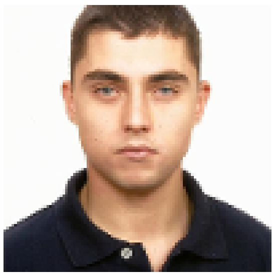
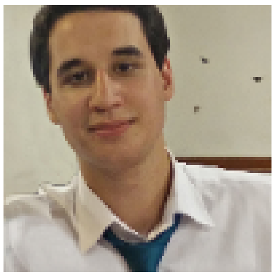
Instituto Superior Técnico — Taguspark

Network and Computer Security

Smartphone as a security token (Proposal)

Group 18


ist173891 — David Gonçalves
david.s.goncalves@tecnico.ulisboa.pt


ist194104 — Leonardo Troeira
leonardo.troeira@tecnico.ulisboa.pt

ist194122 — Francisco do Carmo
francisco.baptista.carmo@tecnico.ulisboa.pt

1 Problem

Password-based authentication has the advantage of being simple and convenient to use. However, there are too many attacks that can be performed remotely that render this authentication mechanism not secure enough on its own—phishing, XSS, SQLI, rainbow tables, dictionary attacks, brute force... to name a few.

As such, if we're seeking stronger security mechanisms, one option is to make use of multi-factor authentication, usually involving knowledge and possession. This poses a trade-off between security and usability however, since the user will be needing to carry something with him all the time (e.g. smartphone).

Besides authentication mechanisms, we're also faced with the challenge of data possibly being stolen, either by sniffing network packets or acquiring control over the victim's file system, among other possible attacks. Therefore, assuring that data is stored and transmitted securely (relying on encryption algorithms) is a must. And again, for additional security, we should have another factor of authentication.

1.1 Requirements

1. The web server must have mechanisms to prevent XSS and SQLI.
2. Communication between client and server must be done through a custom secure channel.
3. The system must provide two-factor authentication.
4. User files must be stored encrypted.
5. User files must be decrypted only if the user has their smartphone nearby the client application.
6. The system must provide fault tolerance. If the system crashes after decrypting the files, make sure that the files are encrypted again.
7. Communication between the smartphone and client application must be done using a secure protocol.

1.2 Trust assumptions

Our problem assumes that the network and user's device cannot be trusted. However, we assume that the user's device is not compromised while it is in use, and that the smartphone and server are not compromised either. We also assume that the server's certificate is trusted.

2 Proposed solution

For our solution, we shall create a simple website—on a server machine—with a register and login forms, for which we shall obviously sanitize user input and make use of SQL parameters in order to prevent XSS and SQLI respectively. The user's password shall be stored hashed and salted in order to prevent it from being easily obtained by potential attackers. As an additional security layer for authentication, we shall use 2FA with the help of Duo SDK, thus leveraging the smartphone's proximity to the user as part of an increased security solution. The server shall also log any user actions.

The user shall be able to download files, from the server's website, which come encrypted with a symmetric key. The user's key pair shall be generated at the registration phase, and they shall be stored in the user's smartphone, which will necessarily need to be connected to the user's device in order to be able to receive it, and we shall make a simple Android app for this purpose. The smartphone and user's device shall connect through Wi-Fi, utilizing a secure protocol (e.g. WPA2/3). For extra security, the user's private key should be symmetrically encrypted with a password, which should be different than the user's login password (e.g. KeyStore). The symmetric key used for each file shall also be stored in the smartphone encrypted in the same fashion as the user's private key.

We shall create a simple client application for viewing these encrypted files that were downloaded from the server. In order for the client application to decrypt these files, the smartphone must be connected to the user's device. The system shall be fault tolerant—as soon as this connection is lost, the client application shall clean its state, thus the user shall no longer be able to view the encrypted files. We plan on using a stream cipher for this purpose, as in theory we can have large files that don't fit in volatile memory.

For registration and login, an AES key shall be exchanged using a secure protocol (e.g. Diffie-Hellman). Subsequent communication between the user and the server shall be done using symmetric encryption with this key, in order to guarantee confidentiality. For integrity, each message shall be hashed together with random padding (using SHA-2) and for authenticity this hash shall be signed with the sender's private key.

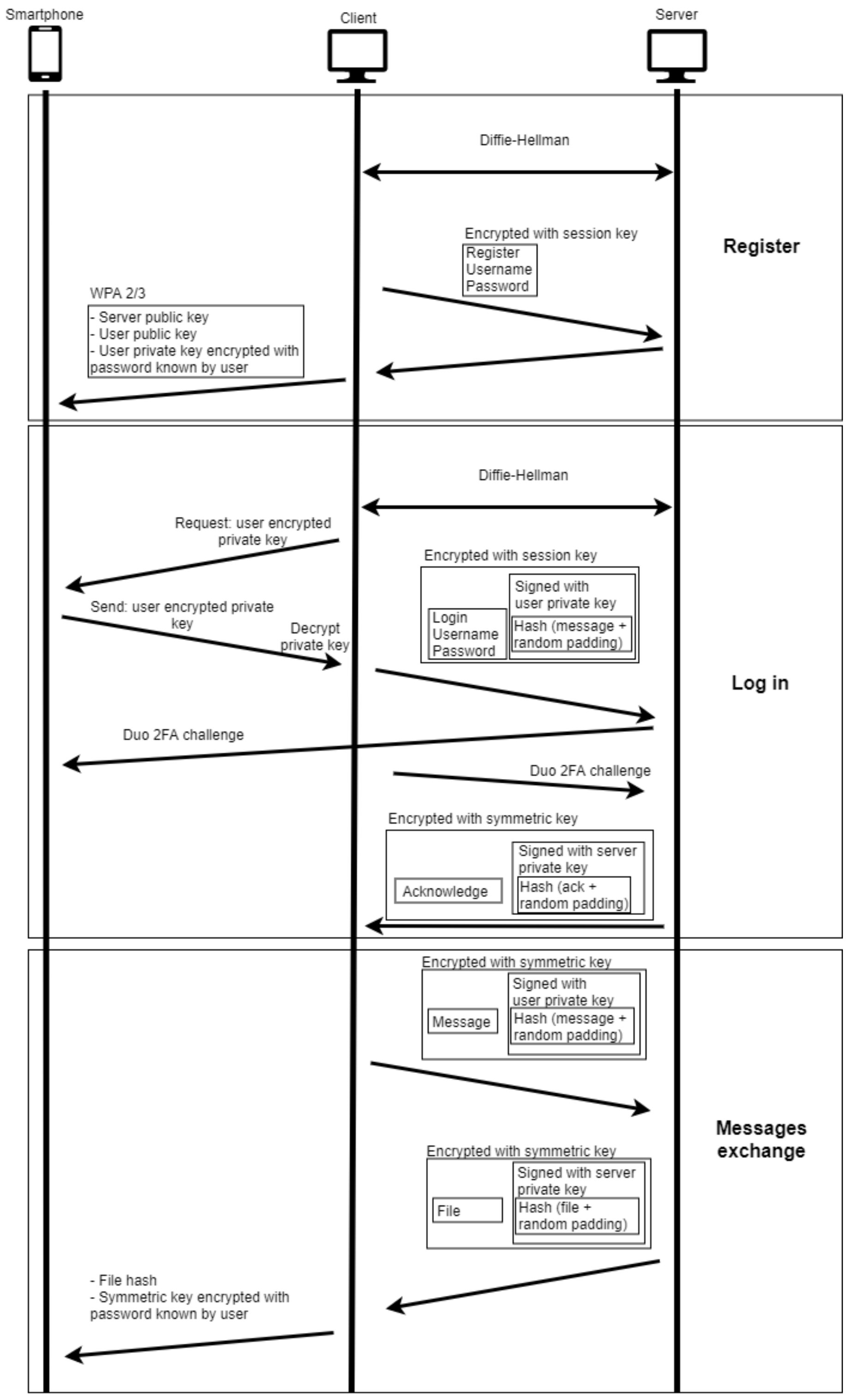

Smartphone
Client
Server
Diffie-Hellman
Register
Encrypted with session key
Register
Username
Password
WPA 2/3
- Server public key
- User public key
- User private key encrypted with password known by user
Diffie-Hellman
Request: user encrypted private key
Encrypted with session key
Signed with user private key
Login
Username
Password
Hash (message + random padding)
Send: user encrypted private key
Decrypt private key
Log in
Duo 2FA challenge
Duo 2FA challenge
Encrypted with symmetric key
Signed with server private key
Acknowledge
Hash (ack + random padding)
Encrypted with symmetric key
Signed with user private key
Message
Hash (message + random padding)
Messages exchange
Encrypted with symmetric key
Signed with server private key
File
Hash (file + random padding)
- File hash
- Symmetric key encrypted with password known by user

Figure 1: Secure communication

3 Plan

3.1 Versions

- **Basic:** For our basic solution we shall create a website with a register and a login forms, with mechanisms to prevent XSS and SQLI attacks, that stores passwords hashed and salted, creates a key-pair for each user upon registration and gives it to them. The server shall also be able to encrypt trivial text files—context doesn't matter—with a symmetric key shared with the user at login, and allow the user to download them. We shall then create a simple client application (typically JavaScript on the website) so that these files can be viewed by the user, using a stream cipher.
- **Intermediate:** On top of the basic version, we shall implement 2FA using Duo SDK and add another use for our smartphone, which is proximity detection. The user's key pair, server public key and symmetric key for each file shall be stored in the smartphone, we'll design a simple Android app to take care of this task. The communication with the smartphone shall use a secure Wi-Fi protocol (WPA 2 or 3). We shall also log user actions on the server side.
- **Advanced:** Finally, to complete our solution, we shall guarantee that as soon as the connection to the smartphone is lost or the client application crashes, the encrypted files shall no longer be visible. Additionally, we'll design our own custom secure channel for communication between server and client, as explained in the previous chapter. We'll also encrypt the user's private key and files' symmetric keys using an AES key derived from a password that only the user knows about.

3.2 Effort commitments

	73891 - David Gonçalves	94104 - Leonardo Troeira	94122 - Francisco Carmo
4 Nov - 10 Nov	Create Server/Client Sides	Create Android App	Create Server/Client sides
11 Nov - 17 Nov	Create secure channel between server and client	Configure 2FA with Duo SDK	Create secure channel between server and client
18 Nov - 24 Nov	Add security to data with cryptography	Add security to data with cryptography	Configure standard secure channel between smartphone and server
25 Nov - 1 Dec	Add fault tolerance to server	Add fault tolerance to server	Add fault tolerance to server
2 Dec - 8 Dec	Test System/Correct bugs	Test System/Correct bugs	Test System/Correct bugs
9 Dec - 11 Dec 17H	Write Report	Write Report	Write Report

4 References

- **Duo SDK:** Duo's trusted access solution is a user-centric zero-trust security platform to protect access to sensitive data at scale for all users, all devices and all applications.

  https://duo.com/product/every-application/supported-applications/apis

  Found: Yes

  Installed: No

  Tested: No