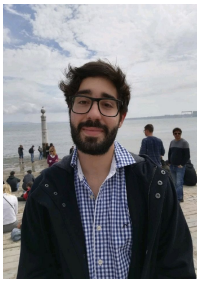
Smartphone as a security token

Network and Computer Security

Group A31

Tomás Silva
83862



Ricardo Gueifão
87699

Samuel Vicente
87704

1. Problem

The chosen topic is regarding a smartphone as a security token. In this scenario, a bank has three internal servers where one is for the critical system of the bank (like transactions, login and changes to the profile), one is for non-critical systems checking the bank statements) and the other is for communication with the exterior. The exterior corresponds to the web application and the smartphone.

The main security problem here that needs to be solved is the access to the critical server of the bank that needs to be controlled, not allowing attackers to gain access to this system. Another problem that needs to be addressed is the confidentiality of the information that is passed between the internal servers to the external entities. That said, all the connections among the internal servers and the external entities must be secure.

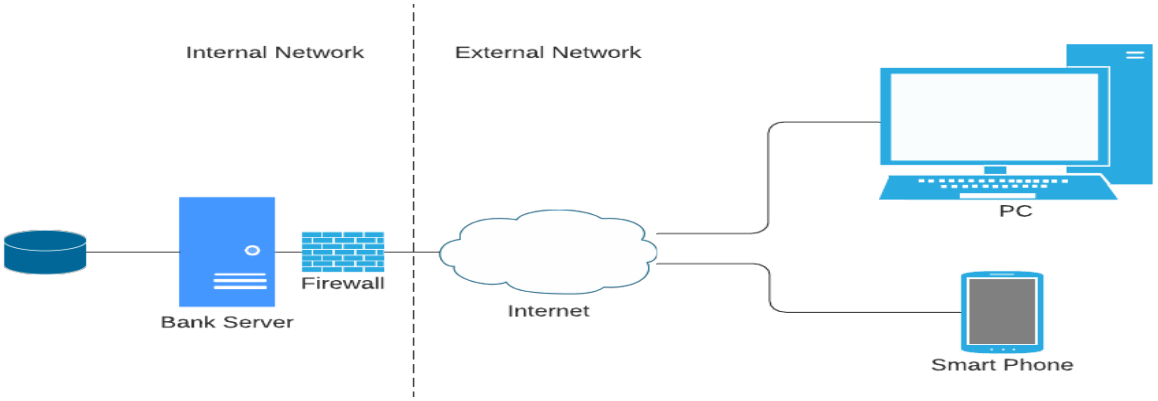
1.1. Requirements

- **Confidentiality:**
  - The data passed between the server and the client (both in the web application and in the smartphone) must be encrypted
- **Integrity:**
  - The data passed between the server and the client (both in the web application and in the smartphone) can not be tampered with and should be preserved.
- **Authentication:**
  - To connect to the server of the bank the user must perform two-factor authentication
- **Non-repudiation:**
  - The critical data or operations passed between the server and the client (both in the web application and in the smartphone) must be signed
- **Freshness:**
  - The data passed between the server and the client (both in the web application and in the smartphone) must have an incremental nonce.

1.2. Trust assumptions

Regarding the trust between the entities of this system, we assume that our server can partially trust the client, both in communications with the smartphone and with the web application. We trust the client **when** the client follows our protocol.

2. Proposed Solution



2.1. Overview

Our solution consists of three elements, a server that handles the bank operations (check balance, withdraw and deposit) and a client that connects to our server via a web application that wants to perform these operations and the client smartphone that acts as a security token.

In order to login to our web application, the client needs to perform two-factor authentication (something you have: the password to log in and something you are: the biometric fingerprint that the smartphone reads). When authenticated the user gains access to the account and it can perform different operations. We divide these operations into two categories: critical and non-critical. The non-critical operations are read-only operations (check balance, check past transactions and check profile), the critical operations are the ones that change the account (perform transactions, change profile information), to perform these operations the user must grant authorization using the smartphone.

2.2 Deployment

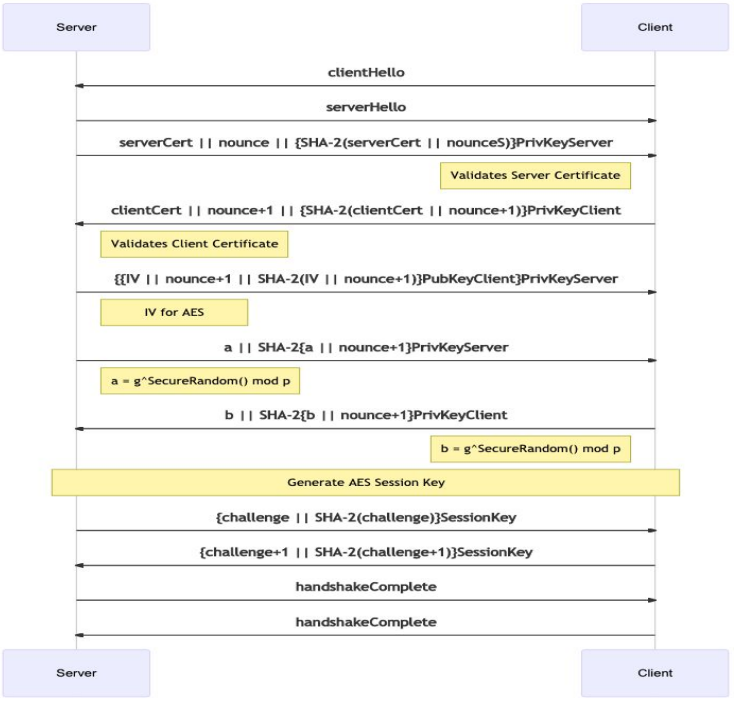
The server will be deployed in a virtual machine, this serves the web application and performs the operations, it also handles the authorization protocol with the smartphone.

2.3. Secure channel(s) to configure

The bank server will have a self-signed certificate generated by OpenSSL, the required cryptographic operations will be done using the Java Crypto library in the

smartphone and the Pycryptodome library in the server. To deal with two-factor authentication we will use Duo SDK.

2.4. Secure protocols to develop



Properties

Our protocol offers confidentiality and integrity since every message between the server and the client is exchanged using TLS since we sign every message our solution also provides non-repudiation. To guarantee freshness we use incremental nonces in the messages.

Languages used

Our server will run in python with an SQLite database, the web app will be a simple application also made in python, our mobile application will be made in Java.

Key Distribution

Our servers start with a pair of RSA2048 keys generated and a self-signed certificate. This will be used to authenticate and sign the server messages. The client then generates its own RSA2048 key pair to sign its own messages, this is done to prevent man-in-the-middle attacks when performing Diffie-Hellman's.

The client and the server then, using Diffie-Hellman's will generate an AES key. In Diffie-Hellman, a and b are ephemeral to provide us with perfect forward secrecy.

3. Plan

3.1 Versions

- **Basic:** The server securely communicates with the web application using our protocol
- **Intermediate:** Two-factor authentication and authorization via smartphone isenabled
- **Advanced:** Firewall set up with DDoS mitigation.

3.2 Effort commitments

Tasks to do:

Develop Server - Server

Develop apk - App

Firewall and DDoS protection - DDoS

Testing - Testing

Develop Web app - Web App

Write Report - Report

Prepare Demonstration - Demonstration

Communications - Coms

Write Documentation - Documentation

	16/11 - 22/11	23/11 - 29/11	30/11 - 6/12	7/12 - 11/12
Ricardo	Server	Server	DDoS	Testing
Samuel	App	App	Documentation	Report
Tomás	Web app	Coms	Coms /Demonstration	Testing

4. References

1. Duo SDK
2. OpenSSL
3. Pycryptodome
4. Java
5. Python
6. SQLite