Instituto Superior Técnico, Lisbon, Portugal
HDS Coin, stage 1

Gonçalo Batista, 80946
Sheng Wang, 86324
Gonçalo Garcia 90869
Group 29 - Alameda

April 5, 2018

1 Introduction

The purpose of this project is to create a service for users to maintain their ledgers with security in mind. The users can send transaction to others users and receive if there is any transaction from others users.

2 Architecture

Let us consider the Figure 1 which represents a normal transaction between Alice e Bob. First of the all Alice registers to the server and then receive the confirmation, secondly Bob registers to the server and then receive the confirmation. After Bob registered in the server, Alice may send transaction to him, getting as response a send confirmation. Consequently, Bob checks if there is any transaction for him, in case there is no transaction he receives null, otherwise he will get a list of transactions for finally to receive those transactions. Table 1 clarifies message's encryption, excepted register and register confirmation, used to be exchanged between the client and the server:
M-Message, which is exchanged content;
N-Nonce, which is a random generated number;

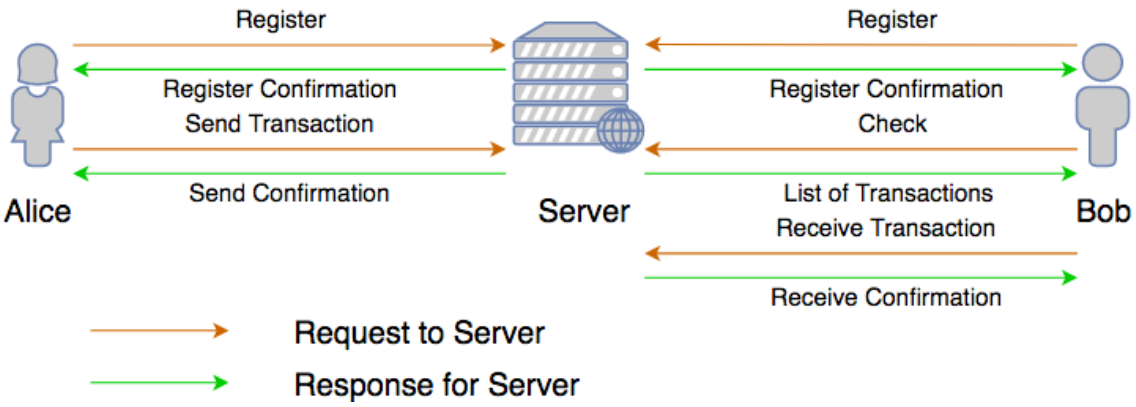

Figure 1: HDS Coin scheme

Request to Server	{M || N || {H[M || N]}C1 }K || {K}S2
Response for Server	{M || N || {H[M || N]}S1 }K || {K}S2

Table 1: Message Table

H-MD5;
K-AES;
C1-Client's Private Key;
C2-Client's Public Key;
S1-Server's Private Key;
S2-Server's Public Key;

3 Implementation

In the first stage, it is implemented the server, which may has crash failures, i.e. it can crash at any time, and the client who connect to it. There can be more than one clients connected to the single server. In our project for testing purpose, it is already created three users, namely Alice, Bob and Charlie with their own key pair, i.e. private key and public key, and others' certificate, including server's one, for obtaining their public key.
The client registers with his public key, and the server creates a ledger with his public key, balance, a list of pending transactions and a list of transactions and saves it on a json file. We consider this request is not a security threat, so as mentioned above the register's request has no encryption. The reasons are because it cannot modifies the balance of anyone and it only disclosures the public key of user which is visible for everyone.
Every time a client sends a transaction to other his balance is subtracted, in case that after subtraction is negative, the operation is canceled, otherwise this transaction is added to the list of pending transaction of other's ledger. Hence other client may check his depending transaction. The client may receive any pending transactions obtained by check, every time a client receive a pending transaction, this is removed from list of pending transactions and added to the list of transactions of both clients.
If the client makes a request of send transaction to the server and the connection fails. the message is stored in a json file and resend in next time when the client executes the request of send transaction.

4 Attacks and Assurances

Spoofing
Every message has hashed information encrypted with the sender's private key, this gives sender's signature.

Tampering
Every message has hashed information encrypted with sender's private key, which may alert receiver if the data is changed.

Non-Repudiation
Every message has hashed information encrypted with sender's private key, and this private key authenticates the sender.

Information Disclosure
Every message is encrypted with aes, and aes is encrypted with server's public key before sending to the server. Only the server who generated the certificate of that public key may have the private key to decrypt the aes and then using aes to decrypt the message.

Replay attack

The message has a nonce, which is a random generated number with class SecureRandom, every time the server or even the client receive the message save the nonce, and the message with that nonce cannot be used again.

5 Conclusion

According with same experiences done, it works perfectly. It is implemented with security in mind, and it can prevents against the list of attacks mentioned.