Instituto Superior Técnico
SEC16/17
MEIC-A

Group 4:
João Santos 67011
Francisco Martins 76061
Paulo Anjos 87822

Stage 1:
Our objective is to build a secure Password Storage Manager, in which the Client has all his passwords safely saved inside the server's "walls".
To implement the communication between the Server and the Client requests we used REST, as we found it easier to work with and more efficient.
As the client initiates (init), his communication, he builds a pair of asymmetric keys with RSA algorithm, a KeyStore where his respective keys will be stored and a session key is exchanged between the two parties (server+client).
We assume that the Session Key is constant for all users in all sessions, so it is not stored in the KeyStore.
We considered that the client should have guarantees that their data would be protected against integrity attacks, non repudiation, replay attacks, that it should be confidential and have an authenticity mechanism. To do that, for each request, the user hashes the respective user's username and domain with a no salted hash, being signed with the user's Private Key afterwards and then ciphered with the Session Key exchanged before. With those methods we aim to guarantee that the data is confidential, robust against integrity attacks as the data is signed, and it is authenticated. Although things had to be treated differently with the user's password, since it's a more fragile data, and everyone had access to the session key, the username and the domain were easily decrypted. So before signing the password data, we cipher it with the user's Public Key. This makes it so, that only the user can decipher the hashed data, since only he has his own Private Key.
Our system doesn't protect against replay attacks. A possible solution would be using timestamps in each request, being then impossible for an attacker to repeat old messages.
Another fault in this architecture is that the session key is static and equal to anyone using the system, so the user's domain and username won't be totally confidential. Our initial thought to solve this was to use a Diffie-Hellman approach for the sharing of the session keys between the parties. The code didn’t work (the lines are commented) so we maintained a “hardcoded” session key.