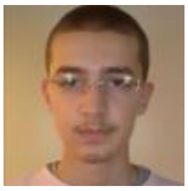
Remote Document Access

SIRS Alameda Group #20

67011	João Santos	
78131	Ricardo Mota	
79710	David Cardoso	

Introduction

In a collaborative office application where groups of users can create and edit documents remotely there is a higher need for a secure environment and communication channel. This is because we want the documents in said application to be truthful and valid, if anyone can just attack the application the attacker can just access the documents and alter, or even transfer, them and then they most likely will lose its integrity and confidentiality. Thus, why we need a secure application to fight against these security issues.

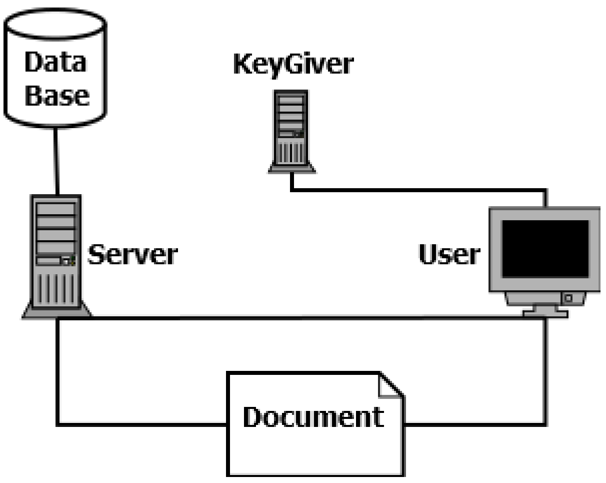
The main problem being solved here is to prohibit a foreign, or even intern, attacker to the application of having access to any of its content, or just to content they are not allowed to access in case of an internal attack. In case of it being an internal attack, it’s to make sure a registered user cannot see any content he is not permitted to see, making it stay confidential to the owner and its valid contributors.

Security Requirements

The main security requirements are as follows:

- **Integrity:** All messages and Documents exchanged between the User and the Server are not modified. Local, or Remotely, saved Documents can only be modified by the Owner or its Contributors.
- **Authenticity:** A User must login to the Server to validate his identity.
- **Confidentiality:** All messages and Documents exchanged between the User and the Server cannot be read by someone else; also about the Documents, these can only be read, locally or remotely, by its Owner and Contributors.
- **Non-repudiation:** Each User has a unique identifier (username) so that all messages and Documents exchanged with the Server are exclusively associated with that User
- **Message Freshness:** There is a unique identifier (nonce) in each message exchanged between the User and the Server to prevent repetitions of the same message. This way a file cannot be reverted to an older version nor maliciously modified.

Proposed Solution



With our solution, we aim to have a secure communication between a registered User and the Server, for this we will also use a Database to keep record of all registered Users and authenticate their logins. This DB will also keep a record of the Documents that exist on the Server with the ID of the Owner and the Contributors of said Document. Not only does the Server keep the Documents saved in it for the User to access them remotely, the User can also have them saved locally for faster and easier access, creation and editing.

Basic Solution

For this part, we will create the basic structure of the Server, User and Database to have everything up and running allowing the communication between Server and User, being then able to create and edit Documents remotely or locally, with manual synchronization. The notion of Contributors will also be made available. And an Authentication mechanism will also be implemented (login from the User to the Server) to ensure **Authenticity**.

Intermediate Solution

On this stage, to ensure **Integrity**, we are proceeding with permissions for accessing the Documents the Users are owners of, or contributing in. For this to come to fruition, a message digest [SHA-256(sender+nonce+fileHash)], will be used, to ensure that **Non-Repudiation** exists, and a nonce UUID v4 for **Freshness**, in the exchange of messages and Documents between the User and the Server. We will also encrypt the Documents on the Server to provide **Confidentiality** from Users who are not permitted to access it. We assume that the key used to encrypt the Documents is safely secured only accessible by the Server application. All actions done on the Server are saved on a log.

Advanced Solution

In this last part, there will also be implemented an automatic synchronization between Server and User for the Documents. We'll also take steps to guarantee that the Key used to encrypt the Documents on the Server is on a folder only accessible by the Server application and no one. To further enforce the existence of **Non-Repudiation**, we will now implement an association of Keys. Each User has its own Private Key and the Server has the Public Key of all Users. These will be used to encrypt the messages exchanged from the Users to the Server, and they will be distributed on a secure communication channel. Another entity, called KeyGiver will generate these Private and Public Keys. The message digest used in the intermediate solution will also now be encrypted with the sender's private key.

Tool References

Python: https://www.python.org/

JSONRPC for Python: https://github.com/joshmarshall/jsonrpclib/

Crypto for Python: https://cryptography.io/en/latest/

MySQL for Python: https://dev.mysql.com/doc/connector-python/en/connector-python-installation.html

Work Plan

Week	João	Ricardo	David
22-28 October	Project Proposal	Project Proposal	Project Proposal
31 October–4 November	Implementing Server side base structure	Implementing Server side base structure	Implementing DB and User side base structures
7–11 November	Implementing User side login	Implementing Server side login authentication	Implementing Doc permissions on the Server and DB
14–18 November	Same as the previous week	Same as the previous week	Same as the previous week
21-25 November	Implementing automatic sync	Implementing hidden Server Key	Implementing KeyGiver and Keys exchange
28 November–2 December	Same as last week and testing	Same as last week and testing	Same as last week and testing