Faculty of Sciences of the
University of Porto

Master's Degree in Information Security

Security and Applications in Trusted Hardware

Portable Password Manager

João Silva Rui Fernandes Tiago Garcia

May, 2022

Abstract

This report describes the practical assignment of the Security and Applications in Trusted Hardware course of the Master's Degree in Information Security at the Faculty of Sciences of the University of Porto.

In this report, we briefly describe the application we developed and discuss the decisions we made.

Contents

List of Figures

1 Introduction

Smart cards have become more affordable as their computing capability and storage capacity have increased, opening doors to a wider range of security measures, making them perfect candidates for authentication devices.

Smart cards' security mechanisms, together with their multi-application operating systems, allow for a wide range of uses. They are, for example, frequently utilized in public transportation application, such as Andante [1]. Other use cases include functionalities related to authentication and digital signature (*e.g.* Chave Móvel Digital [2]).

In this report, we explore the use of smart cards in privacy-sensitive applications, namely password managers, provide an overview of existing solutions and discuss why they may not be adequate in certain scenarios. Finally, we propose a solution consisting in a password manager using a smart card simulator.

Structure of the Report

The remainder of the report is structured as follows:

- Section 2, **Problem**, describes the problem of password re-usage across multiple services and addresses existing solution to this problem.
- Section 3, **Proposed Solution**, describes the architecture of our proposed solution to the problem of password re-usage across multiple services, and discusses some implementation details.
- Section 4, **Conclusion & Future Work**, concludes the report and suggests areas of future improvement.

2 Problem

Current authentication schemes are based on something the user has (*e.g.* a smart card or a security token), something the user knows (*e.g.* a password or a PIN), or their physical characteristics (*e.g.* a fingerprint or iris pattern) [3], being password-based authentication the most widely used.

One problem with password-based authentication is that infrequently used passwords are easy to forget, and it is hard to remember secure passwords. In fact, to avoid memorizing numerous passwords, users tend to re-use the same password on different platforms [4]. As a result, discovering a user's password grants an attacker access to multiple of the user's systems. Some solutions address this issue by requiring users to remember only one password.

2.1 Existing Solutions

In this section, we describe two different solutions to the problem of password re-use across multiple services, namely Single Sign On and Password Managers.

2.1.1 Single Sign On

User authentication can be delegated to an external service, known as Single Sign On (SSO). The purpose of such services is to authenticate users and securely communicate successful authentication to the services the user wants to use. This way, users only need to remember the password they use for the SSO service. Examples of SSO services include Shibboleth [5].

The main issue with this approach is their low portability and their availability, or lack thereof. For one thing, the service to which the user wants to authenticate must trust the SSO system. If this is not the case, the user still needs to remember a new password for the service. In other words, SSO systems only work with services that trust them.

On the other hand, a service may not trust a SSO system due to the lack of availability guarantees. In other words, in case of a denial of service (DoS), the SSO may be disabled or rendered inaccessible, restricting user's ability to authenticate themselves.

2.1.2 Password Managers

Password managers are designed to store and provide access to a user's passwords. The user only needs to memorize a *master password* and store his passwords in the password manager. Examples of password managers include Bitwarden [6] and KeePass [7].

End-to-end encryption ensures that only the user has access to their master password and password vault – it cannot be accessed by the service provider or intercepted by hackers. However, since no one else can access the user's data, it is crucial that they remember the master password. Moreover, in order to safely store the passwords, symmetrical ciphers (*e.g.*

AES) are used. These algorithms use either the master password, or a transformation of it such as its hash value, as a key to encrypt the stored data. However, other solutions exist. For example, by default, Firefox does not require a master password, using a default key to cipher passwords database.

Finally, despite being more resistant to DoS attacks, password managers are still susceptible to brute-force attacks.

3 Proposed Solution

3.1 Security Requirements

As any password manager, this eliminates the user need to remember multiple passwords. Thus, the following security requirements must be met:

- The password manager should be capable of generating complex, random passwords.
- Recovery of the master key should not be allowed.
- It shouldn't be possible to access passwords without the correct decryption key.
- If the smart card (which contains the decryption key) is lost, it should not be possible to recover data. This limits usability but makes the system more secure.

Unlike other existing solutions, we do not store the passwords in the card itself. Instead, it is only used as a means to perform encryption and decryption, and to generate random passwords. Smart cards are especially well suited for this type of application since they are extremely secure by design, and tampering leads in the destruction of the information stored therein.

Finally, the use of smart cards in this application has the following benefits:

- **Applet isolation:** Different applets can safely coexist in the same card. Each applet is assigned to an execution context that controls access to the objects assigned to it. The boundary between one execution context and another is often called an *applet firewall* [8].
- **Cryptographic Operations:** The cryptographic key stored on the card is very well protected both physically and logically, and are therefore extremely hard to steal. In fact, the card contains security features that enable the protection of sensitive cryptographic data and provides a secure processing environment. Finally, it allows us to perform cryptographic operation without never revealing the key that was used.

3.2 Assumptions

In this work, we make the following assumptions:

- The card is tamper-proof. Thus, it is not possible for an attacker to access the cryptographic key without previous authentication.
- Only the user knows the smart card access PIN. This means that if somebody authenticates with the user PIN, he is the expected user.

3.3 Architecture

The architecture of the proposed solution is illustrated in Figure 1:

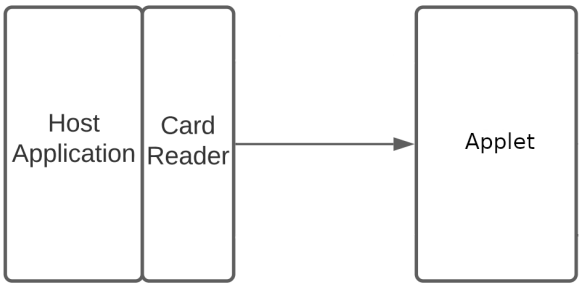

Figure 1: System Architecture

Here, a request-response protocol is used to communicate with the host application, in which application protocol data units (APDUs) are exchanged.

In total, there are three entities:

- **Host application:** Responsible for storing the encrypted passwords (generated by the Java Card) in a SQL database.
- **Card Acceptance Device (CAD):** Responsible for making requests to the Java Card and forward the reply to the host application.
- **Java Card:** Responsible for authenticating the user, generating random passwords, as well as performing encryption and decryption operations.

Essentially, the card support two operations: generating and retrieving passwords, which will be addressed in more detail in the following sections.

3.3.1 Password Generation

To generate a password, 16 random bytes are generated, which are then encrypted using the AES master key. Then, the password is sent to the host application[1], which is stored in a database, together with a description.

This is illustrated in the following sequence diagram:

[1]The AES encrypted password is then encrypted once again using the CAD's public key.

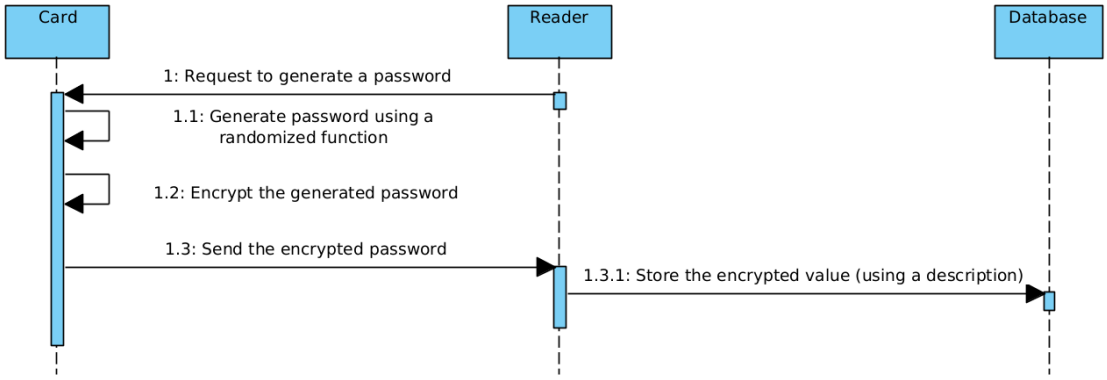

Figure 2: Password Generation

Here, it is worth noting that the password never leaves the smart card. This means that even if the whole database is compromised, the attack cannot decrypt the passwords because he does not have access to the key that was used to encrypt them, which is stored in the Java Card.

3.3.2 Password Retrieval

To retrieve a password, the encrypted password is first retrieved from the database based on the description provided by the user. The Java Card then decrypts this password using the AES master key stored therein, and returns it to the host application. Here, the password is not returned in plaintext – before returning, the card encrypts the password with the CAD's public key. This way, an eavesdropper cannot intercept the password, and the received message is then decrypted with the CAD's private key and, finally, the password is shown to the user.

This is illustrated in the following sequence diagram:

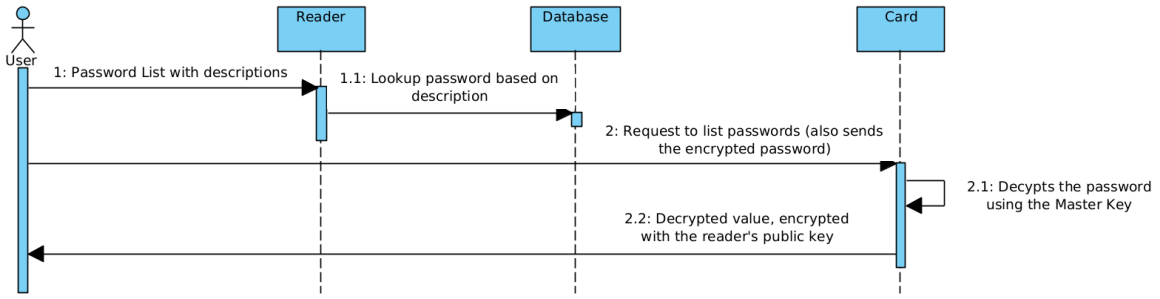

Figure 3: Password Retrieval

3.4 Attacks

As mentioned in Section 4, due to time constraints, communication between the Java Card and the CAD is not encrypted. In an ideal scenario, this would be encrypted using a public key encryption scheme (*e.g.* RSA or ECC), which would prevent eavesdropping in the sense that an attacker, even if capable of intercepting messages, would not be able to make sense of the data being transmitted.

Under these circumstances, because communication is not being encrypted, the system is susceptible to *replay attacks*, as illustrated in Figure 4. In this case, an eavesdropper could capture a message, say msg_1, drop it, modify it, and retransmit the resulting message, msg_1', and the receiver would have no way of knowing that the message has been altered.

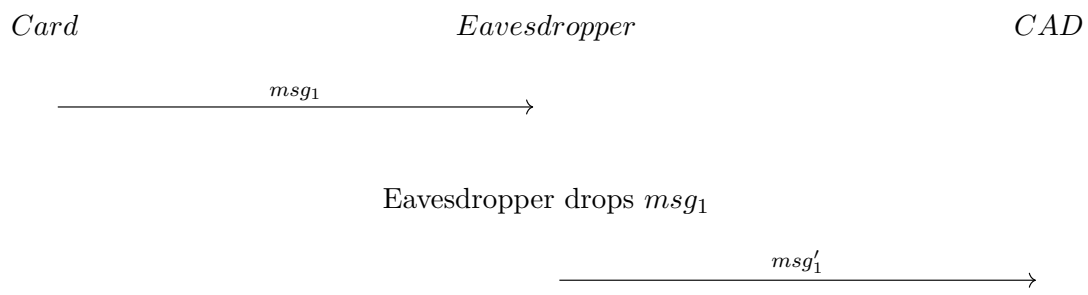

Figure 4: Replay attack

However, as mentioned, it is quite straightforward to solve this problem – it only requires encrypting communication.

4 Conclusion & Future Work

Due to some difficulties in setting up the development environment, we were not able to implement all the functionalities as initially planned.

For one thing, communications between the Java card and the card acceptance device (CAD) are not encrypted. To do this, communication from the Java Card to the CAD would be encrypted with the CAD's public key, which would then be decrypted with its private key. On the other hand, communication from the CAD to the card would be encrypted with the card's public key, which would then be decrypted with the corresponding private key. To achieve this, it would be necessary to use public key encryption schemes such as RSA or ECC.

Additionally, two more approaches could be optimized regarding the implementation of this work. Firstly, the current work is generating a completely random password, meaning the bytes generated might not correspond to a classical password (with only letters, numbers and special characters). Secondly, since this work relied only on the simulator, a few limitations were discovered, in particular regarding the random bytes generated by this simulator. The first limitation was that the tool was incapable of supporting the `SECURE_RANDOM` algorithm. The second limitation was that each time the card was booted, the same seed would be inserted into the `PSEUDO_RANDOM` algorithm, which lead to the same passwords being generated every time the card was rebooted.

To conclude, despite such difficulties, we still managed to achieve a functional implementation of a password manager with respect to the specification. This allowed us to have a better understanding of the specificities of developing an application that make use of smart cards.

References

[1] "Transportes Intermodais do Porto," https://www.linhandante.com/, (Accessed on May 05, 2022).

[2] "A Chave Móvel Digital," https://www.autenticacao.gov.pt/a-chave-movel-digital, (Accessed on May 05, 2022).

[3] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2021–2040, 2003.

[4] D. Florencio and C. Herley, "A large scale study of web password habits," Microsoft, Tech. Rep. MSR-TR-2006-166, November 2006. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-large-scale-study-of-web-password-habits/

[5] S. Suoranta, A. Tontti, J. Ruuskanen, and T. Aura, "Logout in single sign-on systems," in *Policies and Research in Identity Management*, S. Fischer-Hübner, E. de Leeuw, and C. Mitchell, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 147–160.

[6] Bitwarden Open Source Password Manager. https://bitwarden.com/. (Accessed on April 14, 2022).

[7] KeePass Password Safe. https://keepass.info/. (Accessed on April 14, 2022).

[8] "An Introduction to Java Card Technology," https://www.oracle.com/java/technologies/java-card/javacard1.html, (Accessed on May 04, 2022).