HDS Notary

Cíntia Almeida (MEIC - 85139) Kevin Corrales (MEIC - 94131)
Mário Silva (MMA - 93799)

April 2019

Abstract

We develop a Highly Dependable Notary application (HDS Notary) that certifies the transfer of ownership of arbitrary goods between users. A number of dependability guarantees are implemented, with the assumption that the Notary is a trustworthy entity.

1 Introduction

1.1 Problem

The goal of the project is to create an application where users can realise ownership transfers with recourse to a central authority for the validity of transfers, that eliminates the need for trust among users. Firstly, a transfer is considered *valid* if:

1. the good to be transferred is currently owned by the seller;
2. both the seller and the buyer have expressed interest in executing the transaction.

Since we don't assume the trustworthiness of users, various possible behaviours are possible, which will have to be guarded against. For instance, a user may attempt to sell the same good twice, or even fake her identity. Another type of failure may be a crash of the system, in which the Notary service must guarantee no loss of corruption of its internal state, and have an up-to-date mapping of Goods to Users.

1.2 Overview of the system

A typical transaction should run as follows:

1. Seller expresses her intention to sell Good to the Notary, which will change the state of Good from *NOTONSALE* to *ONSALE* if Good belongs to Seller;
2. Buyer requests to know the state of the Good from the Notary;
3. Buyer expresses his intention to buy Good from Seller
4. Seller sends a TransferRequest to the Notary
5. The Notary checks the validity of the transaction and returns a Certificate

To ensure dependability, all buy and sell request, as well as the certification by the Notary must be non-repudiable.

2 The system in more detail

2.1 Good counter

Associated to each Good in the system there is a counter which is a number that changes each time the state of the Good changes as well. This implementation aims to ensure protection against *replay attacks* since a message is only valid if it has the correct counter (an intercepted message could not simply have that number replaced since that number is also signed).

2.2 Signatures and Certificates

To ensure non-repudiation, every message sent either by a user or the Notary must contain a signature that only the sender could produce with his private key but that can be verified by any other person in the system using the public key. This is achieved with an RSA encryption of the message.

This method ensures non-repudiation, since, given a new set of data, only the owner of the private key can create its respective signature. The Notary certifies every valid transaction with a Certificate generated using the Portuguese Citizen Card.[1]

3 Possible attacks, errors and countermeasures

3.1 Man-in-the-middle attack

This type of attack is characterised by the interception of data between two parties, possibly even unbeknownst to them. These data can then be simply blocked, in which case there is no communication, or they can be sent altered or unaltered to the intended receiver.

The alteration of the message will render the signature invalid, so the message can't be simply altered without knowing the sender's private key.

3.2 Replay attack

In this attack there is the repetition of a message outside of its intended context. In our case, for example, if Alice sends a buyGood request to Bob, can she be sure that Bob won't use her request at a later time if he is again in possession of that certain Good?

This attack can be countered by a guarantee of freshness of the messages. The *good counter* ensures that messages are only valid in the time window in which the good is in a certain (i.e. once it changes states, either becoming on sale or changing owner, the counter also changes). However, it is also necessary to include a guarantee of freshness of the message from the Notary, and in that case the User making the request necessarily issues a challenge (random String of 20 characters) in the message that is subsequently checked.

Another possible attack would be for a user to login with an incorrect identity. In our system, if a user attempts to connect with an ID that is already in use they get rejected.

References

[1] *Manual técnico do Middleware Cartão de Cidadão*, version 1.61.0, January 2016