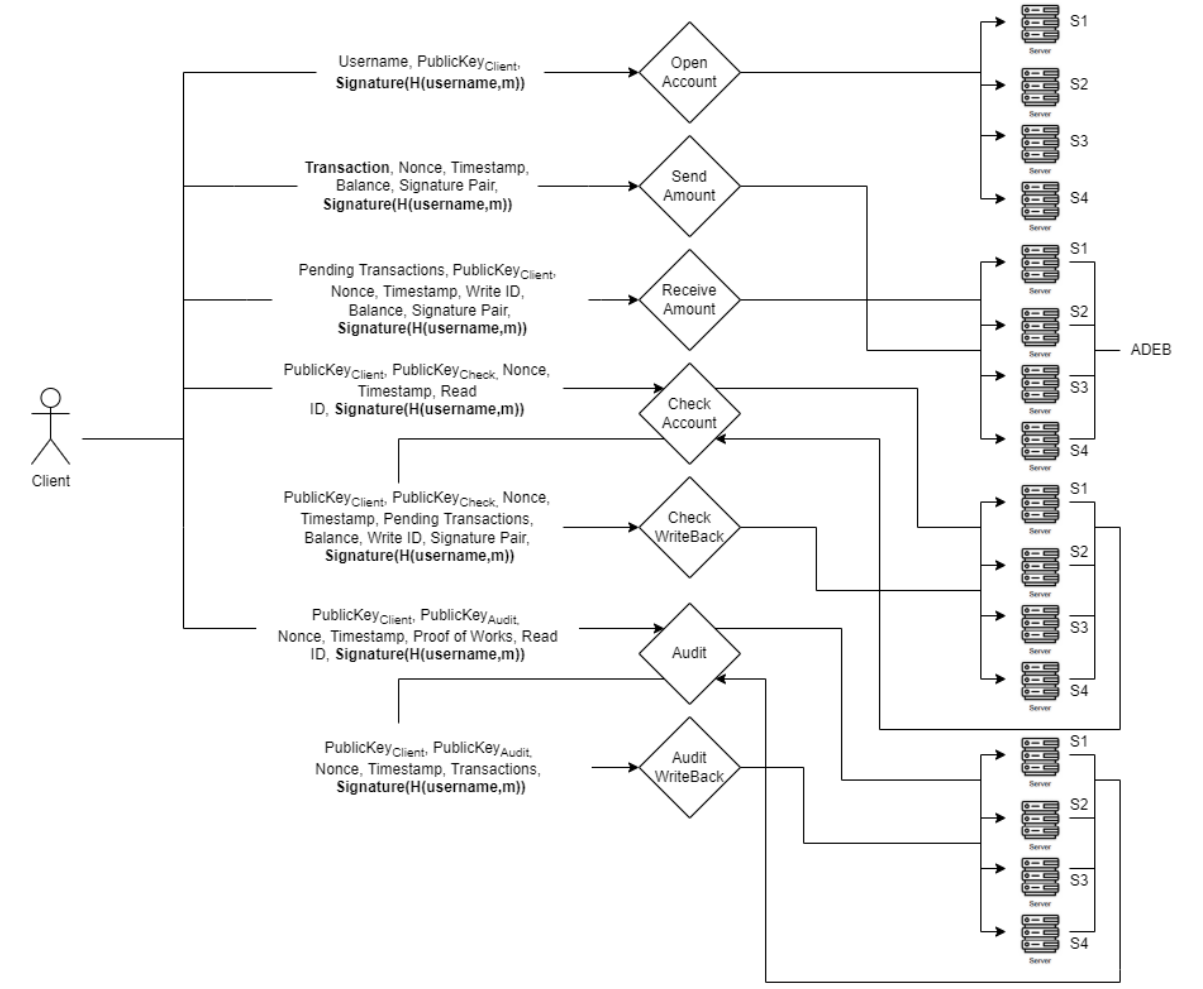
Highly Dependable Systems - Stage 2

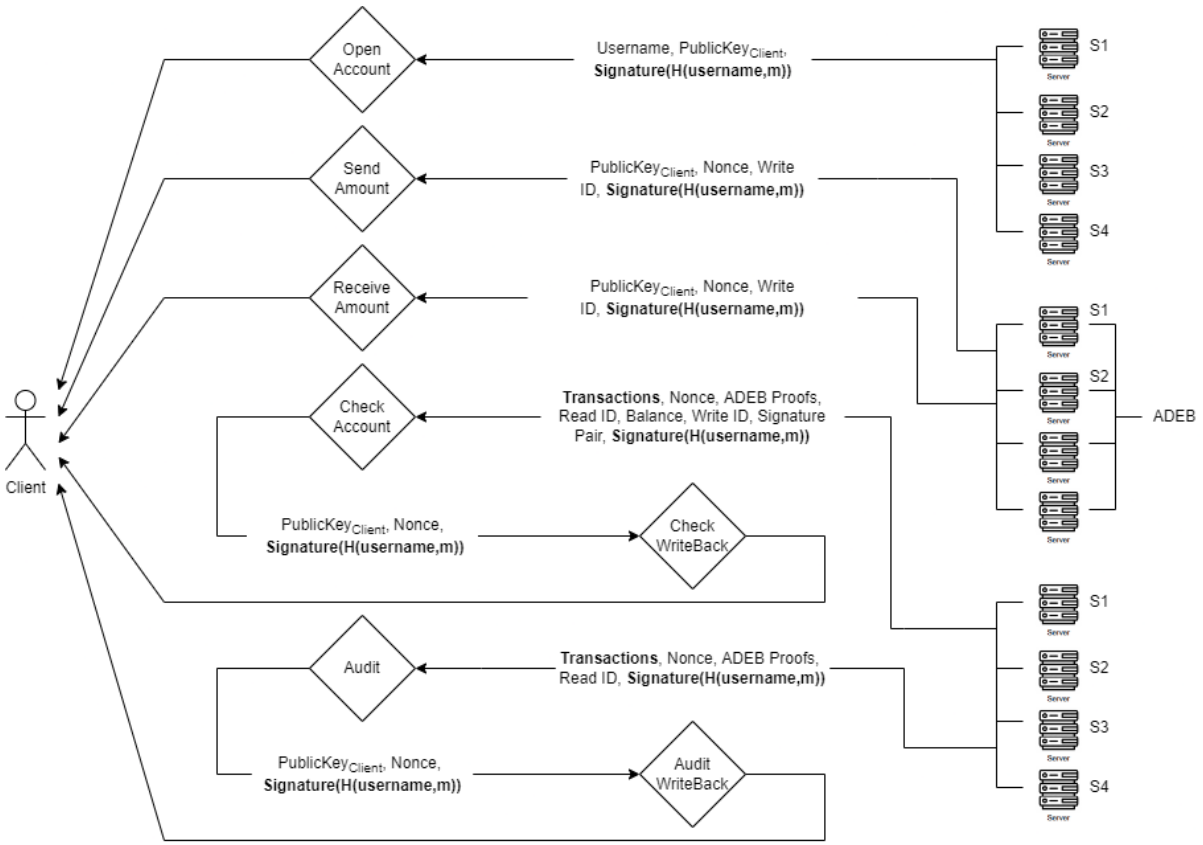
Universidade de Lisboa - Instituto Superior Técnico
Mestrado Bolonha em Engenharia Informática e de Computadores 2021/2022

Bruno Silva 102172
Guilherme Saraiva 93717
Maria Gomes 102203

The system

Below are two schemes of the whole system - Client Requests and Servers responses.





Base Considerations

- In a scenario where there are fs byzantine Servers, the number of Servers that are created is $N = 3fs + 1$.
- Our system is asynchronous. Therefore, users can perform reads and writes at the same time, but each user can perform a write at a time.
- All exceptions thrown are signed along with the received nonce, with the server's Private Key, to prevent spoofed message rejections.
- Before every write, a read is made so the user doesn't need to keep his information to perform the write operations.
- After the login, the user requests his Read ID to be used in the read operations.

(1, N) Byzantine Regular Register

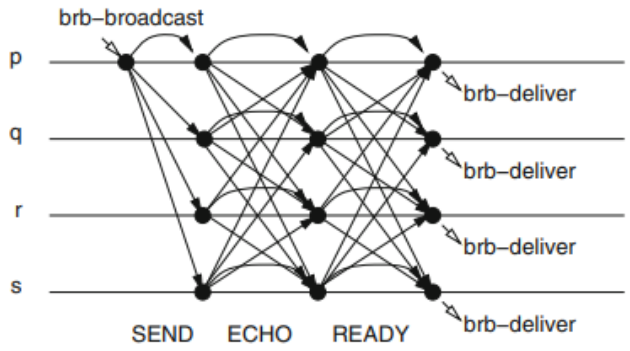
A client makes a request to all Servers and waits for (N+fs)/2+1 (BQ) number of responses. Guaranteeing that if a write is already finished, that result will always be returned by at least one Server to the client.

(1, N) Byzantine Atomic Register

When a client performs a read, a writeback is made in order to help the servers that are writing the same record.

ADEB

When a client performs a write (send/receive), servers will perform an ADEB in order to confirm that every server receives the same record to avoid inconsistencies between them.



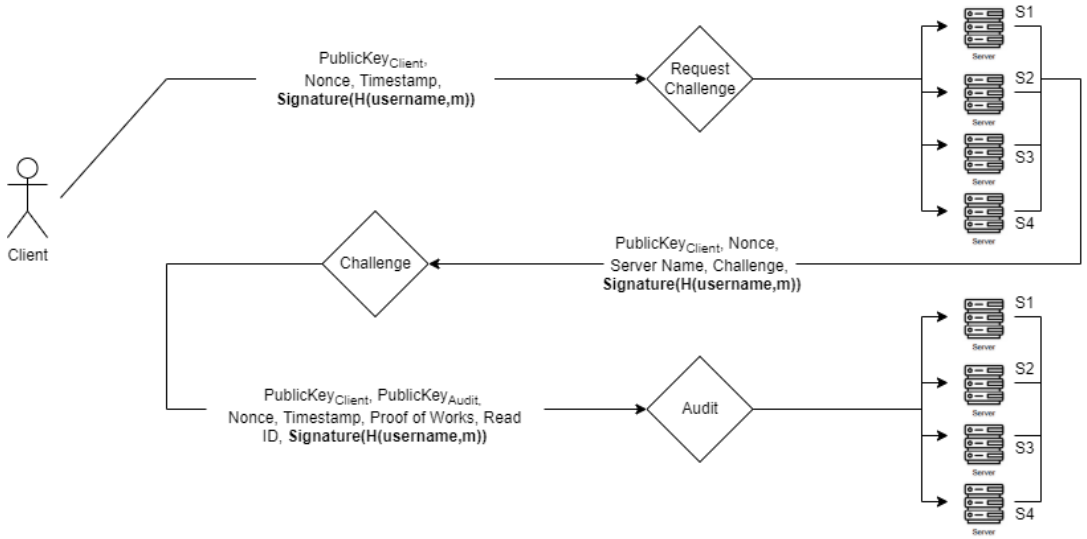
Spam Combat Mechanism

In order to mitigate an attack where a client could flood the system with computational expensive requests, a Proof of Work mechanism has been implemented. We implemented this mechanism on **audit** operation because if the user has a big number of transactions, the cost of verifying each transaction would be very high.

This protocol starts with the client requesting a challenge to the servers. Each server generates a challenge that consists of a hash of a signed random string. Besides that, each challenge is associated with a timestamp in order to prevent the client from precomputing them.

After receiving each challenge, the client computes and generates Proof of Works and sends them to the servers within the audit request.

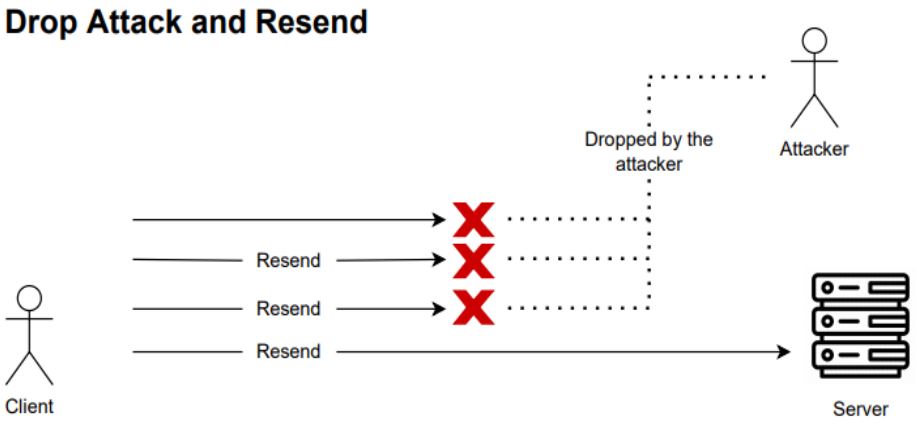
Finally, each server chooses the corresponding PoW and easily verifies if it matches the generated challenge and if the timestamp is due.



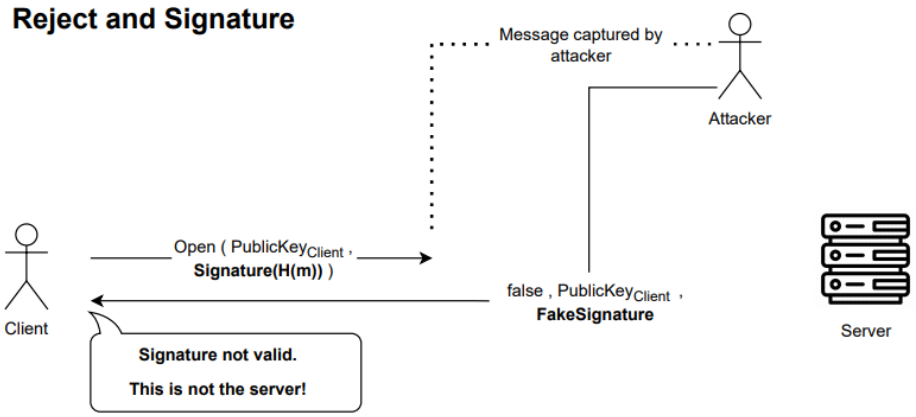
Attacks and Security

The following are possible attacks and how we protect against them:

- **Dropping**: Resend the message until the receiver gets it. The attacker can continue dropping the messages, but our system will continue resending them until it has confirmation that they arrived.

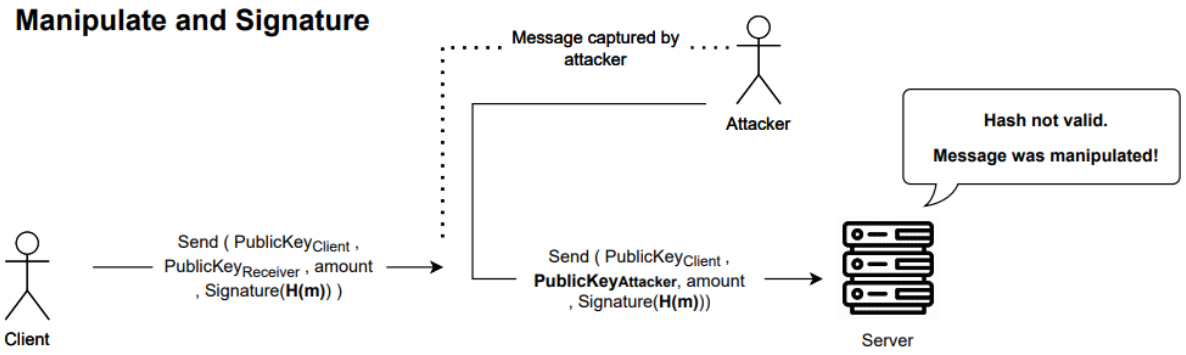


- **Rejecting:** Every message sent includes a signature of the respective sender. This signature is signed with the sender's private key, which guarantees that the user created that message. On the receiver's side, the signature is verified using the respective public key. If the public key does not open the signature, then it can be assumed that the message was not created by the "real" sender.



- **Manipulating**: Our system guarantees integrity by signing every message (digital signature using the sender's private key) with a hash of its content. Because the attacker cannot replicate the signature, he might alter the content that comes on the message, but the content inside the signature will remain unaltered. Upon receiving the message, the receiver verifies the signature.

  If the hashes do not match, it assumes that the message was altered and rejects it.

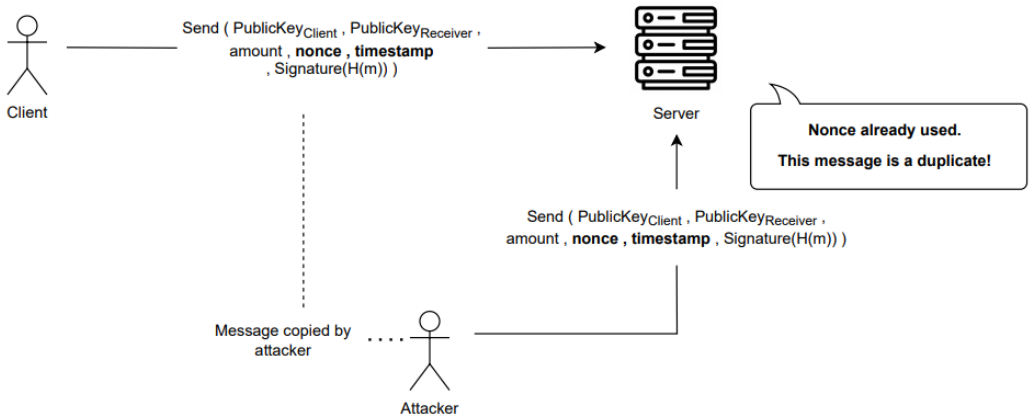


- **Replay**: Requests are sent with a nonce and a timestamp.

  The nonce can only be used once so any messages with the same nonce and timestamp will be rejected. On the server, if the timestamp is due, it rejects the message. Clock desynchronization can occur between client and server so we added

a period of 10 minutes in which the sender must send the request. The client verifies if the nonce sent by the server matches its nonce+1 and the timestamp is correct.

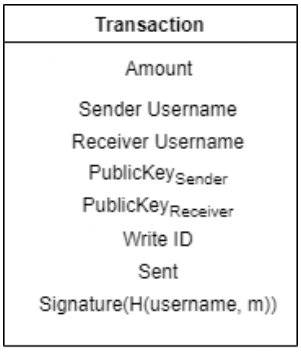
Duplication and Nonce

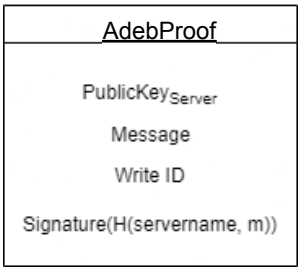


Byzantine Behaviors

The following are possible byzantine behaviors and how we protect against them:

- **Outdated writes:** Each write request/reply contains a sequential number (WID) that both server and client verifies its correctness by choosing the value with highest WID.
- **Outdated reads:** Each read request/reply contains a sequential number (RID) that both server and client verifies its correctness by choosing the value with highest RID.
- **Byzantine Client sends different write requests to each server:** Each server runs ADEB to confirm that a quorum of servers got the same request as theirs.
- **Byzantine Server sends tampered balance to the client:** Each client holds a signature pair of its WID and balance in order to, when performing a check, it is able to verify its own signature and check if the balance or the WID was tampered.
- **Byzantine Server adds new transactions:** Since every transaction is signed by the sender's private key, when performing a read, the client will be able to verify if the signature is correct.
- **Byzantine Server hides/duplicates transactions:** Since the list of the performed transactions is sequential by each transaction WID, we can easily check if there's any transaction removed or repeated.
- **Faulty Servers**: Since we only wait for a quorum of responses, we can tolerate a max number of faulty servers
- **Byzantine Client-Server interaction:** After each ADEB, a list of *AdebProofs* is stored on each client. This proves consists of a signed *ready* message by each server in order to, when a client performs a read, it can check if every signature matches and the WIDs that are inside the proof are also correct.





Integrity

Our system provides integrity by signing every message with digital signatures, hashed with the message's content.

To ensure the message was unaltered, both client and server hash the message's content and compare it to the signature's content. If the hash is equal, it means no modifications were made since the attacker cannot replicate their signature.

Dependability

To guarantee that the requests are being made by a bank client, the client must log in using an username and password.

The system offers non-repudiation by signing most functionalities with the sender's digital signature. Because only the sender is able to have its own private key, it is guaranteed it was he who signed those messages. The system does not offer confidentiality since any client can check and audit other client's accounts.

The system is stateful so if the server is rebooted, it will keep all the information about its clients. This was implemented by doing atomic writes on the file that stores the data.