the timestamp is within a threshold. We set the threshold for 10 seconds. If it isn't, operations proceed normally. If it is, it will then check if the nonce corresponds to any nonce saved in a HashMap having the timestamp as a key. If it is, it can mean a replay attack and the notary rejects the packet. If it isn't, the notary will then verify if the message isn't corrupted. It will start by hashing the original message and verify the signature of the hashed message sent, using origin's public key. If the message is corrupted, notary will reject the packet. If not, it will check what operation does the sender want to do and perform it if he is able to do it (for example if the origin's ID wants to sell a good, if he actually owns it). The notary will then answer with his own message following the same format as above. However, the signature process is a bit different. First, the notary will have to put in a pin to be able to access his citizen card, and then put in his pin again to sign the message. The notary's cryptographic keys are generated using RSA and for the signature it is used SHA1 with RSA. The message is then sent to the user where a very similar process to the one described is taken in place. For all the communication between users and between users and the notary and for all operation purposes, the protocol above is followed. The good's state and ownership is maintained using a database, therefore if the server crashes, it will recover to its previous state.

Results

The results were pretty successful with manual testing resulting. Extensive and automatic testing might reveal some unwanted problems. The pair timestamp nonce is useful to have instead of just one of them, since it allows a grace period where a request can get delayed and still be accepted and the nonce ensures no replay attacks can happen (albeit it they can still

happen however have a minimal probability close to zero). The elliptical curves keys implemented are used for efficiency since they are allowed to have 224 bits instead of the 2048 required with RSA. For the citizen card it was mandatory to use RSA. Sybill Attacks are prevented just by knowing or not knowing a user's public. If it doesn't exist it is a fake user. Overall, the dependability requirements are maintained and integrity ensured.