Secure Password Management With Smart Cards

Nuno Pinheiro

Instituto Superior Técnico
nunopinheiro@ist.utl.pt

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: Single Password Authentication, Password Managers, Smart Cards

1 Introduction

In the last years, a rising number of services, both for single machines and distributed systems, have been introduced. These services handle different resources from private data to public information, requiring a reliable and secure access control system. 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[1] namely 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 behaviour attributes, such as his fingerprints. From these factors, the most commonly used is the first[2], something the user knows, or password-based authentication.

The access to different services requires the user to memorize different passwords to authenticate himself towards these services. A common approach to avoid this memorization effort is the re-usage of the same secret among different services[3], the user shares the same password with the different services. This means that 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 resources[4].

To avoid these password re-usage problems, different approaches have been considered, allowing the user to have a multi-password security strength by memorizing only a single password. The existing approaches are based on: Single Sign-on services[5], consisting of a dedicated service to user authentication, which can be susceptible to denial of service attacks, preventing the user from accessing his resources; Hash-based passwords[6] solutions that deterministically generate passwords from a master password, which main problem is the relationship between the master password and the generated passwords, making them susceptible to brute force attacks; and password managers systems[7] that store passwords of the user encrypted with a master password, with the main problem of storing the data on devices which are not secure.

The work herein presented builds upon these single-password authentication solutions to design a password manager based on smart cards, which allows users to authenticate towards different services using different strong passwords, but requiring the memorization of only one, possibly weak, password.

Section 2 covers the state of the art, covering the current solutions to single-password authentication and introducing smart cards and their role on the authentication process. Section 3 describes

the proposed solution for a single-password authentication mechanism, more specifically, a password manager based on smart cards. To evaluate the proposed solution, Section 4 describes a deployment of the system and its obtained properties. Section 5 concludes this paper with some final remarks and future work directions.

2 State of the art

To avoid the memorization effort and mitigate the effects of password disclosure, Single Password authentication mechanisms have been proposed. These mechanism allow a user to authenticate himself towards different systems requiring only the memorization of one single password.

One type of single password mechanism are password managers. A password manager is an application that stores and provides access to passwords. These passwords are typically stored on the machine of the user.

Section 2.1 describes different types of single password authentication mechanisms solutions and compares the different properties provided by each one. Given that the proposed solution considers the use of smart cards as secure storage devices, section 2.2 describes this technology. Section 2.3 describes mechanisms that can be used to control access to resources.

2.1 Single Password Authentication Mechanisms

This section covers single sign on, hash based passwords and password manager solutions that allow users to have a multi-password security-strength with the memorization of a single password.

Single Sign On Most services implement their own authentication system. A different approach to this is to delegate the user authentication to an external service. This kind of services, known as Single Sign On(SSO), consist of a service with the single purpose of authenticating the user and securely communicate a successful authentication to the services the user wants to access. This allows the user to memorize only one password to authenticate with the SSO service, providing access to multiple services.

To allow the usage of a SSO system, the service to which the user wants to authenticate must be aware of the SSO system and must trust in it. This implies that if a service does not trust in the SSO system of the user, the user will still need to memorize a new password for the service. This implies that SSO systems have low portability, only working with systems which trust them. One reason that can lead to not trusting in SSO systems is the 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[8], but none has been widely accepted, meaning that a user still needs to access multiple SSO systems 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.

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. 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, the user then repeats the generation process, obtaining the password for that given service. Some services can use more parameters than those previously specified, such as the user name, adding more entropy to the generated passwords. In most cases, 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: It must be computationally hard to calculate the master password from the generated passwords and the algorithm

must be deterministic. The first property grants that in case of disclosure of one of the passwords it is not computationally feasible to discover the other passwords. 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 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 all his services.

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 not integrated standalone password generator, generate the passwords from this standalone application, and manually insert the passwords on these systems. This also grants availability of the system since the user can run this standalone application in a smart phone or web-based application.

Password Managers Password Managers are applications which store and provide access to the passwords of the user. The user only needs to memorize a master password and store his passwords in the password manager.

Two main types of password managers exist: Password managers internal to applications, such as a component of a browser with this functionality[9], or Password managers external to the applications, which work as a server for other applications to retrieve and store the passwords used by them[10]. By choosing different passwords for different services, the user can achieve a truly multi-password strength, requiring him to remember only one master password. These passwords are also independent since they are not generated from a root value.

To protect the stored passwords, symmetrical cipher algorithms are used. These algorithms use the user password, or a transformation of it such as its hash, as a key to encrypt the stored data. This means that only with the presentation of the user password, will these systems be able to get the passwords.

As in the case of hash-based password, this means that if an attacker performs a brute force or dictionary attack to the data stored in the password database, he will be able to retrieve the master password and access the data. An advantage of passwords managers is that the attacker will first need to access the passwords database, while in hash-based passwords, an attacker only needs one of the generated passwords.

Some solutions do not implement one or more of these features. By default, Mozilla Firefox does not require a master password, not requiring the user to authenticate, and the cipher of the passwords data base is performed with a default key. Google Chrome delegates the ciphering and authentication to the operating system. In the case of running over the Windows operating system, the encryption is made by the CryptProtectData provided by the Windows API, assuring that the data can only be accessed after the user is properly authenticated in the operating system.

An important feature which must be considered on password managers external to the applications is the access control policy that defines which applications can access the user passwords. Two examples of this type of systems are Gnome Keyring[11] and KDE KWallet[12]. Gnome Keyring access control policy is based on the user space in which the applications run. After the user authenticates, any process running over the same user space as the user can access the data. KDE KWallet policy requires the user to accept or deny this access from an application when it tries to access the passwords for the first time in a session.

In spite of this, in case of computer loss or non legitimate access, an attacker will be able to access the passwords' database since this database is stored on an insecure device, and although cryptographically protected against trivial attacks, it will not resist to dictionary or brute force attacks.

Some solutions try to mitigate brute force attacks by storing the passwords in a different device. IDKeeper[13] proposes a password manager which passwords are protected inside a smart card and can only be accessed after user authentication with a pin. The risk of a successful brute force attack is mitigated since such an attack has to be online instead of offline, limiting the speed of the attack to the speed of the card. Tapas[14] proposes a password manager based on dual possession. In this solution the password manager is composed by two components, one in a personal computer and the other in a

smart phone. The user is not required to memorize a password since he performs authentication with each device through the possession of the other. The database containing the passwords is stored in the smart phone, and the key to decipher the data is stored on the computer. In case of non legitimate access to one of the devices, an attacker will only have access either to the ciphered database or to the keys. The security of the database is assumed since it is protected by a key stronger than a user memorized password.

2.2 Smart Cards

In solutions like password managers, where data is encrypted with possibly weak passwords, there is some probability that an attacker can access the data and decipher it. Smart cards[15] are a solution that is able to mitigate this problem. They consist of a card, usually with the size and shape of a credit card, which can store and process data.

Smart cards are designed to be tamper-resistant, meaning that it is not possible to successfully get data stored inside the card by using physical attacks. Their design also covers hardware-specific processors which implement cryptographic functions in such a way that side-channel attacks can be avoided.

These properties allow smart cards to be used on authentication protocols based on something the user owns. Some smart card solutions also allow authentication of the user by fingerprint matching inside the card. In these scenarios, the smart card secure storage allows for the authentication key and user fingerprint template to be securely stored inside the card.

An usual technology to perform communication between a computer application and a smart card is PC/SC[16]. This technology consists of a multi-platform infrastructure providing a common interface to access smart cards with readers from different manufacturers. To provide smart card and reader abstraction PC/SC architecture consists of a resource manager and several reader handlers, as depicted in figure 1. Client applications communicate with the resource manager which forwards messages to particular reader handlers. Each handler is then responsible to implement the correct behaviour to communicate with its particular physical reader.

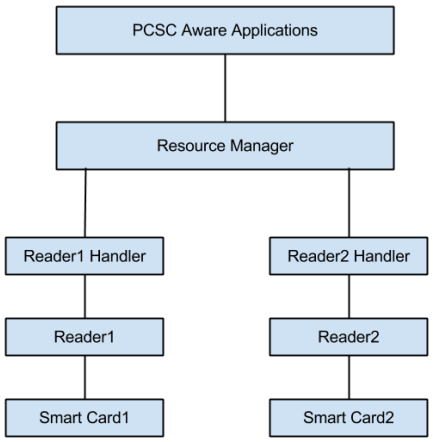

Fig. 1. PC/SC simplified Architecture

Although smart cards are used in several solutions as trusted components, PC/SC does not grant security properties, such as authentication and confidentiality, to the messages. Designers of systems based on smart cards are responsible for assuring these properties in case of card replacement or channel eavesdropping.

2.3 Access Control Mechanisms

Section 2.1 presented two password managers which are external to applications. This type of solution requires an access control policy to define which applications can access the stored passwords. These solutions use operating system and Inter Process Communication (IPC) mechanisms to implement such policies.

In classic operating systems access control is performed based on the user. When a user launches a process it will have all the permissions associated with the user, and all the processes spawned by this will be granted these same permissions. Even if a system is protected by authentication, there may still be threats after the user authenticates. In a scenario where a user would be required to insert a password to allow him to access a ciphered disk, in case of malware running on the user 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.

A different approach that can be used to provide a better access control is to grant access to resources only by 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[17].

When two applications are communicating through an IPC channel, one of those may want to assure some properties of the other. To be able to assure those properties there are two requirements. First, the IPC technology must provide mechanisms to identify the processes. Second, the underlying operating system must provide services which based on such identifiers provide information regarding the properties to be verified. DBUS technology[18], used by Keyring and KWallet to communicate with client applications, provides mechanisms to know the process identifier (PID) of the communicating process. With the PID, an application is able to access information about the corresponding process from procfs[19], a file system which directory structure represents information regarding processes. Examples of information that can be read from this file system are the command line arguments used to launch the process and its executable.

3 Proposed Password Manager Solution

As discussed above, being password-based authentication the most common method, 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 without having a relation between different passwords, as opposed to hash-based passwords, and discovering this single password will not provide access to all the systems unless the password database can be accessed. A problem that keeps existing with most password managers is that, when the encrypted password database can be 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 (SC) to protect data. SC can be used to cipher the data to be stored outside the card, protecting the data with keys cryptographically stronger than a common password, and can even be used to store data inside the card itself. By using smart cards to cipher the externally stored passwords, the password database could still be attacked, but the computational resources required to attack it will be significantly greater given the size of the key. When the passwords are stored inside the card, brute force attacks can be avoided by forcing a maximum number of authentication tries, after which the card will be blocked or the data of the user is deleted.

The solution herein proposed consists of a password manager based on smart cards. In this solution, the passwords are protected inside the card. The passwords can only be accessed after user authentication towards the smart card. We mitigate the risk of accessing the passwords since an attacker needs access to the card, and authenticate himself. In this solution, besides 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 main components, technologies used and the taken trust model. Section 3.2 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. Section 3.3 describes how the proposed solution controls access from applications to the passwords. To enable the proposed solution to be used without requiring a smart card reader, section 3.4 proposes the access to a smart card inside a smart phone.

3.1 Solution Overview

Solution Components The proposed solution is a password manager external to the applications whose purpose is automating the authentication process of the user with different services, without the user having to remember the passwords for those.

The system is composed by two major components. The first is the password manager server. This component runs inside the personal computer of the user and is responsible for receiving requests from client applications to store and retrieve their passwords and to communicate with the smart card. The second component is the password manager applet running inside the card which stores the passwords of the user and answers to commands sent from the server.

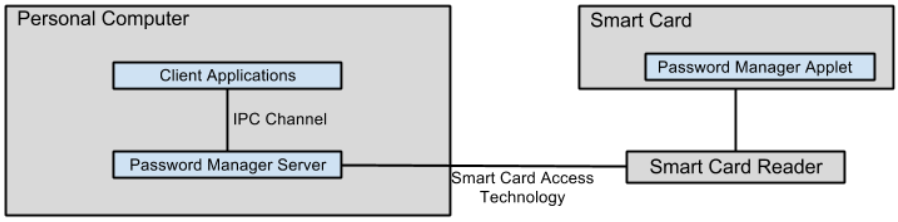

Fig. 2. Password Manager Architecture

Trust Model In order to properly design the proposed solution a trust model must be defined for the system on each of its components. The following defines the trust assumptions defined for 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 assuring no eavesdropping or tampering of messages will be performed. The user may use tamper-evident mechanisms to assure non tampering of the system.
2.2 - An authenticated card is reliable - A card which proved its identity is reliable.
2.3 - Only processes from the same user space can communicate - The used IPC mechanism assures that only processes running over the same user space can communicate with the server.

Password Manager Applet Assumptions

3.1 - The card is tamperproof - It is not possible for an attacker to access the passwords without authentication.
3.2 - No other application internal to the card can access the passwords - If the card supports multiple applications, it also implements a proper firewall between applications.
3.3 - Only the user knows the smart card access password - If somebody authenticates with the user password, he is the expected user.

Components Technologies Different technologies have been chosen to achieve the desired functionalities and the properties required by the trust model.

For the communication between client applications and the password manager, an inter-process communication (IPC) technology is required. It is important for the chosen technology to assure 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 can not be re-used. This property, together with assumption 1.1, which grants that the Password Manager Server is registered before other applications of the user, allows applications accessing the passwords to assure the legitimacy of the Password Manager Server. By registering on the session-bus, it also assures that only applications running in the same user space as the user can access the service.

Regarding the communication between the PC and the SC, it is important to be independent of the manufacturers of the card and the reader. The communication between the password manager server and the smart card is performed using the PC/SC library, providing an abstraction over the used peripherals.

The password manager applet is developed in Java Card Technology and runs inside a Java Card, a SC which runs Java Card applications. This allows the applet to be deployed in SC from different manufacturers as long as they provide a Java Card virtual machine. The Java Card specification allows multiple applets to be deployed inside the same card. Although multiple applets can be deployed on the same card, Java Cards implement a firewall, allowing applets to access only data their own, providing the assumption 3.2.

3.2 Communication Security

In the previous sections no trust assumption for the communication channel between the password manager server and the smart card have been specified. While in some smart card solutions this secure channel is not required, since there is no confidential information flowing from or to the card, as is the case of an application in which the card would only encrypt an hash of a message with its private key. In the case of the proposed system there is confidential information flowing to and from the card.

Another important property to take into account in the considered scenario is that the connection with the smart card cannot be trusted. As in the previously example, when sending an hash value to the card, no confidential information would be leaked. Herein, if the user smart card is replaced by a malicious card, either by physically replacing it, or by a logical channel, this malicious card would be able to receive the user credentials. This allows us to conclude that in this case, as opposed to other common smart card scenarios, mutual authentication is needed. This means that besides the card demanding user authentication, the user application must also be assured of the smart card identity. Assuming a non secure communication channel also implies that the authentication process must be protected, thus the pin or password cannot be sent in clear text.

To achieve this the proposed solution uses Encryption Key Exchange[20] as the authentication protocol. This authentication protocol is used since it provides mutual authentication, assuring the identity of both the user and the card, while protecting the secrecy of the user authentication Pin/Password, which is never sent in clear text.

After user authentication the data must be transferred by a secure channel. This is achieved by encrypting the data in order to avoid message leaking. Besides authentication, the previously specified authentication protocol also generates a temporary session key. This session key is used to cipher the messages when sending or receiving the passwords from the SC, assuring the confidentiality of the messages.

In addition to confidentiality, it is also important to assure freshness of the messages, avoiding replay attacks, and message authentication, assuring the message was not modified in the communication channel. To provide the first property, a nonce field is added to the messages. This nonce is a randomly generated value which is returned by the applet in each message. For the second property, a digest of the whole message is added at the end of the message.

The resulting format of the messages is depicted in figure 3.

3.3 Access Control

Section 2.1 presented two password managers external to applications which control the access to the passwords. The access control policy Keyring performs correspond to the access control policy of classic operating systems, it allows access based on the user space of applications. The one performed by KWallet can be considered stronger since, besides the user space restraints, the user needs to authorize

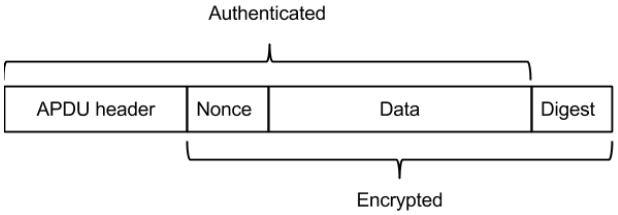

Fig. 3. Message Security

each accessing application. The remaining problem is that this authorization is only for the current session and the information provided to the user is the name of the application. Since an attacker can define the name of its application, it will be able to perform masquerading attacks which may trick the user into believing in that application.

The proposed solution uses mechanisms presented in 2.3 to perform access control based on user permission. The user defines which applications he wants to grant access to the passwords. When a process wants to access the passwords, the server will assure the identity of the process by checking if permissions to this application were granted by the user. To perform this assurance, the server uses DBUS to get the PID of the client process and uses procfs to access its executable code. It then calculates the hash of the code and compares with the user registered values. The hash calculation avoids attacks which modify the trusted applications.

By using these mechanisms the user can detect and avoid masquerading attacks.

3.4 Smart Phone Integration

The usage of smart card solutions depends on the presence of a smart card and its corresponding reader. Although the smart card can be easily transported in the wallet of the user, most computers do not provide a smart card reader either embedded or attached. The transportation of such a reader by the user may be considered as a restraint for the usability and availability of the proposed solution.

To enable the usage of the proposed solution without requiring a smart card reader, a module was created to enable access to a smart card contained in a smart phone. This module was develop using Virtual Smart Card Architecture, a solution that implements a PC/SC reader handler and allows virtual smart cards to be connected.

The developed virtual smart card forwards the messages from the computer to an App on the smart phone. This App then forwards the messages to the smart card, returning its response to the computer. The architecture of the integration with the smart phone is depicted in figure 4.

The developed module assumes a smart phone with the Android operating system and a Go-Trust® microSD with embedded java card.

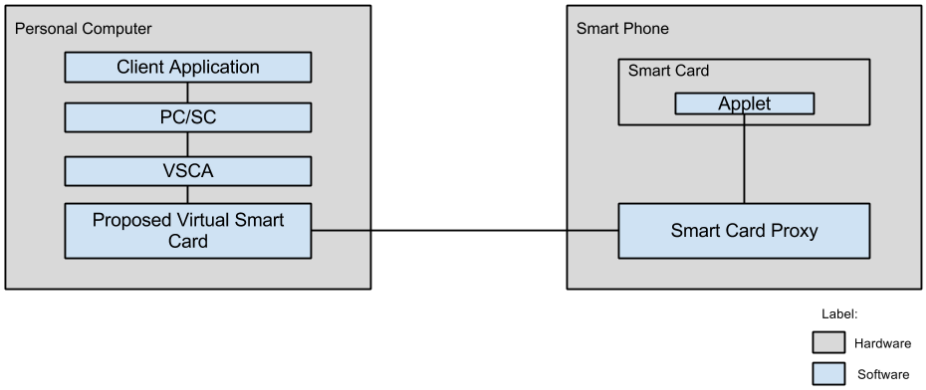

Fig. 4. Message Security

4 System Deployment

To evaluate the proposed solution, an extension for Mozilla Firefox has been implemented, replacing the default password manager. To access the password manager server, the extension uses C++ stubs automatically created by the qdbusxml2cpp tool, allowing access to the service as a common C++ object which abstracts DBUS technology. For a developer to integrate the proposed solution with a different application, several tools are available that create stubs for different technologies and programming languages, providing easier integration.

In terms of usability, the usage of the implemented extension is transparent. Users interact with the password manager functionalities of the browser as if it was not replaced, having its default usability.

The performance of the system was measured with the applet installed in a Go-Trust®microSD with embedded java card, both when physically connected to the computer and when inserted inside a smart phone. Two benchmarks were performed to measure access times comparing the proposed solution with the Firefox default password manager. Figures 5 and 6 present, in milliseconds, the results obtained for the benchmarks when accessing and storing credentials. These results conclude that the infrastructure associated with the proposed solution affects the performance. Although, the measured results are still inferior to one second, not affecting the usability of the system.

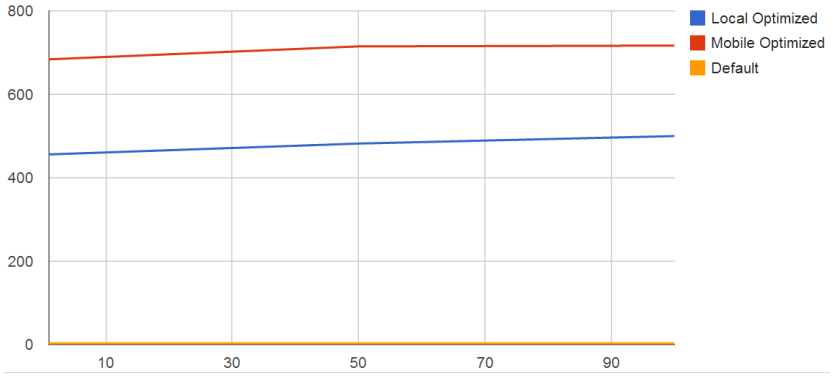

Fig. 5. Get Operation

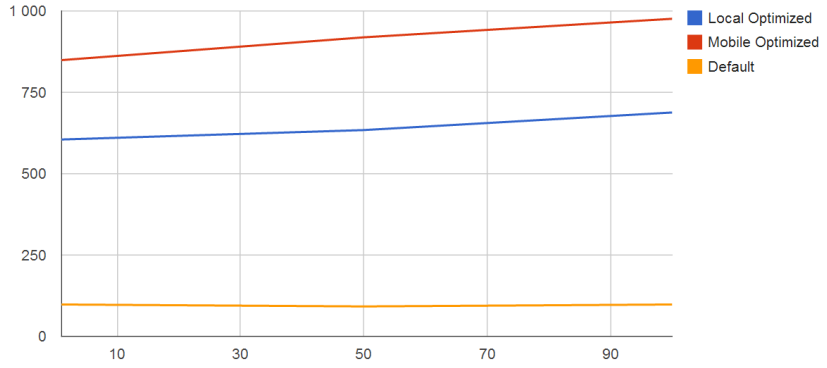

Fig. 6. Set Operation

5 Conclusions

To avoid the memorization of multiple passwords, users tend to reuse the same password over different services, causing a possible vulnerability for those.

Several solutions have been proposed to allow the authentication of multiple services keeping the memorization effort of a single password. SSO are responsible for authenticating the user for different systems, but are susceptible to DOS attacks, and in case of password leaking, an attacker is able to access all the services of the user. Hash based password generators generate different passwords from a master password. In case of disclosure of one of those, an attacker may perform a brute force attack to obtain the master password. Password managers are applications that store passwords, usually on a database in the user computer. Although being encrypted, these passwords can be target of brute force attacks

By protecting the passwords inside the card, the proposed solution is a password manager able to avoid brute force attacks to those. It implements mutual authentication to avoid leaking the password to non legitimate cards, and assures confidentiality, authentication and freshness to the messages. Besides the mechanisms used to assure the presence of the user, the proposed password manager also implements an access control policy based on the identity of the client applications. Since requiring the presence of a smart card reader may lead to a drop on the availability of the system, the proposed solution was complemented with a module which allows the access to a smart card contained in a smart phone.

References

1. O'Gorman, L.: Comparing passwords, tokens, and biometrics for user authentication. Proceedings of the IEEE **91**(12) (2003) 2021–2040
2. Chang, C.C., Wu, T.C.: Remote password authentication with smart cards. Computers and Digital Techniques, IEE Proceedings E **138**(3) (may 1991) 165 – 168
3. Florencio, D., Herley, C.: A large-scale study of web password habits, New York, NY, USA, ACM (2007)
4. Ives, B., Walsh, K.R., Schneider, H.: The domino effect of password reuse. Commun. ACM **47**(4) (April 2004) 75–78
5. Pashalidis, A., Mitchell, C.J.: A taxonomy of single sign-on systems. In: Information Security and Privacy, Springer (2003) 249–264
6. Halderman, J.A., Waters, B., Felten, E.W.: A convenient method for securely managing passwords
7. Huth, A., Orlando, M., Pesante, L.: Password security, protection, and management (2012)
8. Suoranta, S., Tontti, A., Ruuskanen, J., Aura, T.: Logout in single sign-on systems. In: Policies and Research in Identity Management. Springer (2013) 147–160
9. Zhao, R., Yue, C.: All your browser-saved passwords could belong to us: a security analysis and a cloud-based new design, New York, NY, USA, ACM (2013) 333–340
10. HUBER, M.: Agente secreto. Linux magazine (72) (2011) 46–47
11. Larsson., A.: Proposal for inclusion in desktop: gnome-keyring (2003)
12. Staikos, G.: Kwallet-the kde wallet system (2003)
13. Wang, X., Han, Z., Zhang, D.: Idkeeper: A web password manager with roaming capability based on usb key. In: Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on, IEEE (2012) 1228–1231
14. McCarney, D., Barrera, D., Clark, J., Chiasson, S., van Oorschot, P.C.: Tapas: design, implementation, and usability evaluation of a password manager. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACM (2012) 89–98
15. Markantonakis, K., Mayes, K.: Smart Cards, Tokens, Security and Applications. Springer, Dordrecht (2008)
16. Husemann, D.: Standards in the smart card world. Computer Networks **36** (2001)
17. Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. Security & Privacy, IEEE **7**(1) (2009) 50–57
18. Love, R.: Get on the d-bus. Linux Journal **2005**(130) (2005) 3
19. Mouw, E.: Linux kernel procfs guide. Faculty of Information Technology and Systems (2001)
20. Bellovin, S., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on. (1992) 72–84