Instituto Superior Técnico, Lisbon, Portugal

HDS Coin, stage 2

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

May 15, 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 transactions to others users and receive them if there is any transaction from different users. In stage 1, there were some problems left, namely non-repudiation, which is fixed by storing the sender's signature in every transaction, file corruption, which is fixed by adding a backup file, negative or null value, which is fixed, and having to store nonces forever which is also fixed by using a timestamp. In the stage 2, we added a (1,N) Byzantine Atomic Register Algorithm. This is, there will be server replication, and clients are able to request operations like read (check and audit) and write (send and receive) to those servers. In addiction, some of servers may be byzantine as well as the clients.

2 Architecture

Figure 1 shows the architecture of our proposal. Client broadcasts a request to all servers, then every server broadcasts the same request to all other servers, making sure that every other servers received the same, then sends a responce to client. Manager is just a program where we can control all servers, e.g. start, crash, delay or even change client's requests.

3 Implementation

We implemented the algorithm taught in the classes. It is based on "write-back" in Section 3.1 and on "authenticated double-echo broadcast" in Section 3.2. There are several properties that are assured to perform algorithms mentioned above.

Termination - If a correct process invokes an operation, then the operation eventually completes.

Validity - A read that is not concurrent with a write returns the last value written; a read that is concurrent with a write returns the last value written or the value concurrently written.

Ordering - If a read returns a value v and a subsequent read returns a value w, then the write of w does not precede the write of v.

No duplication - For every process p and label , every correct process delivers at most one message with label and sender p.

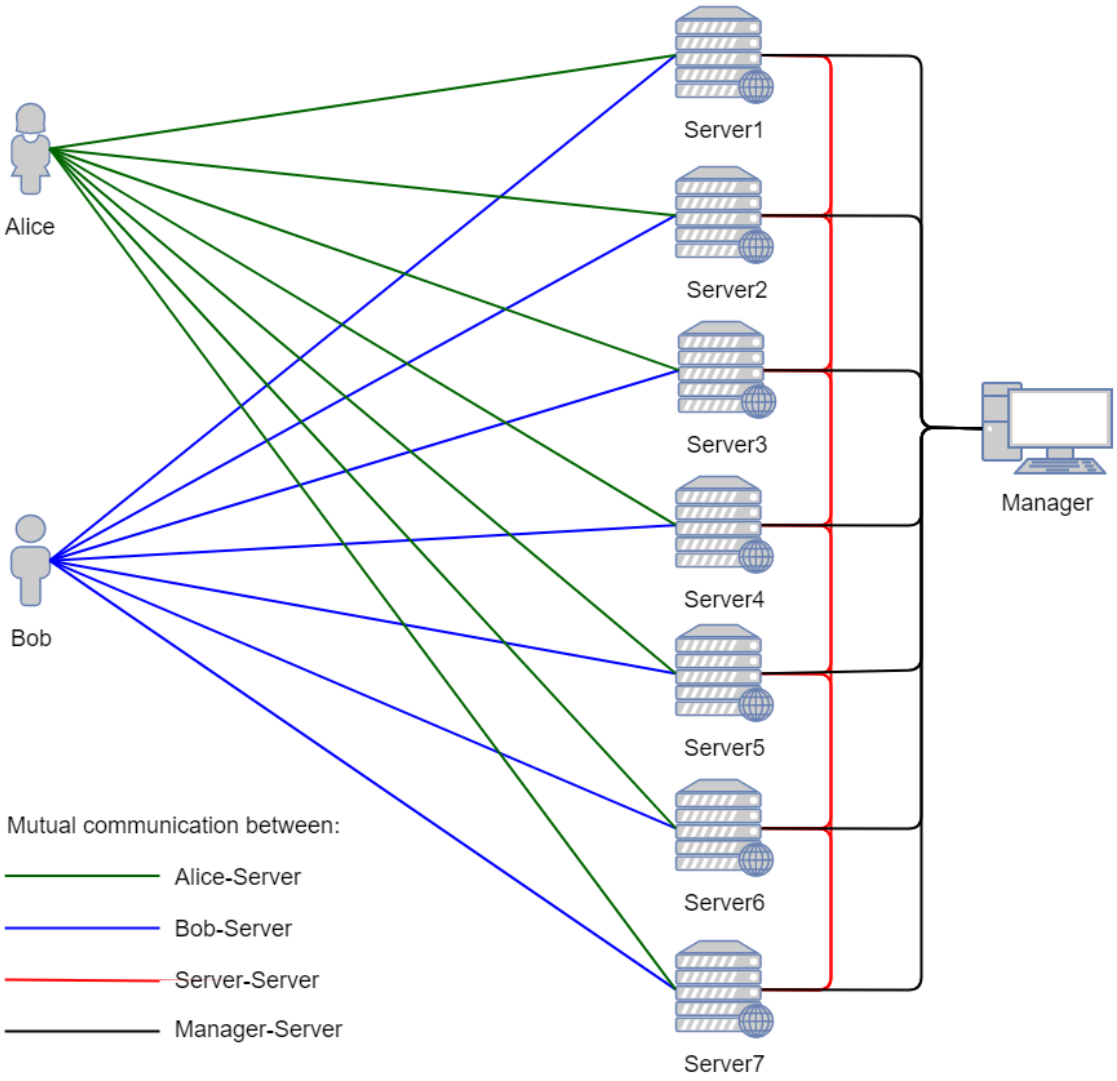

Figure 1: HDS Coin scheme

Integrity - If some correct process delivers a message m with sender p and process p is correct, then m was previously broadcast by p.

Agreement - If a message m is log-delivered by some correct process, then m is eventually log-delivered by every correct process.

3.1 Client side

In order to implement a 1-N Atomic register, we rely on an authenticated write-back-algorithm. When performing a read operation (check, audit) the client will send a message to each server, which will then send its reply. Once more than (n*f)/2, where n is number of servers and f is maximum number of servers that may fail, of messages is received, the client will pick the highest timestamp, and send that message to all of the other servers before delivering it to the user. This ensures that, before the users sees the value, the previously outdated servers will become on par with most advanced one.

Every message traded in this process has a digital signature which is verified at each step. A failure to verify the signature results in a failure of the client's request.

Obviously there are some nuances in the algorithm to take into account byzantine clients and servers. These will be explained in the Attacks section.

3.2 Server side

The algorithm "authenticated double-echo broadcast" is used with the intention of providing byzantine reliable broadcast. It is considered double-echo because it has an echo and a ready stage before the delivery of the message.

Every time a client sends an operation to server, the server stores the message and broadcasts the integrity check obtained from digital signature, as read a readID, of that operation's message to other servers as echo, i.e. server says that it received the message. Once server receives more than (n*f)/2, it broadcasts a ready message with that operation's message and again its integrity check, i.e. the server is ready to receive the message. As may often happen, a server may be delayed, hence it may not be ready before receiving ready from other servers. In this case, the server can conclude that the message is ready if it receives more than f ready and broadcasts ready to others. After a server received more than 2f ready, the message is considered valid and then delivered.

The algorithm guarantees that there is no correct server that will deliver a message which is not delivered by others correct servers.

4 Attacks and Assurances

Spoofing
An attacker may try to impersonate another client. In our system we use digital signatures on every message to ensure that this is impossible. A message will always be tied to the owner of the private key used to sign it. A failed signature check will result in an Exception and the message will be discarded.

Tampering
Since messages are sent along with a digital signature, which calculated a digest of the whole message, an attempt to alter the message's content will result in a non verifiable digital signature. If this happens the message is discarded.

Non-Repudiation
As stated in the spoofing example, the usage of digital signatures make sure that a message signed with a given private key will always be tied to the owner of that key. This way no one can state that a message wasn't sent by them unless the key was somehow stolen.

Information Disclosure

The message ciphering process uses AES to cipher the content. The key for the AES cipher is itself ciphered with the receiver's public key and sent along with the content. This ensures that only the person who owns the private key associated to that public key is able to decipher and read the content.

Replay attack

Each message is sent with a logical timestamp. This timestamp is incremented every time a write operation is performed, since only the writes have the ability to cause harm to the ledger. Each time a message is received, the timestamp is compared against the the server or client's timstamp. If the received timestamp is not higher than preview's one, the message is discarded. This prevents old messages from being resent causing double spending.

Byzantine Server

A byzantine server does not need to comply to the rules of the system, and as such, the system must take action to maintain it's correctness despite this. If the server does not respond, it will not affect the server as we only require a byzantine quorum of responses to make a decision. The server can also send a wrong message, this is especially dangerous in a read operation as the write-back could propagate this information to the correct servers. To prevent this, each time a server receives a write back (Section 3.1) it will verify the signatures of the transactions received and recalculate the balance.

Byzantine Client

A byzantine client may also try to perform double spending by sending different messages to each server. To prevent this, we use Double Echo Byzantine Reliable Broadcast (Section 3.2), which ensures that a message is only delivered by a correct server if it's delivered by all correct servers.

5 Conclusion

The stage 2 improved security concerns left in stage 1. In addiction, it added authenticated double-echo broadcast algorithm and authenticated write-back-algorithm to provide (1,N) Byzantine Atomic Register.