Highly Dependable Systems

Sistemas de Elevada Confiabilidade
2018-2019

HDS Notary

Stage 1

The goal of the project is to create a Highly Dependable Notary application (HDS Notary). The main goal of the HDS Notary is to certify the transfer of ownership of arbitrary goods between users. HDS Notary has a set of dependability guarantees, further discussed below, which will be strengthened on Stage 2.

1. Goals and overview

The HDS Notary application maintains information regarding the ownership of goods, e.g., which house belongs to which user. This information needs to be updated in a dependable way to reflect the execution of transactions that transfer goods between exactly two users, which we denote in the following sections as **buyer** and **seller**.

An initial set of **users**, the **set of goods they own** (represented via a set of tuples <goodID, userID>) and the **notary identity** are assumed to exist when the application first starts. The initial set of users and goods is immutable and known by every user. In other words, only the initially existing users are allowed to use the application, and only the initially existing goods are allowed to be transferred. Also, the notary initially has access to the set of users (and their public keys) and the corresponding set of goods they own.

After this initialization step, the users can start executing **transactions,** i.e., sell/buy goods. The notary should be contacted once the seller of a good has agreed to transfer one of his or her goods to a buyer. The role of the notary is to certify that the transfer is valid, namely that:

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

Users can perform the following operations:

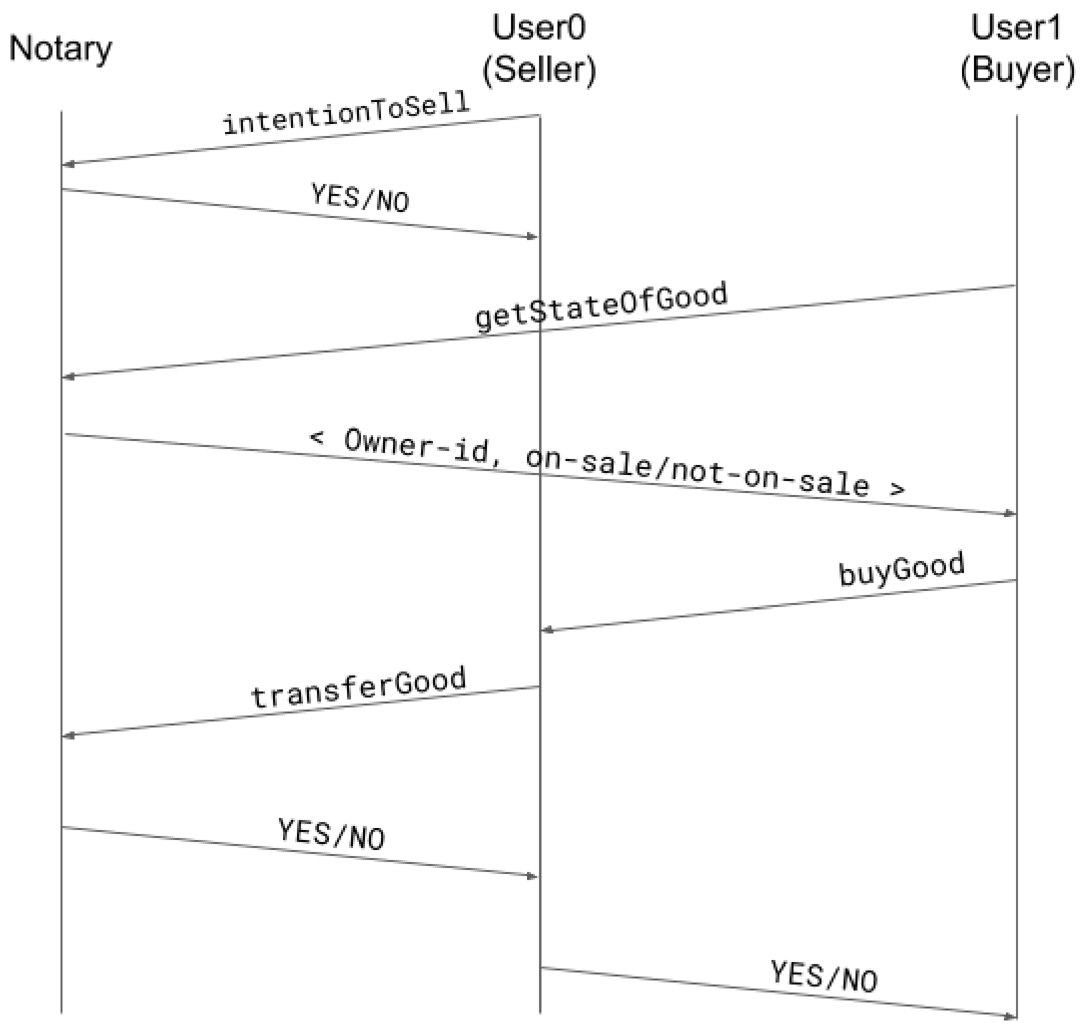
1) obtain the status of any good, i.e. its owner and whether it is on sale;
2) express their intention to sell any of their goods or buy any good currently available on sale;
3) submit their transaction requests to the notary.

For simplicity, assume the money transfer associated with the transaction is done outside the system, i.e. the Notary needs to validate the transaction only according to the rules above and disregard any monetary balance or transfer from the buyer to the seller.

Finally, the Notary is equipped with a Portuguese Citizen Card (*Cartão de Cidadão)*, which shall be used to certify its identity as well as provide strong cryptographic guarantees on the transactions it validates.

2. Workflow of transactions

Below the sequence diagram illustrating the operations of the system is illustrated:



1) Seller expresses her intention to sell one of her own goods, by invoking the method intentionToSell() on the Notary. The notary replies acknowledging or rejecting the operation depending on whether the seller is or is not the current owner of the good.
2) The buyer obtains the state (current owner id and on sale vs not on sale) of a given good by invoking the getStateOfGood() method.

3) The buyer expresses his intention to buy a given good from a seller by invoking the buyGood() method on the seller.
4) Upon receiving the first expression of interest to buy one of her goods, she requests the validation of the transaction by the invoking the transferGood() method on the notary.
5) The seller informs the buyer of the outcome of the transaction.

Upon receiving a TransactionRequest the Notary must validate it, as specified in Section 1. In case the transaction is deemed as valid, the notary must:

1) alter the internal state of the mapping Goods → Users;
2) send back a certification attesting the validity of the transaction;

From this point onwards the buyer is considered the legitimate owner of the good.

3. Dependability requirements

In Stage 1 the Notary is assumed to be centralized (a single server) and trustworthy, i.e., it will follow exactly the protocol specified above. These restrictions will be lifted in the second Stage. Moreover, the Notary must hold a Portuguese Citizen Card, which is used to certificate the transfer of ownership. Note that even though the server is honest, it can crash and later recover. The implementation of the Notary service should guarantee no loss (or corruption) of its internal state in the presence of crash faults.

Users are identified through a Public-Key Infrastructure (PKI). For simplicity, this should be done using self-generated asymmetric keys. The set of users and their Public Keys can be assumed to be static and known by all the users and the Notary.

Users cannot be trusted and might try to attack the system for their own benefit, such as selling the same good twice, or to disrupt the regular system operation.

All the buy/sell requests, as well as the certification emitted by the Notary, should be non-repudiable.

For transparency purposes all the communication should take place in clear, i.e., no confidentiality. An attacker can interfere with the messages being exchanged in the network, including message drops, manipulations and duplications.

Hence, students have to analyze potential threats (examples may include man-in-the-middle, replay or Sybil attacks), and design application level protection mechanisms to cope with them. There are several approaches to address these issues, so it is up to you to propose a design and justify why it is adequate.

3.1. Design and implementation requirements

The design of the HDS Notary system consists of two main parts: a library that is linked with the application and provides the API specified above, and the server that is responsible for keeping the information associated between users and goods. The library is a client of the server, and its main responsibility is to translate application calls into requests to the server. Anomalous inputs shall be detected by the server, which should generate appropriate error messages and return them to the client-side.

The project must be implemented in Java using the Java Crypto API for the cryptographic functions.

We do not prescribe any type of communication technology to interface between the client and the server. In particular, students are free to choose any message passing technology available in Java such as sockets, remote object interface, remote procedure calls, or SOAP-based web service.

However, **HDS Notary operates under the assumption that the communication channels are not secured**, in particular solutions relying on secure channel technologies such as TLS are not allowed.

For simplicity, students can assume that the set of existing users, their addresses and any public/private key associated with them, is static and known a priori.

4. Implementation Steps

To help in your design and implementation task, we suggest that students break this task into a series of steps, and thoroughly test each step before moving to the next one. Having an automated build and testing process (e.g., JUnit) will help you progress faster. Here is a suggested sequence of steps:

- **Step 1**: As a preliminary step, before starting any implementation effort, make sure to have carefully analyzed and reasoned about security and dependability issues that may arise and the required counter-measures. Only then, move to step 2.
- **Step 2**: Simple server implementation without dependability and security guarantees. Design, implement, and test the server with a trivial test client with the interface above that ignores the crypto parameters (signatures, public keys, etc.).
- **Step 3**: Implement the Citizen Card usage by the Notary.
- **Step 4**: Develop the client library and complete the server – Implement the client library and finalize the server supporting the specified crypto operations.
- **Step 5**: Dependability and security – Extend the system to support the dependability and security guarantees specified above.

5. Submission

Submission will be done through Fénix. The submission shall include:

- a self-contained zip archive containing the source code of the project and any additional libraries required for its compilation and execution. The archive shall also include a set of demo applications/tests that demonstrate the mechanisms integrated in the project to tackle security and dependability threats (e.g., detection of attempts to tamper with the data). A README file explaining how to run the demos/tests is mandatory.
- a concise report of up to 4,000 characters addressing:
    - explanation and justification of the design, including an explicit analysis of the possible threats and corresponding protection mechanisms,
    - explanation of the dependability and security guarantees provided.

The deadline is **April 12 at 17:00**. More instructions on the submission will be posted in the course page.