Secure Password Management Using Smartcards

Nuno Miguel Grilo Fonseca Pinheiro

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Examination Committee
Chairperson: Prof. Mário Rui Fonseca dos Santos Gomes
Supervisor: Prof. Ricardo Jorge Fernandes Chaves
Supervisor: Prof. Carlos Nuno da Cruz Ribeiro
Members of the Committee: Prof. João Pedro Faria Mendonça Barreto

November 2013
The current security model was not designed to protect users from themselves, and this goes a long way towards understanding why security is so difficult.

Rik Farrow
Acknowledgments

This work would not have been achieved without the help of many people.

I would like to thank my coordinators, Professor Ricardo Chaves and Professor Carlos Ribeiro, for their support. Their knowledge and experience provided me the right guidance to achieve this work. Their different views allowed interesting and important discussions which led this work to a better direction.

I would like to acknowledge GOTrust Technology Inc. for providing their Software Development Kit. This SDK allowed me to use and explore their secure microSD java card, without which most of the technical work of this thesis would have never been possible.

It is also important to acknowledge three open source projects which helped the development of this thesis. These projects are jCardSim (Java Card Runtime Environment Simulator), MUSCLE (Movement for the Use of Smart Cards in a Linux Environment) and Virtual Smart Card Architecture. These projects help the world to give a step forward on the creation of secure solutions based on smart cards.

I would like to thank my family and friends for all the strength and support they gave me in times of need. Two people which deserve to be highlighted are my brother and mother. A special thanks to them, for being everyday right by my side.

Last but not least, I would like to thank all who participated in the development of this thesis, either by performing the usability tests, proof-reading, or by discussing the proposed solution.
Abstract

Currently, most user authentication services are based on passwords. To avoid memorization of multiple passwords, users tend to re-use the same password on multiple systems. As such, finding the password of the user gives the attacker access to several systems of the user.

Some solutions mitigate this problem by allowing users to remember only one password. These solutions, such as Single Sign On or hash based password generators, require trustworthy external services or the generation of multiple dependent passwords. These systems are susceptible to DOS and brute force attacks. Another solution is the use of password managers that store passwords ciphered by a master password. Although being more resistant to DOS attacks, usually they are still susceptible to brute force ones. These solutions also provide weak access control after user authentication, enabling access from non-legitimate applications.

The work herein presented proposes a password manager based on smart cards where the passwords are securely stored inside the smart card and can only be accessed after mutual authentication, eliminating the problem of brute force attacks. To perform access control, the chosen policy bases itself on the identity of the applications, avoiding masquerade attacks. To enable users to access smart card solutions without requiring a reader, a virtual smart card is also proposed, allowing users to access smart cards contained inside smart phones.

Keywords

Security, Password Managers, Single Password, Smart Cards
Resumo

Atualmente, a maioria dos serviços de autenticação são baseados em palavras-passe. Para evitar a memorização de múltiplas palavras-passe, os utilizadores tendem a reutilizar a mesma em diferentes sistemas. Como tal, descobrindo a palavra-passe do utilizador é possível a um atacante aceder aos diversos sistemas do utilizador.

Algumas soluções mitigam este problema permitindo ao utilizador recordar-se apenas de uma palavra-passe. Estas soluções, como serviços de autenticação única e geradores de palavras-passe baseados em funções de hash criptográfico, requerem serviços externos confiáveis ou a geração de múltiplas palavras-passe dependentes. Estes sistemas são susceptíveis a ataques de negação de serviço e ataques de força bruta. Outra solução é a utilização de gestores de palavras-passe que guardam estas chaves utilizando cifra com uma chave mestra. Apesar de serem mais resistentes a ataques de negação de serviço, tipicamente são ainda susceptíveis a ataques de força bruta. Estas soluções também fornecem um controlo de acesso fraco após autenticação do utilizador, permitindo o acesso por aplicações não legítimas.

O trabalho aqui apresentado consiste num gestor de palavras-passe baseado em cartões inteligentes no qual as palavras-passe são guardadas dentro do cartão inteligente e podem apenas ser acedidas após autenticação mútua, eliminando o problema de ataques de força bruta. Para realizar o controlo de acesso, a política tomada pela solução proposta baseia-se na identidade das aplicações, evitando ataques de aplicações com falsa identidade. Para permitir aos utilizadores o acesso ao cartão inteligente sem necessitar de um leitor, é também proposto um cartão inteligente virtual que permite aceder, a partir do computador, a um cartão inteligente presente dentro de um telemóvel.

Palavras Chave

Segurança, Gestores de Palavras-Passe, Palavra-Passe Única, Cartões Inteligentes
# Contents

1 Introduction 1

1.1 Motivation .................................................. 2

1.2 Thesis Goals .................................................. 3

1.3 Requirements ................................................ 4

1.4 Document Structure ......................................... 4

2 State of The Art 7

2.1 Single Password Authentication ............................... 8

2.1.1 Single Sign On ........................................... 8

2.1.2 Hash Based Passwords .................................. 9

2.1.3 Password Managers ................................. 10

2.1.3.A Mozilla Firefox ........................................... 11

2.1.3.B Google Chrome .......................................... 11

2.1.3.C KDE KWallet ........................................... 12

2.1.3.D Gnome Keyring ........................................ 12

2.1.3.E Kamouflage ............................................... 13

2.1.3.F Web Solutions .......................................... 13

2.1.3.G Physical Device Solutions .............................. 14

2.1.4 Summary .................................................... 15

2.2 Smart Cards .................................................. 17

2.2.1 Overview ................................................... 17

2.2.2 Smart Card Technologies ............................ 18

2.2.3 Smart Card Formats ....................................... 19

2.2.4 Communication ........................................... 20

2.2.5 Smart Card Access Technologies ....................... 21

2.3 Process Access Control .................................... 23

2.3.1 System Call Access Control ......................... 23

2.3.2 IPC Access Control ..................................... 24

2.4 Conclusions ................................................ 24
# List of Figures

2.1 Smart Card Architecture .............................................. 18
2.2 ID-1 smart card ....................................................... 19
2.3 Subscriber Identity Module (SIM) card .............................. 20
2.4 Go-Trust® Micro Secure Data (microSD) with embedded smart card ................................................... 20
2.5 Apdu format and field size in Bytes ................................ 21
2.6 Personal Computer/Smart Card (PC/SC) Architecture ........ 22
3.1 Password Manager Architecture ...................................... 29
3.2 Message format and protection ...................................... 31
3.3 Authentication Protocol ............................................... 32
3.4 TLV message format and variation .................................. 33
3.5 Access control components and workflow .......................... 35
4.1 Inter Process Communication (IPC) technology between components .................................................. 39
4.2 Hardware Configuration Proposal .................................... 39
4.3 Virtual Smart Card Architecture ...................................... 40
4.4 Proposed Virtual Smart Card ........................................... 40
4.5 Smart Card Get Response ............................................. 41
4.6 Smart Card Get Response in Mobile Scenario ...................... 41
4.7 Smart Card Get Response in Mobile Scenario with optimization ................................................... 42
5.1 Direct Call Benchmark - Credential Creation Operation Performance .................................................. 48
5.2 Direct Call Benchmark - Credential Read Operation Performance .................................................. 48
5.3 Firefox Call Benchmark - Credential Creation Operation Performance .................................................. 49
5.4 Firefox Call Benchmark - Credential Read Operation Performance .................................................. 49
5.5 Ease of use and Speed Summary ...................................... 50
5.6 System Acceptability .................................................... 51
B.1 Security Related APDUs ................................................. B-2
B.2 Functional APDUs ........................................................ B-3
D.1 Authentication Screen .................................................. D-2
D.2 Application Access Permission Screen ............................. D-2
List of Tables

2.1 Comparison of Single-Password solutions .......................... 15
2.2 Comparison of password managers ................................. 16
3.1 Credential Attributes ............................................. 33
5.1 Comparison of Single-Password solutions including proposed solution .......................... 46
5.2 Comparison of password managers including proposed solution ................................. 47
C.1 Credential Attributes Masks ................................. C-2
Acronyms

APDU  Application Protocol Data Unit
API   Application Programming Interface
CPU   Central Processing Unit
CRUD  Create, Read, Update, Delete
EKE   Encrypted Key Exchange
EPROM Erasable Programmable Read-Only Memory
GUI   Graphical User Interface
HTTPS HyperText Transfer Protocol Secure
IPC   Inter Process Communication
microSD Micro Secure Data
ms    Milliseconds
mTAN  Mobile Transaction Authentication Number
PC/SC Personal Computer/ Smart Card
PID   Process Identifier
RAM   Random-Access Memory
ROM   Read-Only Memory
SC    Smart Card
SIM   Subscriber Identity Module
SSO   Single Sign-On
TLS   Transport Layer Security
TLV   Type-Length-Value
TPM   Trusted Platform Module
USB  Universal Serial Bus
VSCA  Virtual Smart Card Architecture
XML  Extensible Markup Language
XPCOM  Cross Platform Component Object Model
XPIDL  Cross Platform Interface Description Language
1 Introduction

Contents

1.1 Motivation .................................................. 2
1.2 Thesis Goals .............................................. 3
1.3 Requirements .............................................. 4
1.4 Document Structure ...................................... 4
1.1 Motivation

In the last years, a rising number of services, both for single machines and distributed systems, has been introduced. These services handle different resources, from private data to public information, requiring a reliable and secure access control system to protect them. For this access control to be possible, an authentication scheme that allows people to prove their identity is required.

Several authentication schemes exist, based on at least one of three factors: something that the user knows, such as a simple pin or a complex password; something that the user owns, such as a physical token; or something that the user is, proven by his physical or behavioural attributes, such as their fingerprints. Knowledge based authentication is the most easily deployed, given the fact that there is no need for extra peripherals to perform its mechanisms. However, since this knowledge has to be memorized by users, they tend to choose weak passwords that can be more easily attacked in comparison to the other existing factors. Possession based authentication usually requires the distribution of one physical peripheral per system, making them harder to deploy and to be accepted by users. A currently popular and scalable alternative to the distribution of such peripherals is Mobile Transaction Authentication Number (mTAN) which consists on the usage of one time passwords received through short message service, which proves the user is in possession of the mobile phone. Authentication based on the physical or behavioural properties of the user also depends on the existence of a physical device to scan a user template confirming his identity. A problem related with this kind of authentication in a remote scenario is that sending this sample over the internet can lead to it being eavesdropped and reused, enabling an attacker to steal the user identity. In contrast to password or token based approaches, where users can change their password or token, in the biometric scenario users will not be able to change their biological properties. These presented properties lead to password-based authentication being the most commonly used method.

Due to the knowledge based mechanism being the most habitual, a user which accesses multiple systems is often required to share a secret with each of them. Each of these secrets must be memorized by the user, which requires an effort that the user wants to avoid. A common approach to avoid this memorization effort is the re-usage of the same secret among different services. This password re-usage leads to a new problem: in case of password disclosure from one of the services, all of these will be compromised, and an attacker can have access and control over all of the user’s resources. This attack is possible since the knowledge based authentication mechanism assumes that only the user knows its password but, with several services knowing this same secret, this assumption is broken.

The disclosure of the password may be due to different reasons, either user misuse, such as sharing a password for one service with another person, or an external factor, such as an attack to a remote server which stores the password. One recent attack that alarmed the online community was the access to the LinkedIn passwords database. This attack provided access to unsalted hashes of passwords which allowed access to the passwords of the users. Knowing these passwords and other user data, such as their e-mail, attackers can access other services of the users in which...
the same credentials are used. Another problem that increases the risk of password disclosure is that, traditionally, most services do not provide support for third party solutions to perform operations on the behalf of the user. When the user wants to delegate a functionality from a service to a third party system, the user is required to share his password with this system so the third party can authenticate with the user's identity. This sharing process increases the number of systems storing the password. Solutions have been proposed for services to provide access permission to third party software without users sharing their password[? ].

Regarding the problem of users re-using their passwords, several solutions have been proposed to allow users to perform authentication with multiple services without requiring the memorization of multiple passwords. Such solutions can be considered as part of three categories: Single Sign On systems, a centralized third party system which performs authentication on behalf of the systems; Hash-based password generators, which generate unique passwords for different services; and Password managers, applications which manage a protected database containing user's passwords.

This document analyses these solutions in detail, identifying their security concerns, and studies mechanisms that can be used to solve them. Based on these mechanisms, a secure password manager will be proposed.

1.2 Thesis Goals

Understanding the risks associated with the usage of the same password in different services, and the trend for services to use authentication based on text secrets, it is concluded that there is a need for solutions that help users authenticate with different services, providing the same security strength of multiple passwords, but only requiring the memorization effort of a single password.

To cater such a system, this thesis has as its main goal the development of a password managing solution which allows the user to perform authentication towards multiple services, reducing the risks related with password re-usage. In terms of quality, the following properties are those that must be considered when designing the proposed solution and when analysing the existing state of the art:

**Usability**

Most security mechanisms tend to be disabled or not used by the users[? ] if they do not provide a good usability or if they prompt the user for information too many times. The solution must be user-friendly and perform its operations with the smallest user interaction or impact.

**Security**

The solution must assure that the performed authentication towards the different systems can only be performed with the user's authorization. It is necessary for the solution to be secure, since if compromised it would put at risk all the protected passwords.

**Performance**

Adding too many security mechanisms to a system may cause its performance to drop. The solution must have the least performance impact possible.
Portability

The solution main goal is to enable users to authenticate with multiple systems. This implies that the solutions must be designed to be portable, providing easy integration with those systems.

Availability

Since the solution is to be used to assist the user in authenticating towards multiple services, if the system is not available, the user will not be able to authenticate himself. This means that besides assuring the solutions is reliable and will not fail, it must also be remotely accessible or physically transportable for the user to be able to authenticate himself.

The main goal of the work presented in this document is to achieve a solution that fulfills the presented properties and provides a way for users to authenticate with several systems in a secure way, avoiding memorization effort.

1.3 Requirements

Based on the goals of this thesis, formal requirements can be specified. These requirements must be fulfilled by the password manager proposed in this document, namely:

1. The user must be able to authenticate himself towards multiple systems through the presentation of a single password.

2. The user must be able to authenticate himself towards multiple systems without having to share his memorized password.

3. The system performance must not affect the usability of the system.

4. The user must be able to use the system independently of his location.

5. The system must be easy to integrate with other systems.

6. The user interaction must be as simple as possible by automatizing the authentication process.

7. The system must assure user presence to perform its functionalities.

1.4 Document Structure

This document describes the proposed password manager solution which uses state of the art mechanisms to ensure a secure authentication with multiple systems.

To describe the proposed solution and the main gains from using it, this document is structured in the following chapters:

Chapter 2 presents the state of the art analysis used as the base to design the proposed solution. This analysis consists of: single password authentication solutions, which have the same goal as the proposed solution, which is to allow a user to authenticate towards several systems by memorizing only one password with the same security strength as multiple password memorization;
smart cards, cards with secure processing and storage capabilities; and process authentication, mechanisms that, besides user authentication, allow resources to have an access control based on which application is trying to access them.

Chapter 3 describes the proposed solution, a password manager based on smart cards, its architecture and how the desired goals are achieved, taking into account the existing state of the art herein presented.

Chapter 4 discusses the implementation details of the proposed solution. It describes which technologies are used to implement each component of the proposed architecture and their role. It also describes the implementation of a solution which allows access to a smart card from a computer through a smart phone instead of a smart card reader.

Chapter 5 performs the evaluation of the proposed solution. Based on its the design and implementation, this chapter describes how the desired goals and requirements are achieved. It also compares the proposed solution with other single password authentication mechanisms and solutions in the state of the art.

Chapter 6 concludes this document. It summarizes the developed work and introduces further work directions that can be developed based on the proposed solution.
State of The Art

Contents

2.1 Single Password Authentication ........................................ 8
2.2 Smart Cards ............................................................... 17
2.3 Process Access Control .................................................. 23
2.4 Conclusions ............................................................... 24
This chapter covers all the research performed to design and implement the proposed solution. Section 2.1 presents the different existing mechanisms that can be used to perform authentication towards multiple services without the user having to memorize a password for each of these. Special detail is given concerning password manager solutions. Section 2.2 describes smart card technology, namely cards with secure processing and storage capabilities, usually used as a trust component. Section 2.3 presents mechanisms that can be used to perform access control based on the identity of processes, protecting resources with more granularity than access control based on the user.

2.1 Single Password Authentication

A significant amount of effort has been spent along the years to provide secure authentication. However, due to the effort of memorizing multiple passwords, users tend to re-use the same password among different services, leading authentication mechanisms to lose some of its security properties. Passwords are supposed to be unique for each user and, for each user, unique on each system so the compromising of one system does not imply losing security in another. However, most users tend to use the same password on different systems, since it is hard to memorize a unique password for each system. This password sharing among different services can lead to an attacker having access to all user’s resources in case of disclosure of the password from one of the services.

Different solutions have been proposed to allow users to authenticate with multiple systems reducing the memorization effort to a single password. Although having the same goal, these solutions can be divided into three categories covered in this section: Single Sign On systems, solutions which provide a centralized authentication service, covered in section 2.1.1; Hash based password generators, which generate passwords for the user, covered in section 2.1.2; and password managers, systems that store passwords for the user, covered in section 2.1.3.

2.1.1 Single Sign On

When a service requires user authentication, it usually implements its own authentication mechanism, sharing a secret with each particular user. A different approach is to delegate the user authentication to an external service. This type of service, known as Single Sign-On (SSO), consists of a service with the single purpose of authenticating the user and securely communicating a successful authentication to the services the user wants to access. Besides releasing the authentication and secret sharing responsibility from each particular service, SSO solutions allow users to memorize only one password to authenticate with multiple services, reducing the effort of memorizing multiple passwords.

To allow the usage of a SSO system, the service with which the user wants to authenticate must be aware of the SSO system, requiring the service to be designed or adapted to allow delegation of the authentication process. This service must also trust the SSO system implying that if a service does not trust the SSO system of the user, he will still need to memorize a new password for the service, which leads to SSO systems having low portability and only working with systems which trust them.
One reason that can lead to the disuse of SSO systems is the lack of assurance of their availability. In case of a denial of service attack or misbehaviour of the SSO system, it can be turned off or made unavailable, disabling user authentication.

Different SSO solutions have been proposed, such as OpenID and Shibboleth, but none has been widely accepted, still requiring users to access multiple SSO accounts to access different services. This implies that SSO services are not yet truly single-password solutions.

In terms of security, on one hand, the fact that the user password is only shared with one service mitigates the risk of it being disclosed. On the other hand, the main problem of this type of solutions is that, in case the user password is disclosed by an attacker, either by a brute force attack to the password or misbehaviour of the SSO system, all the user services which are in the realm of the SSO system can be accessed.

A popular use case for SSO systems is the centralization of the authentication within services from a company. This is either applied inside corporations where employees have a single account for internal services, or in public services such as Google, where different services can be accessed through the same account. In these cases trust is assumed, since authentication is provided by the same entity as the services.

A new trend is for social networks, such as Facebook and Twitter, to provide SSO services, allowing users to authenticate with third party services through the social network account.

### 2.1.2 Hash Based Passwords

Hash Based Password systems are systems that generate different passwords for different services based on a single master password provided by the user, allowing a user to share different passwords with different services without having to memorize them. When a user registers in one service, he inserts his master password and the name of the service in the password generator. Based on this data, a password is deterministically created. The next time the user wants to access that service, he repeats the generation process, obtaining the password for that given service. Some services can use more parameters than those previously specified, such as the username, adding more entropy to the generated passwords. In most cases, to improve usability, the entire process is automated, only requiring the user to insert his master password.

The algorithms used by these solutions, to generate the passwords, require two properties:

1 - It must be computationally hard to calculate the master password from the generated passwords

2 - The algorithm must be deterministic

The first property assures that in case of disclosure of one of the passwords it is not computationally feasible to discover the other passwords, protecting the access to other user services. The second property allows the user to regenerate all his passwords without requiring the application to store any information. To achieve these properties the generators typically use cryptographic hash functions which already provide these properties by definition.
The main disadvantage of these solutions is the fact that all the user’s passwords are related. This means that if one of the generated passwords is disclosed, an attacker can use dictionary or brute force attacks to disclose the master password, thus allowing the generation of all the passwords of the user and access to all his services. To mitigate the risk of these attacks, some solutions generate the passwords through multiple rounds of hashing or by using computationally heavier hash functions, slowing the process of generating the passwords, ensuring that the attack will also be slower.

The main advantage of these solutions is their portability. These systems can be integrated with any system, as long as the user is free to choose his password. In case there is no extension for the browser the user is currently using, he can use a non-integrated standalone password generator, generate the passwords from this standalone application, and manually insert the passwords on these systems. This also guarantees the availability of the system since the user can run this standalone application in a smart phone or web-based application.

One example of this type of system, containing the presented properties, is HashPass. HashPass features a browser extension that can be used to automatize the password filling process, providing different passwords for different sites. It also offers a website in which users can manually type their master password and one parameter, and generate a password, allowing compatibility with systems that do not contain browsers compatible with the extension.

2.1.3 Password Managers

A password manager is a system which stores the passwords of the user for different systems and helps the user to authenticate without him having to remember all his credentials for these systems. Within Password Manager Systems we can identify two types of system:

1. Systems Internal to applications, for instance, a module of a browser that provides this functionality.

2. Systems External to applications or multi-application systems, which store and deliver this data to different applications.

The most significant benefit from this type of solutions is that their implementation and integration into the applications only depends on the client side of these applications and does not change the authentication scheme of the system. The server side of the applications is independent of the existence of a password manager, and does not require to be aware of its existence.

Since an attacker that has access to the passwords also has access to all the resources of the user, the main property that should be taken into account when designing a password manager is the security of the access control mechanism used to protect the passwords. This access control must consider the identity of the user who is accessing the data, usually proven by a password, and, in the case of external password managers, it should consider which application is trying to access the passwords.
In the following sections different solutions that exist in the market are analysed. This analysis is performed in terms of quality attributes and functionalities, identifying the main properties that must be implemented by these type of solutions, and the main pitfalls on the already existing solutions, with a main focus on their security properties.

2.1.3.A Mozilla Firefox

Mozilla Firefox is one of the most used browsers in the world having the functionality of storing passwords inserted by the user and taking care of the automatic filling of these into forms. To implement this functionality, Firefox contains a module, called nsILoginManager, which is responsible for the secure data management. This module contains a well defined interface which can be used by other components of Firefox to store the data.

To allow the desired functionality, Firefox stores the form in which the user inserted the credentials, the name of the server that provided the page, the username and password fields and their user submitted values. All these fields are stored in a SQLite file and are unencrypted except for the username and password values.

The system provides the user two options from which to choose the key to encrypt the data. The first and default option is the usage of a predetermined key which is inside Firefox’s application code. The second option is for the user to define a master password which will have to be inserted into the system in the first data access of each session. In both cases the algorithm used for encryption is 3DES.

An important point to refer about nsILoginManager is that this module is also accessed by Firefox extensions to store their critical data, existing no physical or logical separation between user and extension data. This means that, if a malicious extension is installed, it will be possible for it to access all the stored data.

The primary functionalities identified in the nsILoginManager interface are adding and removing credentials, modifying credentials, disabling saving for particular pages, checking which pages are disabled and getting all the credentials.

2.1.3.B Google Chrome

Like the previously analysed application, Google Chrome is also one of the most used browsers in the market and also has the same password management and filling features.

The storage is done in a similar way to Firefox: the name of the server which provided the page, the form filled by the user, the username and password fields and their user submitted values are stored into a SQLite file. In contrast to Firefox, the username values are not ciphered, only the passwords values. This means that if an attacker gains access to the SQLite file and directs an attack to the file’s owner, their work is halved, since it now only has to guess the password instead of both fields.
When running in a Windows environment, Chrome hands over the responsibility of ciphering the password value to the operating system through the use of the CryptProtectData function provided by Window's Application Programming Interface (API).

This function encrypts the data with a key related to the users’ system credentials. Since this function is provided to all the applications and depends only on the user session, any application running over the same user session can access this data. When running Google Chrome on a Linux operating system, all the data, including the passwords, is not ciphered.

2.1.3. C  KDE KWallet

Unlike the previous analyzed applications that featured password managing, KWallet is a system whose purpose is to work as a server for other applications that require sensitive data storage.

KWallet is the default sensitive data storage system in the KDE environment. The sensitive data unit is called Wallet and these are provided to the applications after the user explicitly grants them access. Wallets are ciphered with the Blowfish algorithm and different wallets may have different keys. The communication platform used between KWallet and applications is the Inter Process Communication (IPC) system DBUS. DBUS ensures that its services are unique, meaning that there is no way that another application can register itself as being KWallet. It also ensures that all communications between two connected applications cannot be intercepted. However, if the process running KWallet is killed, another service can register using Kwallet's identity, allowing applications to be mislead into trusting this service.

When KWallet is connected by an application, it prompts the application name and the required wallet and asks the user for the password to unlock this wallet. In case the user grants access to the application, it will have access to the credentials contained in this wallet for the current session. This solution does not guarantee that the application requesting the wallet is legitimate and another application with the same name can mislead the user.

The most important operations that applications are allowed to perform are the addition and removal of entries from wallets. Although it does not provide an operation to get all the entries such as Firefox’s nsILoginManager, it is possible to obtain entries with queries using wild cards, therefore it is possible to get all the entries of the wallet.

2.1.3.D Gnome Keyring

Like the previous example, Keyring, the default system for the Gnome environment, is a sensitive data manager that can be used by other applications.

The interaction with this system begins at the moment that the user is authenticated by the operating system. Keyring implements a Linux Pluggable Authentication Module that authenticates the user in the Keyring system at the same time that the user is authenticated by the operating system. The communication, just like in the KDE Kwallet, is made over the DBUS inter-process communication platform, guaranteeing the previously explained properties: the Keyring server process is unique and communication between it and the applications cannot be intercepted.
From the moment when the user is authenticated, keyring provides the sensitive data to all applications that request it and are running over the same user space as the user. To ensure this property the system uses one of DBUS’s authentication schemes, which consist in presenting to DBUS one key that is written in a file that is only readable by the user. The data is ciphered with the AES-128 algorithm using as key the SHA-256 hash of the user password used at operating system login.

The system takes special care by storing the deciphered data in non-paged memory. This prevents unencrypted data to be swapped into the disk, avoiding access to this data in case of abrupt system unplug.

To store and retrieve passwords from the system, Keyring allows programs to specify attribute values for the data. The set of stored attributes is defined either by a default schema or program specific schemas. An example of a default schema is for instance ‘Network Passwords’, a template with the attributes user, server, protocol, domain and port. A client program can find a password by providing one or more attributes to the system. Because of such default schemas, the stored data can be used in different applications that are able to recognize the default formats.

2.1.3.E Kamouflage

Kamouflage is a system implemented as a Firefox password manager’s replacement. It focuses on avoiding password dictionary and brute force attacks after the file has been compromised.

To provide a more secure storage, Kamouflage generates decoy passwords based on the user inserted passwords. This way, besides having stored the user set passwords, there are also a set of decoy passwords. To calculate the real passwords set, a master password must be used. The main goal of this system is to increase the effort required by an attacker to discover the true data.

When the attacker discovers the set of decoy passwords, they will be lead to performing online attacks in order to test them. These online attacks can be detected by the services in which these passwords are tested, and security measures can be performed by those.

2.1.3.F Web Solutions

Due to the great mobility of users and the usage of different devices, the ubiquity of their passwords is becoming a necessity.

Three major solutions exist to implement a distributed password manager. The simplest one is the storage of the passwords file in a distributed storage solution. An example of this is the storage of the passwords’ storage file identified for Firefox on a distributed file system. The risk associated with such a solution is that, besides having the problems regarding the used password manager, the passwords’ file may also be accessed by an attacker if the distributed storage system is compromised. A second solution consists on the implementation of a browser extension that stores the data in a remote server. To be functional this solution requires the existence of a compatible browser. An example of this type of application is Norton Identity Safe. The last solution, and the most portable one, is the implementation over javascript, interacting with all the compatible browser and storing the data in a remote location.
Two existing applications, Fortnotes and myLogin, belong to this third model of implementation and both have a similar behaviour. In these solutions, the access to the data requires two memorized passwords, each with a different purpose. The first password is the password used by the user to authenticate himself towards the server. After authentication is successfully performed, the password database can be accessed. For the user to be able to read this database, he must decipher it using his second password. This key is only used to perform cryptographic operations and is never sent to the server. By having two passwords, the system is able to guarantee user authentication without needing any transformation of the cryptographic key stored on the server. In case of a successful attack to the server, the attacker will have to perform brute force attacks to the password databases, not to the keys. Besides these qualities, all communication is performed over HyperText Transfer Protocol Secure (HTTPS), providing an extra layer of security.

2.1.3.G Physical Device Solutions

In most classic password manager solutions, passwords are stored in the user’s computer protected by a password. This password is usually short and weak, allowing attackers to perform successful brute force attacks to the file containing the passwords. To avoid granting the attacker the opportunity of performing such an attack, some solutions purpose the storage of the passwords inside physical secure storage devices, such as smart cards, or through their ciphering with stronger keys stored in a different device.

IDKeeper is a password manager that stores passwords inside a smart card contained in a Universal Serial Bus (USB) key that can be accessed through a browser extension. To access the passwords the user needs to authenticate using a pin. Based on this pin, the browser extension and the USB key generate a session key to encrypt the messages containing the transported passwords.

PwdCave proposes a password manager that uses Trusted Platform Module (TPM), a device with processing and secure storage capabilities included in most modern computers. Besides storing the passwords inside the TPM protected storage, instead of returning passwords to applications, all the security operations are performed by the TPM and only the result of password related computations is returned to the application. This method avoids unprotected passwords to be temporarily stored on memory, avoiding some eavesdropping attacks.

Although not secure storage based, Tapas proposes a password manager based on dual possession. In this solution, the passwords can only be accessed when two devices are present, being these devices the personal computer and mobile phone of the user. Since the user can be authenticated towards each device through the presence of the other, the user is not required to insert a password. The passwords are stored, encrypted, on the mobile phone, while the key to decrypt them is stored in the computer, and both components need to cooperate to provide access to the data. If one of the devices is lost, an attacker will only have access either to the encrypted data, or to the cryptographic key. The encrypted data is assumed to be protected, since the key used to cipher the data can be large and strong, as opposed to passwords memorized by users. The communication

Fortnotes main page http://fortnotes.com
myLogin main page https://www.my1login.com/
between both devices is performed on top of Transport Layer Security (TLS), and the session keys are exchanged through an authenticated and secret out-of-band channel. A problem that exists in this solution is that the password manager can only be used in one computer, the one which has the key to decrypt the passwords.

### 2.1.4 Summary

The main goal and the most significant security property in the analysed single-password authentication solutions is the way they defend themselves when one of the user’s passwords is disclosed to an attacker. Table 2.1 presents a summary on how these categories tolerate this disclosure, evaluating password managers in situations when an attacker may or may not have gained access to the ciphered database.

<table>
<thead>
<tr>
<th>Category</th>
<th>Discovery of one password</th>
<th>Availability</th>
<th>Portability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO</td>
<td>Access to All Systems</td>
<td>Requires connection to authentication service</td>
<td>Server side integration</td>
</tr>
<tr>
<td>Hash Based</td>
<td>Allows brute force attack to master password</td>
<td>Only dependent on the algorithm</td>
<td>Portable</td>
</tr>
<tr>
<td>Password Managers without access to Database</td>
<td>Disables attacks</td>
<td>Requires access to database</td>
<td>Client Side integration</td>
</tr>
<tr>
<td>Password Managers with access to ciphered Database</td>
<td>Allows brute force attack to master password</td>
<td>Requires access to database</td>
<td>Client Side Integration</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of Single-Password solutions

It can be concluded from this table that password managers offer the best security properties when there is no access to the database, offering almost the same properties as hash based password generators in case of access to the database and one stored password. This table also compares the availability of the different categories. SSO solutions require the availability of the authenticating service, which can be compromised. The availability of a password manager depends on the access to his database, being dependent on its implementation. Hash based password generators tend to be the most available, due to the fact that the generation algorithm can be implemented in multiple devices and web services, and is available to the user in any situation. Another important property to be considered is their portability, SSO being the least portable due to its requirement of server awareness and integration. Hash based password generators are the most portable given the fact that passwords can be generated in any device, although their usability is improved through client side integration.

Table 2.2 summarizes important security properties of some of the analysed password managers. In this table the key type column defines the length of the key used to protect the data, assuming
<table>
<thead>
<tr>
<th>Password Manager</th>
<th>Key Type</th>
<th>Brute Force Attacks</th>
<th>Availability</th>
<th>Applications Access Control</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Chrome</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>Based on user space</td>
<td></td>
</tr>
<tr>
<td>KWallet</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>Based on user permission</td>
<td></td>
</tr>
<tr>
<td>Keyring</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>Based on user space</td>
<td></td>
</tr>
<tr>
<td>Kamouflage</td>
<td>Password</td>
<td>Offline but requires testing the password</td>
<td>File Synchronization</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Web Solutions</td>
<td>Password</td>
<td>Offline in case of system attack</td>
<td>Online</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>IDKeeper</td>
<td>Pin</td>
<td>Online</td>
<td>Card Transportation</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Tapas</td>
<td>Strong Key</td>
<td>Offline</td>
<td>Key Synchronization</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2.2: Comparison of password managers**

passwords and pins are weak keys. The brute force attack column defines how brute force attacks must be performed to try to access the passwords. Offline attacks refer to attacks in which the encrypted passwords can be accessed and an attacker can try every key to decipher them. Online attacks require the attacker to test each key against the system meaning that, not only is the attack slower, but also that security policies can be defined to block access to the passwords after a certain number of tries. The applications access control column refers to how the access control is performed when an application wants to access passwords stored on external password managers.

In most solutions, if the device that stores the database containing the ciphered passwords is lost, the database can be accessed, enabling offline attacks to be performed on it. Opposed to online attacks, which have their speed reduced to the speed of the system being attacked, these databases can be copied to additional and more powerful machines, increasing the speed of offline brute force attacks. Although a simple attack to web solutions would be an online brute force attack to the account of the user, if the system were to be exploited, and an attacker gained access to the data contained in the server, the attacker would be able to perform offline attacks to the databases.

The availability of most of these systems depends on the existence of a file containing the passwords database. This file can be synchronized either through a cloud storage service or through physical transportation, for instance a [USB] pen drive. In both these cases, the copy of the file to such solutions leads to an augmented risk of the database being accessed to, since this file is shared in more locations.

Although not an external password manager, Google Chrome encryption is performed through Windows functions. The cryptographic key used by these functions is user space based. Any application running on the same user space uses the same key implying that it can access the passwords of the corresponding user. When using both this system or Keyring, if the user is mislead into running any malicious application, this application can access all the data without having to exploit any of the system's privilege escalation vulnerability, since these solutions allow any application running over the user scope to access the data.
In the case of KWallet, although access control is based on user permission, the information provided for the user to make this decision is based on the name of the application. Since an attacker can define his exploit application name, it can mislead the user into trusting a malicious application. This leads us into believing that one important security feature that must be integrated into the system is the ability of specifying explicitly which applications can access the data.

The analysed solutions are only some of the ones that exist in the market, but their design, mechanisms and problems are the same that are found in other solutions.

Despite the problems with existing solutions, password managers are important systems that can be secure, user-friendly and transportable if properly designed. The two following sections present mechanisms to deal with the two major problems identified: protection against offline attacks in case an attacker gains access to the device in which data is stored and control of applications accessing the data.

### 2.2 Smart Cards

In section 2.1.3 we identified several password managers that stored data into a disk or remotely. One of the problems that can arise in these systems occurs when the physical devices in which the data is stored are compromised. If a malicious user has access to these peripherals, he can access the encrypted data. With the increasing performance of cryptanalysis solutions through the usage of special purpose hardware, there is a trend for offline brute force and dictionary attacks to be more effective, turning ciphering with user passwords into a weak protection.

To prevent the occurrence of such attacks it is important to find a way that, based on a password, can protect the user's data with methods other than cipher. This section introduces smart cards as a way to solve such problems.

An overview of smart cards, their architecture, and their main properties are discussed in section 2.2.1. Section 2.2.2 details how these smart cards can be programmed to implement the desired behaviour. Given the broad use of smart cards in a novelty of applications, these exist in several formats, presented in section 2.2.3. Section 2.2.4 describes the messages that must be used to communicate with smart cards, followed by section 2.2.5 that covers technologies which are used by computer applications to communicate with these smart cards.

#### 2.2.1 Overview

Smart cards are card-shaped computers which contain digital data that is supposed to be read by systems instead of people. In the market both cards with and without chips exist. Cards without chips usually contain a magnetic stripe in which data is written. In the case of cards with chips both memory cards and processor cards exist. The obvious difference between these is that memory cards do not contain processing power. By adding processing capabilities to the card, it is able to authenticate the user or the system with which it is communicating before providing the stored data. It also allows the card to do computation, for instance, signing messages instead of providing the user the secret key.
In the purpose of this text only smart cards with processing capabilities are considered.

The architecture of smart cards consists on a Central Processing Unit (CPU) connected to the Input/Output of the card through which communication with the card is performed. The CPU is also connected to three memories, each with its own purpose. The Read-Only Memory (ROM) is used to store the card operating system and run time environments. Some applications, which are not expected to be replaced on the life cycle of the smart card, can also be stored in this memory. The Random-Access Memory (RAM) is used to temporarily store volatile memory to be used during the current session. The Erasable Programmable Read-Only Memory (EPROM) is used to store data and applications. This architecture is depicted in figure 2.1.

![Figure 2.1: Smart Card Architecture](image)

The communication with other system can either be done by contact interface with electrical signals, as specified in ISO/IEC 7816, or by wireless communication, as specified in ISO/IEC 14443. These standards, besides specifying the protocols to be used, also specify their physical characteristics such as voltage or wave frequency, depending on the case.

For smart cards to be secure and useful for security solutions they must be tamper-resistant. Most smart cards have anti-physical tampering mechanisms and security functions such as ciphering and hashing functions are implemented by hardware, in a way that avoids side channel attacks.

An important property provided by these devices is their transportability since they fit in a user pocket or wallet allowing it to travel with the user.

Due to their properties, smart cards are a popular solution used for possession based authentication, most of the times being used in parallel with biometric or knowledge mechanisms to provide multi-factor authentication. By using these solutions, the sensitive data, such as the key for possession based authentication or the user biometric template, is stored inside the card and cannot be externally accessed.

2.2.2 Smart Card Technologies

When designing a smart card system it is necessary to analyze the implementation and deployment options in the market.
For implementation it is necessary to define in which operating system or run time environment the application will run. These can be divided into native environments or interpreted environments. Native environments allow applications to run without an interpreter allowing applications to achieve faster response times and less energy consumption. Interpreted environments, although associated with performance and energy costs, allow applications to be portable since their implementation does not depend on a particular hardware. It also allows the development of applications to be simpler, for this type of environment usually provides an application binary interface which implements the most fundamental and generic functions in smart cards applications. This improves smart cards projects in terms of project development time and cost.

One example of a run time platform for smart cards is the Java Card Technology. Java Card is an open source platform that allows applications developed for it to run in a large range of smart card devices. It also allows more than one applet to be deployed on the card. To develop applets, which is the name for Java Card applications, the programmers can use a subset of the Java language. Since applications run in a virtual environment, the process of debugging applications is also simplified, since in the development phase they can be ran into a Java Card virtual machine in the developer's computer. Before deploying the applet, it must be translated into a CAP file which is a specific file format runnable by the Java Card virtual machine on the card.

Although Java Card specifies the file format, it does not specify how to deploy, or how to communicate with the applets in a multi-applet card. One of the existing standards that defines these two issues is Global Platform, an open and interoperable infrastructure for Smart Cards, devices and systems.

2.2.3 Smart Card Formats

The market has evolved and different types of smart cards were created to enable different solutions. One property of the cards that had to be taken into account was their size and format, while keeping communication standards. ISO/IEC 7810 takes this into account and specifies different card formats.

The typical format for smart cards is the one used for banking and ID cards. Different cards from different providers and for different purposes usually compromise to this standard size (85.60 X 53.98 mm and 0.78mm thickness) specified in ISO/IEC 7810 as ID-1. Having a standardized market allows companies to be independent on specific suppliers while developing a system. This smart card format is depicted in figure 2.2.

![ID-1 smart card](image)

Figure 2.2: ID-1 smart card
Other formats are specified in ISO/IEC 7810 as ID-000 which are Subscriber Identity Module (SIM) cards. SIM cards have a size of 25 X 15 mm and a thickness of 0.78 mm. These cards are specially designed to identify and authenticate subscribers on mobile devices, especially mobile phones. Other examples of usage of this kind of cards are mobile network adapters. An example of a SIM card is depicted in figure 2.3.

![SIM card](image)

**Figure 2.3:** SIM card

With the increase of the smart phone markets, the interest in applications with smart cards for this kind of devices has also increased. One problem is that usually these devices do not have external digital interfaces preventing them from communicating with ID-1 smart cards. Although most mobile phones already contain a smart card in the SIM format, these usually do not allow the installation of custom applications and are closed platforms when provided by telecommunication companies. To solve this problem, a new type of smart card was introduced. This solution consists in a smart card embedded in a flash memory Micro Secure Data card (microSD) card. This type of cards is compatible with most smart phones in the market and can be deployed either closed with a single application or open for the user to install multiple applications. They can also be accessed from the computer through a microSD card reader. One example of this type of smart cards is GoTrust® Secure microSD Java, depicted in figure 2.4.

![GoTrust microSD Java](image)

**Figure 2.4:** GoTrust® microSD with embedded smart card

Depending on the use case, some solutions also distribute smart cards embedded in other devices such as USB sticks. An advantage provided by such a solution is the fact that the USB stick, besides containing the smart card, also provides direct communication, not requiring a smart card reader.

### 2.2.4 Communication

To communicate with smart cards, a specific type of message must be used. These messages are on the Application Protocol Data Unit (APDU) format, as specified by ISO/IEC 7816-4. The protocols used to communicate with smart cards are client-server based, where the card is the server, which only answers to requests from the client.

The messages sent from the client must be in the request APDU format as follows:
• CLA - 1 byte which represents the class of the instruction to be executed
• INS - 1 byte representing the instruction to be executed
• P1-P2 - 2 bytes used as parameters for the instruction
• LC - 1 byte representing the size of data sent in the message
• DATA - LC bytes with instruction specific data
• LE - 1 byte representing the maximum number of expected bytes on the response message

The messages sent from the card, the responses, follow the following format:

• Data - up to LE data bytes as specified in LE
• SW1-SW2 - 2 bytes representing the result of the message, being 0x9000 the code for a successful execution

Since LC is only one byte, the data field can only contain up to 255 bytes. This means that protocols used to communicate with smart cards need to be optimized so as much information as possible can be sent in this limited space. Some cards also implement a slightly different type of APDU, called Extended APDU, which can support more data in each message. In this document we only consider common APDUs since they are the general case.

Figure 2.5: Apdu format and field size in Bytes

Global platform defines specific APDUs to manage the life cycle of the card, such as installing and selecting applets.

2.2.5 Smart Card Access Technologies

Besides applications installed inside smart cards, systems which use smart cards also require an application to be installed on a host which communicates with the smart card. These applications deployed on the host usually have goals which need to have security properties but since they are deployed on systems which are not trustworthy, security functionalities are delegated to smart cards. The host in which they are deployed may either be a personal computer or a standalone terminal, and they must have a smart card reader either attached or integrated. In the case of the host being a personal computer, different applications may require access to the smart card and this smart card may be connected through a reader provided by any manufacturer.
Different solutions have been designed to provide an abstract access to smart cards. These technologies provide a common interface to communicate through readers from different manufacturers, allowing a better portability of the developed applications. Besides this property, these solutions also provide transactional support to ensure that an application can have exclusive access to a card.

One specific solution covering these goals is Personal Computer/Smart Card (PC/SC). PC/SC is an open specification, and there are several implementations for different operating systems. PC/SC achieves the required goals by specifying an architecture which has as main component a centralized resource manager. When a smart card aware application wants to communicate with smart cards, the messages must be sent to the resource manager. This component is then responsible for forwarding the messages to a reader handler. Reader handlers are PC/SC drivers for specific smart card readers, which must be distributed by each manufacturer. A reader handler receiving a message can then forward it to its specific reader, which will deliver the message to the smart card. The reverse process is then performed to deliver the response to the requesting application. Having different reader handlers for different smart card readers, allows the applications to be unaware of differences that may exist in the behaviours of different readers, and also permits applications to communicate with smart cards through different interfaces, such as NFC and RFID. Figure 2.6 depicts a simplified vision of the PC/SC architecture. The messages transported by PC/SC must be according to the APDU format specified in 2.2.4.

![PC/SC Architecture](image)
2.3 Process Access Control

In classic operating systems access control is performed based on the user’s identity. When a user launches a process, it will have all the permissions associated with the user, and all the processes spawned by it will be granted those same permissions. Even if a system is protected by authentication, there may still be threats after the user is authenticated. In a scenario where a user would be required to insert a password to allow him to access a ciphered disk, and if a malware were to run on the user’s computer, this malware could access the contents of the disk since the user is authenticated. In some scenarios there may be a need for stronger policies to control access to resources.

In section 2.1.3 two multi-application password managing systems were presented, Keyring and Kwallet. One possible vulnerability found in these systems was the weak access control over which processes can access the passwords. Keyring allows any application running over a user space to access its corresponding data, and KWallet requests the user’s authorization by displaying an application name defined by the application. Both these solutions partially solve the problem but can still be maliciously explored by having the user run a malicious application. This access control is similar to that which is provided by classic operating systems.

A different approach that can be used to provide a better access control, is to grant access to resources only to applications defined by the user. This type of policy is performed by modern operating systems such as Android, which requires permission from the user for applications to share resources.

Since processes running over the same user space have access to the same files, it is not possible to provide an authentication based on a secret key, since this key can be accessed by multiple applications.

This section presents mechanisms that can be used in classic operating systems, such as Linux, to provide an access control scheme based on the identity of the process. These mechanisms can be either at system level, as presented in 2.3.1 or can be at application level when using an IPC mechanism, as presented in 2.3.2.

2.3.1 System Call Access Control

When an application launched by a user runs in an operating system, it runs in a special mode called user mode. In this mode, applications access only a limited set of instructions provided by the computer, and are unable to access several resources, such as files and hardware devices. To have access to these resources, applications must use system calls provided by the operating system. The operating system performs access control when a system call is performed to assure the process has permissions over that resource.

Several solutions have been proposed to allow programmers and system administrators to expand or override the existing access control policies on the operating systems.
System call interposition is a technique that consists in intercepting system calls in place of the operating system. To achieve this, an application, which must be running in kernel mode, must change the handler pointers that exist in the system call table. This way, besides being able to change the access control policy, the application may also change the behaviour of system calls.

Other solution are Linux Security Modules, modules that register with the operating system the intent of controlling certain system calls. When such system calls occur, the operating systems calls these modules which may grant or deny access to the call.

Both these mechanisms have been used by several solutions to provide sand boxes and access control policies.

In these mechanisms, since they are required to run in kernel mode, one can access information about the process such as the executable used to launch it. From this information a module may confirm the identity of the process and control the access based on it.

These solutions may affect the performance of the systems since every system call will have to be processed by the interposition module.

2.3.2 IPC Access Control

Besides access control at system level, when two user space applications are communicating there may be a need for one of those to attest the legitimacy of the other. To endorse this legitimacy the processes must be able to identify each other, and based on this identification, be able to confirm some properties that will sanction the legitimacy of the process.

To identify a process, the used IPC mechanism must have functionalities to reliably know the Process Identifier (PID) of the processes communicating, which represents a unique identity of the process in the machine. An example is D-BUS which provides functions to access the PID of the communicating parties and the PID of registered services.

To confirm the desired properties of a process based on its PID the operating system must provide functionalities which, based on this PID allow applications to access information of the process. Over the Linux operating system, this information can be accessed through Procfs, a virtual file system containing information on the running processes. Procfs, usually mapped in '/proc', contains folders whose names are the PID of the process they map. Inside each of these folders, different information on the respective process can be found. Examples of this information are the command used to launch the process and a copy of the executable at the time when the application was launched. With access to this information, an application may change its behaviour based on the properties of the communicating process.

2.4 Conclusions

This section analyses the set of solutions in the state of the art which assists in the user authentication towards different systems without having to memorize more than one password. It presents the Single Sign-On mechanism which solves the problem of having multiple credentials, while intro-
ducing problems such as the lack of trust in the authentication server and single point of failure. It also presents Password generation mechanisms which have as their major advantage their availability and the generation of passwords which are not weak or vulnerable to dictionary attacks. These solutions still have a problem, which is the fact that they create passwords that have are related to one another, thus enabling the derivation of all the user’s passwords in case one of them is discovered.

Following, some password managing solutions were presented. These solutions store and provide the users their password when required, enabling them to choose different passwords without requiring their memorization. The problem in most systems belonging to this type of solution is the storage of the data in non-secure devices, allowing the data to be subjected to brute-force attacks and access control based on the user space and the name of the application.

Although problems exist in all the solutions, password managers can be considered as the ones that offer the most qualities. They allow the storage of independent passwords, since they are generated by the user, not by an hash function with a common key. Another advantage is that the user credentials are stored in a device belonging to the user, in contrast to Single Sign On systems, where the service failure will make all the dependent services fail.

To solve the two major problems found in password managers, two solutions were presented: the smart card technology, which solves the problem of secure storage whilst also providing transportability; and mechanisms that allow the implementation of an access control policy based on the identity of the applications, which can be used to control which applications are allowed to access the data.
## Contents

3.1 Overview .......................... 29
3.2 Trust Model ......................... 29
3.3 Communication Security ................. 30
3.4 Functional Communication .......... 32
3.5 Process Authentication ............... 34
3.6 Conclusion .......................... 35
As discussed above, being that knowledge-based authentication is the most common method of authentication, it is important for users to use different passwords in different services. The common memorization problem can be mitigated by providing users with single-password authentication services. Password managers help users achieve multi-password strength, with uncorrelated passwords, as opposed to hash-based password methods. Discovering a single password will not provide access to all systems, unless the password database can be discovered. A vulnerability common to most password managers is that, when the encrypted password database is accessed, an attacker can perform brute force and dictionary attacks to find the master password and decipher the passwords for the different systems.

Herein we propose the use of smart cards to perform data protection. Using smart cards, there are two options to protect data, both requiring user authentication. The first option is to cipher data inside the card with a strong key, and store this data outside the card. The second option is the storage of data inside the card itself. By using smart cards to cipher the externally stored passwords, the password database could still be attacked through brute force attacks. However, given the strength of the used key, the computational resources required to decrypt the data will be significantly greater when compared to data ciphered with user memorized passwords. When the passwords are stored inside the card, brute force attacks have to be online, meaning that trying different passwords to unlock the card requires physical access to it. Online brute force attacks can be avoided by forcing a maximum number of authentication tries, after which the card will be blocked or the data deleted. The storage of data outside the card allows for more data to be protected due to the storage size limits of smart cards. Nevertheless, the storage of the data inside the card provides a greater availability since the user can transport its data without requiring other data synchronization mechanisms.

The solution herein proposed consists of a password manager based on smart cards. In this solution, the passwords are securely stored inside the card and can only be accessed after the user is authenticated on the smart card. This mitigates the risk of accessing the passwords since attackers would need to access the card, and authenticate themselves. In this solution, aside from the user authenticating towards the card to prove his identity, the card also needs to be authenticated, avoiding the storage of the password on a non-legitimate card.

Section 3.1 gives an overview of the solution, its architecture, with the main components and their role to perform the desired functionalities. It is followed by section 3.2 with the presentation of the trust model, the assumptions taken by the proposed solution regarding its environment. Section 3.3 takes into account its previous section and defines how the proposed solution can assure a secure communication between the personal computer and the smart card and how the authentication process of the user and the smart card are handled. Section 3.4 describes format of the domain data sent to the card and how its performance can be optimized through the format of the messages. Section 3.5 describes how to perform access control in which client applications can access the passwords and how applications can confirm the identity of the proposed password manager. Section 3.6 concludes this chapter by summarizing the proposed solution.
3.1 Overview

The proposed solution’s main focus is to allow users to authenticate in multiple systems using knowledge-based authentication, by having a different secret for each system but only being required to memorize a master password or pin value. To provide such a functionality, a password manager is proposed. The proposed password manager is external to applications, meaning that it can be integrated and used by several applications without requiring modification in their server side. Instead of storing the passwords on the local hard drive, the proposed solution stores the passwords in a password protected smart card. To disable online attacks to the card, in which an attacker could test all the possible passwords until the card granted him access, the proposed solution blocks the application after a predefined number of attempts.

To perform its functionalities, the proposed architecture, depicted in figure 3.1, consists of two main components. The first component is the password manager server, which runs locally on the user computer. This component is the bridge of communication between the smart card and the different applications. When a user wants to authenticate in a system, applications communicate with the password manager server through an [IPC] channel requesting for the desired data. After these requests, the password manager server is responsible for handling all communication with the smart card through a smart card access technology. The second component is the password manager applet. This component is an application running inside the smart card, and implements the desired behaviours of password storage. Since the smart card is the device protecting the data, it is also responsible for performing user authentication.

![Password Manager Architecture](image)

Figure 3.1: Password Manager Architecture

The proposed solution does not assume the smart card as being physically connected to the card, allowing it to be in any of the presented formats and to communicate with the computer through any channel.

3.2 Trust Model

As herein described, the proposed solution is comprised by several components. Each of these components runs over a certain environment, either over a classic operating system in the case of computer components, or over a smart card run-time environment or operating system. To define the behaviour of the components, assumptions must be taken regarding the systems over which they
run. Besides such systems, it is also important to consider the expected properties of the technologies used to perform communication between components.

The following defines the trust assumptions taken by each component of the considered system.

**Application Assumptions**

1.1 - The Password Manager Server is reliable - The client application trusts in the password manager server and may use mechanisms to assure its identity.

**Password Manager Server Assumptions**

2.1 - The system configuration is trustworthy - The proposed solution requires the system (root) to be reliable, ensuring no eavesdropping or tampering of messages will be performed. The user may use tamper-evident mechanisms to confirm the non tampering of the system.

2.2 - An authenticated smart card is reliable - The system cannot trust all the smart cards but only on those which proves to be the smart card expected by the user.

2.3 - Only processes from the same user space can communicate - The mechanisms used to communicate with client applications ensure that only processes running over the same user space can communicate with the server.

**Password Manager Applet Assumptions**

3.1 - The smart card is tamperproof - It is not possible for an attacker to access the passwords without authentication.

3.2 - No other application internal to the smart card can access the passwords - If the smart card supports multiple applications, it also implements a proper firewall between applications.

3.3 - Only the user knows the smart card access password - If someone authenticates with the user password, he is the expected user.

**3.3 Communication Security**

In the previous section, no trust assumption regarding the communication channel between the computer and the smart card has been specified. These assumptions were not made since the connection between these components may be made either using a classic ID-000 smart card through a physical reader, or using a modern smart card inside a smart phone through any wireless channel.

Due to this channel related uncertainty, the proposed solution must ensure the desired security properties of the communication. One property to ensure is the protection of the data sent between the two entities, which is performed as described in section 3.3.1. It is also important for the both parties to assure the identity of the other through authentication, which is performed as described in section 3.3.2.
3.3.1 Message Protection

During the communication between the computer and the smart card, several sensitive data are transported. To assure this data is not disclosed, the communication channel must be encrypted, guaranteeing the confidentiality of the messages. Although the password of the user could be used to cipher this channel, it may be a weak key susceptible to dictionary attacks. Another reason to perform this cipher with a different key, is the fact that the user password is a long term key which, if used to encrypt the communications, increases the number of samples for an attacker to perform crypto-analysis attacks. To mitigate these problems, a temporary key must be arranged in each session, preferably with perfect forward secrecy, ensuring such a key cannot be calculated in case the user password is discovered. This property is important since an attacker eavesdropping the communication may find the password of the user, but will still be unable to decrypt the communication. The exchange of the temporary key must be performed during the authentication process, and how the desired properties of the key are warranted is described in section 3.3.2.

In addition to the disclosure of the passwords, two types of attacks can be performed to the channel. The first is made by replaying the sent messages, where an attacker would resend to the card an old valid message to tamper with the state of the system. Such an attack can be avoided by adding a nonce to the messages. This nonce must be randomly generated by the card and returned to the computer in every response. The other possible attack is for an attacker, which is in the middle of the communication, to tamper with the messages before they are sent to the card. Like in the previous attack, this would tamper the state of the system. To ensure that the message is not compromised, a ciphered digest of the message must be added. When receiving the message, the card must calculate the digest and compare it to the received value to ensure the desired property.

The resulting format of the messages, including the mechanisms herein presented, is depicted in figure 3.2.

![Figure 3.2: Message format and protection](image)

3.3.2 Authentication

In the proposed solution, where passwords are stored inside the card, there is a need to authenticate the user, thusly assuring that no passwords will be handed to an attacker.

In some security solutions using smart cards, there may not be a need to confirm the identity of the smart card. In a scenario where a user would send a non-confidential message for the card to authenticate, if the card were to be malicious, no security property would be broken. This contrasts...
with the proposed solution in which confidential data is sent to the card, implying that besides user authentication, it requires authentication of the card to assure its authenticity.

Not trusting the smart card and requiring its authentication, brings yet another requirement to the authentication process. The key of the user can not be sent to the card, requiring the authentication scheme to use a zero-knowledge password proof mechanism, a mechanism which allows two entities to authenticate without revealing the password.

It is also necessary for the protocol to ensure the freshness of the presence of the communication parties, guaranteeing that an attacker is not repeating messages from previous authentications.

To achieve these desired properties, the proposed solution uses Encrypted Key Exchange [EKE] as the authentication protocol. This protocol provides mutual authentication with a zero knowledge mechanism, and certifies the freshness of both entities. Besides the properties related with authentication, it also generates a temporary key with perfect forward secrecy, required to ensure further secure communications as described in 3.3.1.

![Figure 3.3: Authentication Protocol](image)

Before authentication can be performed for the first time, the user must define a password. This secret must be sent by the user to the smart card, and it is important to ensure its confidentiality. To enable this confidentiality, the smart card contains an asymmetric key pair. The public key of this pair must be used to cipher the new user password before it is sent to the smart card. The private key, which never leaves the smart card, is used to decipher the new user password.

### 3.4 Functional Communication

As herein described, the proposed solution stores passwords inside a smart card, being them transported between a personal computer and a smart card with the security properties presented in 3.3.
Asides from the passwords, password managers are also interested in storing meta-data associated to them. This meta-data contains attributes such as the user name and the name of the service related with the password, that are required to search for credentials in the database and to authenticate the user. The attributes stored by the proposed solution are represented in table 3.1. A set of attributes related to a password is called a credential.

<table>
<thead>
<tr>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
</tr>
<tr>
<td>PasswordNameField</td>
</tr>
<tr>
<td>UserName</td>
</tr>
<tr>
<td>UserNameField</td>
</tr>
<tr>
<td>ServerName</td>
</tr>
<tr>
<td>OptionalField</td>
</tr>
<tr>
<td>OptionalField2</td>
</tr>
<tr>
<td>ID</td>
</tr>
</tbody>
</table>

Table 3.1: Credential Attributes

As described in Section 2.2.4, data sent in the messages exchanged between a client application and the smart card is limited to a length of 255 bytes. Considering that the nonce and digest considered in the communication protocol, specified in section 3.3.1 have a size of 32 bytes each, only 191 bytes of useful data can be transmitted to the card. Since sending multiple messages to the card to perform an operation may drop the overall performance of the system, it is important to be able to send as much credential attributes as possible inside a single message.

A usual format used in communication protocols is Type-Length-Value (TLV)\[1\]. This format enables multiple objects to be sent in the same message. As the name implies, the data sent is the type of the object, its length and its value, being the type and length of fixed size. In the proposed solution, an optimized format based on TLV is used. In this format, the information regarding the types of the objects sent in the message is reduced to one byte. Each bit of this byte represents a type of object, as specified in appendix C and a bit with the value one represents that the corresponding object is being sent. This optimization requires objects to be ordered inside the message. The TLV format and the proposed variation are depicted on figure 3.4.

Figure 3.4: TLV message format and variation

This TLV format is used to send attributes on the messages which perform operations over the
smart card. To manage the passwords, the operations defined correspond to the four basic Create, Read, Update, Delete (CRUD) operations provided by most systems to access objects. Appendix B specifies in detail the format of the APDUs that correspond to these operations. When creating a new credential, attributes are usually already prepared to be associated to this credential. Instead of requiring two operations to be performed, a create operation followed by a set operation, the create operation allows values to be inserted in the newly created credential, avoiding multiple messages from being unnecessarily sent. The Read operation allows a query to be performed based on multiple parameters and these parameters must also be on the TLV format. Although there is no operation to get all the existing credentials, each of those is stored with an identifier generated inside the card. In case a tool wants to access all the credentials, for instance to perform a backup, it can perform access to all of them by performing Read requests to all the identifiers.

### 3.5 Process Authentication

As previously specified, measures are taken to assure the authenticity of both the card and the user in the communication between the password manager server and applets. Besides this channel, it is also important to ensure the security properties regarding the communication between the client applications and the server. Assuming there is an IPC channel which assures data cannot be eavesdropped, it is important to ensure the legitimacy of the processes communicating. The server must be aware of the intents of the accessing processes to avoid the stored passwords to be improperly accessed, granting access only to applications with the user’s permission. The concern of the client applications is to warrant that the newly created passwords are not sent to a non-legitimate server. Both these concerns are related with the possibility of one malicious application running on the same user space as the user.

As described in section 2.1.3, Keyring performs access control based on the user space of the accessing process, while KWallet requires the user to accept the application based on its name. Although the second is a securely stronger approach, it still is lacking because of the fact that an attacker can launch a process masqueraded as an application trusted by the user in which it will trust.

The proposed solution is built upon the one proposed by KWallet, besides allowing users to trust applications based on their name, it also allows the user to register an application to always have access to the passwords. This solution mitigates the risk of an application performing a masquerading attack by tricking the user into trusting an application with a false name. To ensure proper configuration, the user must perform this authorization in a state in which he knows the system is secure.

To provide this application registration the server has two requirements. The first is to have a way of identifying the accessing application and calculating its interesting attributes. The second is being able to store the configuration in such a way that another application running over the same user space cannot tamper with it.

Assuming the proposed solution is using an IPC technology which provides access to the process identifier (PID) of the communicating processes and an operating system which provides information
on the process based on this identifier, from the [PID] the server is able to get information about the process to confirm its identity. One simple way to test the identity of a process would be to check the path of the application, but this method does not ensure that the application has not been replaced or tampered with, since in case of the application being installed in the user space it could be tampered with. An alternative is to calculate a digest of the executable file of the process, guaranteeing it is the one expected for that application name or path. This same method can be used by client applications to ensure the identity of the password manager server.

To guarantee the stored hash values are not tampered by an application over the same user space, these must be stored through an application running over other user space and the file must be configured not to have write permissions for other users. For such behaviour to be performed, the application hash storage component is introduced. This component runs over another user space, either dedicated or root, protecting the file. For the password manager server to be able to access such a component, they communicate over a different [IPC] channel, allowing communication between processes from different user spaces, and the application hash storage component calculates the hash of the server to assure its identity, only allowing the password manager server to change such data. Figure 3.5 depicts the workflow regarding the control of the access protocol.

![Figure 3.5: Access control components and workflow](image)

Regarding the base architectural view, depicted in figure 3.1, the application hash storage component is not depicted since it can be considered as part of the password manager server component.

### 3.6 Conclusion

In this section the proposed solution is presented. This solution consists of a password manager able to integrate with multiple applications, providing a secure protection of the passwords.

This solution is based on two main components. The first is the password manager server, an instance running over the user’s computer to which client applications delegate the password management functionalities. To provide a secure storage, the server communicates with a smart card in which the passwords are stored. The smart card runs the second main component of the proposed solution, an applet that implements the storage operations. To properly design the remaining details of
the system, a trust model is proposed. In this trust model assumptions are taken by each component regarding the underlying system and the communication channels.

To provide a secure communication between the computer and the smart card, several properties are taken into account. The first property is the identity of the user in the computer and the identity of the smart card. By using [EKE] both the user and the smart card can authenticate each other without directly exchanging the password. This authentication protocol also generates a temporary key with perfect forward secrecy assurance. This key is used to encrypt session data, providing confidentiality to the channel. To avoid other common attacks to the channel, such as message replay and message tampering, a random nonce per message and a digest of the whole message are added to the exchanged data, avoiding each of these attacks. These security mechanisms allow the proposed solution to communicate with smart cards independently of the trust in the channel, meaning that even with a smart card accessed through an unreliable wireless channel, the solution will still be secure.

Regarding functionality, the operations provided by the password manager applet correspond to the four CRUD operations. Due to the size constraints of APDUs the data contained in the messages is formatted according to an optimization of the TLV format.

Besides protection regarding the communication between the computer and the smart card, it is also important to ensure the secure communication between the components running inside the computer, guaranteeing that only applications trusted by the user can access the passwords. To provide such features the proposed solution uses mechanisms based on the IPC channel and operating system mechanisms. The password manager server identifies the client applications through the IPC channel, and asks the operating system for their executable code. Based on this executable code the password manager server is able to confirm the authenticity of the applications and perform a proper access control based on the identity of the applications.
4 Solution Implementation

Contents

4.1 Inter Process Communication Technology ................................................. 38
4.2 Smart Card Access ................................................................................. 38
4.3 Applet ..................................................................................................... 42
4.4 Password Manager Server ..................................................................... 43
4.5 Client Integration .................................................................................. 43
4.6 Conclusion ............................................................................................. 44
The proposed architecture is composed by several components which cooperate to provide the desired features and properties. The implementation, the components and their communication requires technologies matching the required properties. This chapter describes the technologies used to implement the system and the properties obtained from each. It also describes some implementation details regarding quality properties of the system.

Section 4.1 describes the IPC technology used to perform communication between the password manager server and both the client applications and the application hash storage. Section 4.2 describes how the password manager communicates with smart cards and how these smart cards can be used inside a smart phone. Details regarding the implementation of the applet are described on section 4.3. Section 4.4 describes the programming languages and frameworks in which the password manager server was implemented and the rationale for using them. Section 4.5 describes how the proposed system can be integrated with Firefox. This chapter ends with some conclusions regarding the overall implementation of the solution in section 4.6.

4.1 Inter Process Communication Technology

For the communication between client applications and the password manager, an IPC technology is required. It is necessary for the chosen technology to ensure the legitimacy of the service being accessed and provide secure point to point communication. For this purpose, the DBUS technology has been chosen.

When an application accesses DBUS it can either access the system bus or the session bus. While the system bus allows any application to connect, the session bus only grants access to applications running over the same user space as that session. DBUS also provides service name registration. When registering a name, this name cannot be re-used. By registering the password manager server on the session-bus, it guarantees that only applications running in the same user space as the user can access the service, conforming to the trust assumption 2.3 - Only processes from the same user space can communicate.

DBUS also allows communicating applications to check the identity of the other communicating party through its PID. Another relevant property provided by DBUS is the service name registration, which allows the password manager server to be registered with a predefined name that can be accessed by other applications. An important property to consider related to the integration with client applications is the fact that solutions exist to automatically generate code for different programming languages reducing the integration effort.

Since the application hash storage component runs in a different user space to ensure protection of the applications database, the communication is performed through the system bus.

4.2 Smart Card Access

Although smart cards allow several security solutions to be implemented, such as user authentication and data protection, there is still a burden related with the transportation of the peripherals
required to communicate with the smart card. Most personal computers do not provide a smart card reader, either embedded or attached, requiring the user to transport such a reader. This drops the availability of the systems, since users require such a device, and drops the usability of the system, since the user is required to transport the reader. The transportation of the card is not assumed as a problem since common smart cards are easily transportable fitting in a wallet, as is the case of debit cards.

An alternative solution to augment the availability and usability of the proposed solution is to use alternative smart card formats and communication channels, not requiring the user to transport a reader. This section proposes the usage of a smart card inside a smart phone. The card can be either in the SIM format or embedded in a microSD. By having such an alternative, an user which already possesses a smart phone may be relieved from carrying or relying on the existence of a smart card reader.

To perform the communication with the smart card the proposed solution uses PC/SC technology. This technology allows applications to abstract from the concrete smart card and readers being used. It is also specified that no secure properties are assumed relating to the communication channel, being such properties ensured by the communication model. This implies that the proposed solution is, by design, independent of the channel being used.

To adapt different smart card readers to PC/SC new reader handlers must be created and installed. Another solution is to use the Virtual Smart Card Architecture (VSCA). VSCA consists of a reader handler which communicates with several virtual smart cards through a socket. This allows virtual smart cards to be more easily deployed and integrated with the system. Several virtual smart cards compatible with VSCA exist. Some simulate real smart card solutions such as the German Identity Card. Another case, more representative of the goal proposed in this section, is PC/SC Relay which allows the communication through PC/SC with a contactless smart card. Figure 4.3 depicts a simplified architecture of VSCA and how it is integrated in PC/SC.

By using VSCA to enable the proposed solution to communicate with a card through any other
means, a virtual smart card must be implemented. To use a smart card contained inside a smart phone, two components were created. The first component is the needed virtual smart card. This component is then responsible to transfer the APDUs to the smart phone. Inside the smart phone, a component called Smart Card Proxy is deployed. This component will receive the APDUs sent from the computer and will be responsible for transferring them to the smart card. This architecture is depicted in Figure 4.4. The presented components only transfer the APDUs, without analysing or processing them. The implemented solution assumes an Android phone containing a Go-Trust® card. The communication between the computer and the smart phone is performed using Bluetooth.

One important property of the proposed virtual smart card is that it is independent of the proposed password manager, and not specific for it. Any application using PC/SC can communicate through the implemented channel, although some security concerns may have to be taken into account for each particular solution. Since the proposed solution does not rely on the security properties of the channel, applying its own mechanisms to assure a secure communication, this virtual smart card can be integrated with the proposed solution without presenting a security risk.

When requesting data from a smart card the number of bytes returned by an operation may be
unknown until this operation is performed. In this case, the smart card will store the data to be returned locally and will respond with the amount of data to be fetched. This implies that a second message must be sent and answered by the card to obtain this data. This process is depicted in figure 4.5. The behaviour of requesting the response of a previously sent message is performed by the applications, not by the middleware.

![Figure 4.5: Smart Card Get Response](image)

When the card is not physically connected to the computer, or some intermediary module performs part of the communication, the transmission of multiple messages may affect the performance of the system. In the case of the proposed virtual smart card, the performance of the system is affected, since the extra messages to obtain the data from the smart card are propagated from the computer through all the intermediary components, as illustrated in figure 4.6. This was the behaviour implemented by the first version of the proposed virtual smart card.

![Figure 4.6: Smart Card Get Response in Mobile Scenario](image)

To optimize the behaviour of the virtual smart card regarding the data fetching, instead of only forwarding APDUs as in the original version, the android application was changed to be aware of the responses returned by the smart card. In case the smart card returns the length of data to be fetched, this data will automatically be fetched by the application and returned to the computer, instead of the original response, avoiding multiple round trips to the mobile phone. This process is depicted in figure 4.7.

![Figure 4.7](image)
Besides the module created to communicate with the smart card inside a mobile phone, a virtual smart card was created to enable a local communication with the microSD smart card through its specific middleware. Although not as relevant as the previous case, this module was also optimized to avoid multiple round trip times, being the answer automatically fetched by the virtual smart card.

4.3 Applet

To implement the desired smart card behaviour, the card was programmed using the Java Card technology. This technology allows to create abstraction over the smart card implementation, allowing the proposed solution to be independent of a specific card manufacturer. Another important property provided by Java Card is the possibility of installing multiple applications inside the same card, allowing the proposed solution to be deployed alongside other applications, not requiring the user to transport a card for each case. To ensure the data’s security, Java Card implements a firewall which disables applets to access data which does not belong to them. This firewall is compliant with the security assumption 3.2.

In terms of the implementation of applets, although Java Card is a subset of Java Standard Edition, some properties of this system must be taken into account. The first property is the fact that, instead of storing objects in RAM, which is volatile and erased after unplugging the system, in Java Card the objects are stored persistently in EPROM. The problem usually related to EPROM is that write access times are much longer than the same access when using RAM. To access volatile memory, besides basic type local variables of methods, the programmers have access to Java Card specific arrays. To ensure a good performance, all the processing performed by the proposed solution implementation writes only on volatile memory. The only exception is when storing the credentials of the user which is an operation made over EPROM. To ensure operations performed over data inside the card are atomic and the system is not corrupted in case of unplugging, Java Card provides a transactional system which guarantees that either all or none operations will have effect. When storing data inside the card, transactions are used to ensure a proper state of the system.
Another relevant property to take into account, is that EPROM space is usually shared by the executable code of applications and their data. This implies that the size of the code of the application affects the amount of data that can be stored. Most of the processing of the applet is very similar. Examples of this are security operations, such as deciphering data and checking the nonce, which must be performed in all the CRUD operations. Such code is implemented only once in a centralized place before performing each operation's particular code. Another case taken into account is the attribute creation and setting operations which are almost the identical, differing only by creating a new object or searching for an existent one.

4.4 Password Manager Server

The programming language and environment to develop the password manager server was chosen based on two main requirements. The first is the compatibility with both D-BUS and PC/SC, the technologies used to perform communication with other components. The second requirement is being able to create a Graphical User Interface (GUI) for the user to perform the card choosing and authentication operations.

The technology chosen to implement the server was Qt, a C++ cross-platform application framework which provides GUI creation through widgets. Qt comes bundled with the QDBUS library, that can be used to create or access DBUS services. QDBUS allows programmers to export the interface of an object as a DBUS interface. It also automatically implements the Introspect method, which allows client applications to read an XML specification of the interface, and can be used to automatically create client stubs. Since the server is implemented in C++, the access to PC/SC can be performed through its C library.

To perform cryptographic functions related with the secure communication, the openSSL crypto library is used, being compatible with C++.

To access information regarding processes, used to perform the mechanisms described in 3.5, the system relies on procfs.

Except for the PC/SC libraries, all these technologies were also used to implement the application hash storage component. In this, component the hashes of the registered applications are stored in a sqlite database.

The created GUI consists of two windows: A window used initially by the user to choose the smart card and perform authentication; and a window which is used for the user to grant or deny access to client applications. These interfaces can be consulted in appendix D.

4.5 Client Integration

In order to be tested, the proposed solution was integrated into the Mozilla browser, Firefox. This application allows the installation of Cross Platform Component Object Model (XPCOM) modules, modules that can be implemented in a range of languages from interpreted languages, such as
javascript, to native languages, such as C++. These modules must implement an interface as described in a Cross Platform Interface Description Language (XPIDL) file. This file provides a high level implementation independent description of the interface, making it possible for different modules to access each other independently of the programming language. Besides enabling the installation of new XPOM modules, most functionalities of Firefox are also implemented through such modules, implying that several XPIDL interfaces come by default and are implemented by it. Firefox enables the installation of modules which implement such interfaces, overwriting their default implementation.

The proposed module implements the nsILoginManagerStorage, a default interface which is used by Firefox to store the passwords. By implementing this interface, the module is able to abstract from the form filling and user interaction functionalities, already implemented by Firefox, and focus on the storage of the passwords.

Due to the need of integration with DBUS, the proposed solution XPOM module was implemented using the C++ language. To access the password manager server this module uses QDBus functions automatically generated through qdbusxml2cpp, a program which transforms Extensible Markup Language (XML) declared DBUS interfaces to client or server C++ code.

4.6 Conclusion

This section described the used technologies and implementation details which were adopted to implement the proposed solution. With these technologies several desired properties were achieved.

By using DBUS to perform communication between computer components, applications can be easily integrated with the proposed system through automatically generated proxies. DBUS also provides session based channels and introspection mechanisms, which are required to provide access control.

Regarding the communication with the smart card and the implementation of the applet, the chosen technologies allow the proposed solution to be independent from particular manufacturers. This is achieved in the communication by using PC/SC which allows multiple readers to be used with the same interface. The applet was implemented using the Java Card technology, allowing the application to be deployed on cards from multiple manufacturers. This technology also allows multiple applications to be installed on the same card, implying that the user is not required to transport a card specific to the proposed solution.

To enable users to use smart card systems without requiring a smart card reader, a module is proposed which enables the user to access a smart card contained in a smart phone. Since the proposed solution is prepared to be secure over insecure communication channels, the proposed module can be integrated with the proposed password manager without representing a security risk.

Besides the implementation of the proposed solution, a Firefox extension was implemented. This extension replaces the default password manager which exists in Firefox and, being this extension integrated with the proposed solution, it enables users to store and access their browser passwords in the smart card with all the required security properties.
# Solution Evaluation

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Solution Comparison</td>
<td>46</td>
</tr>
<tr>
<td>5.2 Solution Benchmark</td>
<td>47</td>
</tr>
<tr>
<td>5.3 Usability Tests</td>
<td>50</td>
</tr>
<tr>
<td>5.4 Requirements Fulfilling</td>
<td>51</td>
</tr>
<tr>
<td>5.5 Conclusion</td>
<td>52</td>
</tr>
</tbody>
</table>
This chapter takes into account the state of the art and both the design and implementation details to assess the proposed password manager and its provided properties. Section [5.1] analyses some of the attributes of the proposed solution comparing it with other solutions existing in the market. Section [5.2] introduces results regarding benchmarks performed over the solution to analyse its performance. Section [5.3] describes the usability tests performed to evaluate the solution from the perspective of the users. Section [5.4] describes how the requirements defined in [1.3] are fulfilled by the proposed solution. Section [5.5] summarizes the properties provided by the solution and how these fit in the goals of this thesis.

### 5.1 Solution Comparison

Due to its nature, it is relevant to assess the proposed system by comparing it with other existing systems. The most significant properties to take into account are those studied for other solutions in [2.1].

Table [5.1] complements the previously presented table [2.1] adding a row representing the proposed solution allowing the comparison of the proposed solution with the existing solution categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Discovery of one password</th>
<th>Availability</th>
<th>Portability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSO</td>
<td>Access to All Systems</td>
<td>Requires connection to authentication service</td>
<td>Server side integration</td>
</tr>
<tr>
<td>Hash Based</td>
<td>Allows brute force attack to master password</td>
<td>Only dependent on the algorithm</td>
<td>Portable</td>
</tr>
<tr>
<td>Password Managers without access to Database</td>
<td>No attacks</td>
<td>Requires access to database</td>
<td>Client Side integration</td>
</tr>
<tr>
<td>Password Managers with access to ciphered Database</td>
<td>Allows brute force attack to master password</td>
<td>Requires access to database</td>
<td>Client Side Integration</td>
</tr>
<tr>
<td>Proposed Solution</td>
<td>No attacks</td>
<td>Requires access to card</td>
<td>Client Side Integration</td>
</tr>
</tbody>
</table>

**Table 5.1**: Comparison of Single-Password solutions including proposed solution

Being a password manager, the proposed solution presents the same properties as the other solutions in its category, with the advantage of never having access to the database given its protection inside the smart card. This property avoids brute force and dictionary attacks performed to other password managers when the file is discovered. The proposed solution also provides an availability only achieved in most password managers through file synchronization, although it still does not provide as much availability as Hash Based password generators.

Table [5.2] complements table [2.2] with a row representing the proposed solution, allowing its comparison with some of the presented password managers.
<table>
<thead>
<tr>
<th>Password Manager</th>
<th>Key Type</th>
<th>Brute Force Attacks</th>
<th>Availability</th>
<th>Applications Control</th>
<th>Access Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Chrome</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>Based on user space</td>
<td></td>
</tr>
<tr>
<td>KWallet</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>Based on user permission</td>
<td></td>
</tr>
<tr>
<td>Keyring</td>
<td>Password</td>
<td>Offline</td>
<td>File Synchronization</td>
<td>Based on user space</td>
<td></td>
</tr>
<tr>
<td>Kamouflage</td>
<td>Password</td>
<td>Offline but requires testing the password</td>
<td>File Synchronization</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Web Solutions</td>
<td>Password</td>
<td>Offline in case of system attack</td>
<td>Online</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>IDKeeper</td>
<td>Pin</td>
<td>Online</td>
<td>Card Transportation</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Tapas</td>
<td>Strong Key</td>
<td>Offline</td>
<td>Key Synchronization</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Proposed Solution</td>
<td>Weak Password or Pin</td>
<td>Online</td>
<td>Card/ Phone Transportation</td>
<td>Based on user permission</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Comparison of password managers including proposed solution

The proposed solution provides password protection due to the capacities of the smart card. Although an attacker may try brute force attacks, and the user may choose a weak password, the proposed solution requires an online attack to be performed, meaning that the attacker will need to communicate directly with the card. However, such an attack has a low probability of succeeding due to the blocking of the card after a fixed number of password tries. Because of this, the user’s choice of password may be weak, but it will not consist of such a risk to the system’s security. Regarding access control, the proposed solution is based on user permission, which is stronger than based on user space since it avoids attacks from applications which may be unconsciously launched by the user. By using the presented mechanisms, it may also avoid masquerading attacks to common client applications.

5.2 Solution Benchmark

Since a reduced performance may affect the usability and utility of a system it is necessary to analyse how the proposed solution performs its most commonly used operations and how said performance scales depending on the amount of transmitted data. It is also relevant to analyse this performance when communicating with a smart card, either directly or inside a mobile phone.

To provide this data, two benchmarks were created. Both benchmarks were performed over an 8x Intel® Core™ i7-2600 CPU running at 3.40GHz, with 4 Giga bytes of addressable memory, and an Ubuntu 12.04.3 LTS as the operating system. Due to various factors, such as operating system scheduling and system load, the same operation’s time span may vary. The data herein presented is an average of at least six taken measures.

The first benchmark consists of a stand alone application performing direct calls to the password manager server. These calls consist of creating new credentials with some data and reading data
from credentials. In both cases, measures were taken with the smart card being accessed through four different channels. These channels consist of the VSCA modules to connect the smart card either inside a smart phone or directly connected to the computer. In both cases the simple and optimized versions, which avoid round trips as described in [4.2] are presented, allowing us to evaluate the success of the optimizations. The access time measures were taken regarding accesses with a size of 0, 1, 50, 100, and 200 bytes of data. These sizes do not take into account the header and security values presented on the message, which is of 46 bytes.

![Figure 5.1: Direct Call Benchmark - Credential Creation Operation Performance](image1)

![Figure 5.2: Direct Call Benchmark - Credential Read Operation Performance](image2)

Figure 5.1 represents the obtained times when writing new credentials and Figure 5.2 represents the obtained values when reading credentials. The first point to consider is the difference between the average time taken to perform credential creation and the average time taken to perform access operations. These have an average difference of 100 ms. The main factor for such difference is the fact that in the case of the create operation, the created object and the values received by the card must be written in the EPROM as opposed to the other case in which no data was being written.

Regarding the optimizations performed on the virtual smart cards, as described in section 4.2, in the case of the operations performed over the mobile solution, the optimization performs an average
of 60ms faster than the original version, since the smart phone fetches and answers data automatically, avoiding the send of two more messages. In the case of the local optimized version, it has an average latency drop of 20ms. These values represent the time that was being taken with round trips to fetch data from the smart card, and demonstrate that, in fact, the optimization improves the performance of the system, especially when the smart card is inside the smart phone, which involves more components and different devices.

The second benchmark consists in a firefox component which performs calls to the nsILoginManager interface, allowing a comparison between the proposed firefox extension and the default firefox password manager. This benchmark was only performed using the optimized versions of the virtual smart cards, avoiding multiple round trips to perform an operation. It allows an overview of the behaviour of the proposed password manager inside a real application. The tests evaluate the behaviour of the application when sending 1, 50, and 100 bytes of data whilst creating and accessing credentials. In contrast to the previous benchmark, one operation over nsILoginManager may correspond to more than one operation over the password manager server, which explains this benchmark greater results.

As depicted in figures 5.3 and 5.4, in both operations the proposed solution performance has a
significant increase of latency. Such a difference was expected, since firefox operations are performed locally on the computer, in contrast with the proposed solution which propagates data to the smart card. In the case of the get operations, the firefox solution is almost instantaneous, given the fact that the whole database is loaded and cached in memory when firefox is launched. This contrasts with the proposed solution, where no caching is being performed. The create operations on firefox are not as immediate, but also perform fast, requiring less than 250 Milliseconds (ms), as depicted in 5.3. The time performance differences between the get operation and the remaining ones is explained by the fact that, during the execution of the latter ones, write operations are being performed.

5.3 Usability Tests

To assure the usability and viability of the system, usability tests were performed. Given the performance results, described in section 5.2, these tests are also important to assure that the latency added by the proposed solution is not perceptible by an human and does not affect the user experience.

The usability tests were performed with Firefox as a client application, and the smart card either physically attached or contained inside a smart phone. Eighteen people participated on these tests.

The tests consisted in four tasks:

1 - Authorize access to the card from Firefox and add a new credential. Confirm the credential was stored.

2 - Decline access to the card from Firefox and confirm the credential is not being accessed.

3 - Grant permanent access to the smart card from Firefox. Confirm the credentials are being accessed.

4 - Confirm Firefox has access to the password without requiring further access permission.

After performing these tasks, the users answered a survey regarding the solution. The survey is presented in appendix E, and a summary of the results is presented in appendix F.
Regarding the usability of the system, most users considered the system easy to use. All the users considered the system as being fast, concluding that the performance loss described in section 5.2 does not affect the user experience. These results are presented in figure 5.5.

Regarding the viability of the system, most users do not consider the transportation of a smart card as a problem, although the transportation of a reader is not well accepted. Thirteen of the eighteen participant users answered that they would use the proposed solution. This result is depicted in 5.6.

![Figure 5.6: System Acceptability](image)

From the comments and suggestions performed to the system, the most relevant to consider are the worry of the users from losing the passwords in case of loss or corruption of the card, and the fact that the proposed solution raised their interest on its integration with other systems, such as Google Chrome.

### 5.4 Requirements Fulfilling

In section 1.3 the requirements of the proposed solution are defined. During this documents both the design and implementation details of the solution have been introduced. This section presents how the presented properties and behaviour of the system match the defined requirements.

1. **The user must be able to authenticate himself towards multiple systems through the presentation of a single password**

   The proposed solution implements a Password Manager, mechanism described in Section 2.1.3. By design, these solutions allow the authentication with several systems through the presentation of one password. This requirement can be proven by the usage of the system.

2. **The user must be able to authenticate himself towards multiple systems without having to share his memorized password**

   By design, single password solutions allow users to authenticate with several systems with a single memorized password, that they do not have to share with any of the said systems. Being the proposed solution a password manager, the password memorized by the user is used to authenticate with the smart card and is never sent to other systems.
3 The system performance must not affect the usability of the system

As shown in section 5.2 the solution presents some performance loss. However, from the results of the usability tests described in section 5.3 this performance loss does not affect the user experience.

4 The user must be able to use the system independently of his location

By design the proposed solution is based on a Smart Card, such as those presented in section 2.2. These smart cards have the property of being easily transported. The proposed solution is thusly easily transported and can be utilized by the user in any location without requiring transportation of its data into any insecure devices or synchronization through other systems.

5 The system must be easy to integrate with other systems

By using DBUS as an IPC mechanism to communicate with client applications, as described in 4.1 the proposed solution can be easily integrated with client side applications without significant effort. Since the proposed solution, as a password manager, is only integrated with the client side of such applications, it is also independent of the server side of the systems with which it is integrated.

6 The user interaction must be as simple as possible by automatizing the authentication process

Through the automatic storage and retrieval of the credentials of the user by client applications, the user’s interaction with the systems at hand during the authentication process is reduced to the bare minimum. Because of the access control based on the identity of the processes that is provided by the proposed solution, the user is also relieved from the authorization process when dealing with applications in which he trusts. From the results of the usability tests, described in section 5.3 the system is considered by the users as easy to use.

7 The system must assure user presence to perform its functionalities

The authentication mechanisms specified in 3.3 and the protection of the data inside a smart card, allow the system to be sure of the user’s presence, avoiding brute force and dictionary attacks which can be used against other password management solutions.

5.5 Conclusion

The evaluation of the proposed solution concluded that the goals proposed by this thesis were achieved.

The proposed system and its implementation provide a single password authentication mechanism solution, more specifically a password manager, which provides strong security properties when compared to other state of the art solutions. Being a password manager, the discovery of one of the stored password through an attack to one of the services of the user does not affect the security of the
stored passwords. In comparison with other existing password managers, the proposed solution protects passwords from offline brute force attacks through the protection of the data inside a password protected smart card. The risk of successful online brute force attacks to the card is reduced through the limitation of the number of times an attacker can test passwords against the smart card. The proposed solution also implements mechanisms which allow users to control which applications can access the passwords. The transportation of the smart card assures its availability, without requiring the synchronization of the passwords in multiple devices through other mechanisms.

The presented performance results compared the proposed solution to Firefox, and demonstrated a considerable difference on the times required to access or store data. The usability tests confirm that this difference is not noticeable by the users. The usability tests also confirm that the system is easy to use and that the proposed system was well accepted.

Based on its design, implementation, and performed tests, it is concluded that the proposed solution fulfills all the requirements and goals of this thesis, providing a fully functional password manager that securely stores the passwords of the user inside a smart card.
Conclusions and Future Work
Most authentication mechanisms are based on secrets that the user shares with the system. There is a constant increase on the number of services that users require and, consequently, a need for them to share multiple passwords with those services. Since memorization of one secret per service would require an effort which users do not wish to handle, they tend to share the same password with different services. This generates a security risk, since if the password for one of the services is disclosed, an attack to all of the user’s resources would be enabled.

Three categories of solutions exist that allows users to authenticate with multiple services using only one password, but without having to share this secret with each and all of them. These categories are: Single Sign On (SSO) systems, centralized components to which a service may delegate authentication responsibilities, whose main problem is the fact that its centralization implies a single point of failure that would cause the rupture of the entire system; Hash-based password generators, systems which based on a master password generate different passwords for different services, whose main obstacle is the fact that an attacker with enough computing resources might be able to calculate the master password using one of its generated passwords; and Password Managers, applications that store the user’s passwords and, which are usually poor solutions on account of the data being stored through ciphering with weak keys. Other of Password Managers’ problems, in the case of solutions which behave as a server for other applications, is the weak access control policies performed.

The presented solution is a password manager whose main focus is to protect the user’s data with a password, but without the risk of successful offline dictionary and brute force attacks which exist in other solutions given the weak keys used to cipher the data. As such, the passwords are stored inside a smart card, a device which provides tamper-proof storage and secure computation capabilities. To access his data, the user has to be authenticated by the smart card. Online dictionary and brute force attacks are avoided by blocking the smart card after a maximum number of authentication tries. Besides authenticating the user, the smart card is also authenticated, assuring the user does not send his data to a non-legitimate smart card. To ensure the password of the user is not disclosed through the communication channel, the authentication is performed using zero knowledge proof through encrypted key exchange. All the communication performed with the smart card uses security mechanisms assuring the confidentiality, authentication and freshness of the messages. These properties allow the proposed solution to securely communicate with a smart card independently of the physical communication channel. The proposed solution, which can be integrated with multiple applications, performs an access control based on permissions of the user, only providing access from applications registered by the user. This access control policy avoids masquerading attacks, which mimic the identity of common applications.

The solution was implemented based on multiple technologies: DBUS as an Inter Process Communication Channel, QT to enable user interface and Java Card to program the smart card. To test the viability of the proposed solution, a Firefox component was developed, enabling the replacement of its default password storage mechanism with the proposed solution.

In order to perform the communication between the computer and the smart card, PC/SC technology was used. This technology allows the developed solution to be independent from the used
physical communication channel. To keep the users from having to transport a smart card reader, a module was created to enhance PC/SC, enabling smart card aware applications to communicate with a smart card which is inside a smart phone. Since the proposed solution does not rely on the security properties of the channel, applying its own mechanisms to ensure a secure communication, this module can be integrated with the proposed solution without presenting a security risk.

Future Work

The performed work presented completeness allowing it to be used in most scenarios. Nonetheless, there is still some work that can be produced to either augment its functionalities or to enhance its security in the case of trust models weaker than the considered one.

Access from Smart Phone

The work done only took into account an access to passwords from a personal computer. There may be some interest in accessing them from a smart phone in its browser application. The design of the proposed solution took this possible use case into account and the applet is independent from the presented computer components, meaning that any application may communicate directly with the applet through the interface exposed in Annex B.

It may also be of interest to communicate with the applet from different components at the same time, for instance the server on the computer and an application on the mobile phone. This communication may lead to a desynchronization of the nonce used to assure message freshness, described in 3.3.1. To solve this problem, each component may use a different logical channel. By having different logical channels, the applet is able to have different nonces for each channel and to always keep them synchronized. All the functional APDU messages of the applet have the P1 field empty as a safeguard for this use case in the future.

Dual Possession with secure storage

The proposed solution still requires user interaction to perform authentication when the smart card is connected to the computer. An alternative solution, explored by Tapas as presented in section 2.1.3 is the automatic access to passwords through the possession of two devices. In this solution the user performs authentication with one device through the presence of the other. This concept could be introduced in the proposed solution, being the smart card and the personal computer the two possessed devices. Instead of storing the key which deciphers the data in the computer, the computer would only store a key that would enable access to the smart card. In such scenario, the passwords could be protected inside a smart card, possibly inside a smart phone, disabling offline brute force attacks. Such a solution would also be able to register multiple computers on the same smart card, enabling its usage in multiple computers, as opposed to Tapas.
Password Manager API
The following code is the representation of the DBUS interface for the proposed password manager server and may be used to automatically generate stubs to integrate with applications.

```xml
<inteface name="pt.utl.ist.ist164825.passwordManagerServer">
    <method name="registerApplication">
      <arg type="s" direction="out"/>
    </method>
    <method name="getCredentialsForAttribute">
      <arg type="as" direction="out"/>
      <arg name="attributesMask" type="y" direction="in"/>
      <arg name="attributes" type="as" direction="in"/>
      <arg name="cookie" type="s" direction="in"/>
    </method>
    <method name="getCredentialsAttributes">
      <arg type="as" direction="out"/>
      <arg name="attributesID" type="i" direction="in"/>
      <arg name="attributesMask" type="y" direction="in"/>
      <arg name="cookie" type="s" direction="in"/>
    </method>
    <method name="setCredentialsAttributes">
      <arg type="s" direction="out"/>
      <arg name="attributesID" type="y" direction="in"/>
      <arg name="attributesMask" type="y" direction="in"/>
      <arg name="attributes" type="as" direction="in"/>
      <arg name="cookie" type="s" direction="in"/>
    </method>
    <method name="createCredentials">
      <arg type="s" direction="out"/>
      <arg name="attributesMask" type="y" direction="in"/>
      <arg name="attributes" type="as" direction="in"/>
      <arg name="cookie" type="s" direction="in"/>
    </method>
</interface>
</node>
```
Password Manager APDU specification
**Figure B.1: Security Related APDUs**

Get Public Key

APDU Header

Set Password

User Password Ciphered With Public Key

APDU Header

Get Session Key

Temporary Public Key ciphered with user

APDU Header

Answer Challenge

Received Nounce followed by new Nounce both ciphered with session key

APDU Header
Create Credentials

<table>
<thead>
<tr>
<th>0x80</th>
<th>0x05</th>
<th>--</th>
<th>--</th>
<th>Lc</th>
<th>Nonce</th>
<th>Attributes Type</th>
<th>LV Data</th>
<th>Digest</th>
</tr>
</thead>
</table>

Figure B.2: Functional APDUs
Credentials Type Mask
<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Binary</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>10000000</td>
<td>Password</td>
</tr>
<tr>
<td>0x40</td>
<td>01000000</td>
<td>PasswordNameField</td>
</tr>
<tr>
<td>0x20</td>
<td>00100000</td>
<td>UserName</td>
</tr>
<tr>
<td>0x10</td>
<td>00010000</td>
<td>UserNameField</td>
</tr>
<tr>
<td>0x08</td>
<td>00001000</td>
<td>ServerName</td>
</tr>
<tr>
<td>0x04</td>
<td>00000100</td>
<td>OptionalField</td>
</tr>
<tr>
<td>0x02</td>
<td>00000010</td>
<td>OptionalField2</td>
</tr>
<tr>
<td>0x01</td>
<td>00000001</td>
<td>ID</td>
</tr>
</tbody>
</table>

Table C.1: Credential Attributes Masks

Both the password manager server and applet receive as parameters the type of the attributes which are being communicated. The types of the attributes must presented through a byte in which each bit corresponds to a specific type as specified in the following table.
Graphical User Interface
**Figure D.1:** Authentication Screen

**Figure D.2:** Application Access Permission Screen
Usability Survey
Secure Password Management Using Smart Cards

The goal of this survey is to perform an evaluation regarding the usability of the solution created on the scope of the master thesis "Secure Password Management Using Smart Cards". Besides the usability evaluation, the performed questions also allow to evaluate the acceptance of the solution.

* Required

1. **Gender**
   *Mark only one oval.*
   - Male
   - Female

2. **System Usability**
   Evaluate the system regarding the presented attributes. (1 - Poor, 5 - Excellent)
   *Mark only one oval per row.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of Use</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Velocity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utility</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. **Would you use this system?**
   *Mark only one oval.*
   - Yes
   - No

4. **Security**
   Evaluate the following properties regarding their importance on the security provided by the system. (1 - Does Not Improve Security, 5 - Greatly Improves Security)
   *Mark only one oval per row.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storing Multiple Passwords</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storing the passwords in a smart card</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requiring password authentication to access the system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controlling the Applications Access</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. **Passwords Memorization**
   Using multiple services implies sharing a password with each of those. Which practices do you use?
   *Check all that apply.*
   - [ ] Reusing the same password in multiple systems
   - [ ] Memorization of a different password for each system
   - [ ] Password manager and password reusage
   - [ ] Password manage storing a different password for each system

6. **Smart Card Transportation**
   To use the proposed solution, the user needs to transport a smart card and eventually its reader. Evaluate from 1 to 5 your tolerance to the transport of the following devices. (1 - Do not tolerate, 5 - Device transportation does not affect)
   *Mark only one oval per row.*
   
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smart Card</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smart Card and Corresponding Reader</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smart Card inside a Smart Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. **With which frequency do you use the Portuguese citizen card electronic functionalities?**
   *Mark only one oval.*
   - [ ] Never
   - [ ] Rarely
   - [ ] Sometimes
   - [ ] Frequently
   - [ ] Very Frequently

8. **Comments / Suggestions**

   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
Usability Survey Responses
Summary

Gender

Male (Masculino) 13 72%
Female (Feminino) 5 28%

Ease of Use [System Usability]

1 0 0%
2 1 6%
3 0 0%
4 7 39%
5 10 56%

Velocity [System Usability]

1 0 0%
2 0 0%
3 0 0%
4 5 28%
5 13 72%

Utility [System Usability]
Security [System Usability]

Would you use this system?

Yes (Sim) 13 76%
No (Não) 4 24%

Storing Multiple Passwords [Security]
Storing the passwords in a smart card [Security]

Requiring password authentication to access the system [Security]

Controlling the Applications Access [Security]
Passwords Memorization

- Reusing the same password in multiple systems: 9, 33%
- Memorization of a different password for each system: 12, 44%
- Password manager and password usage: 3, 11%
- Password manager storing a different password for each system: 3, 11%

Smart Card [Smart Card Transportation]

- 1: 0, 0%
- 2: 0, 0%
- 3: 2, 11%
- 4: 2, 11%
- 5: 14, 78%

Smart Card and Corresponding Reader [Smart Card Transportation]

- 1: 2, 11%
- 2: 3, 17%
- 3: 7, 39%
- 4: 2, 11%
- 5: 4, 22%

Smart Card inside a Smart Phone [Smart Card Transportation]
With which frequency do you use the Portuguese citizen card electronic functionalities?

- Never: 10 (56%)
- Rarely: 7 (39%)
- Some times: 1 (6%)
- Frequently: 0 (0%)
- Very Frequently: 0 (0%)