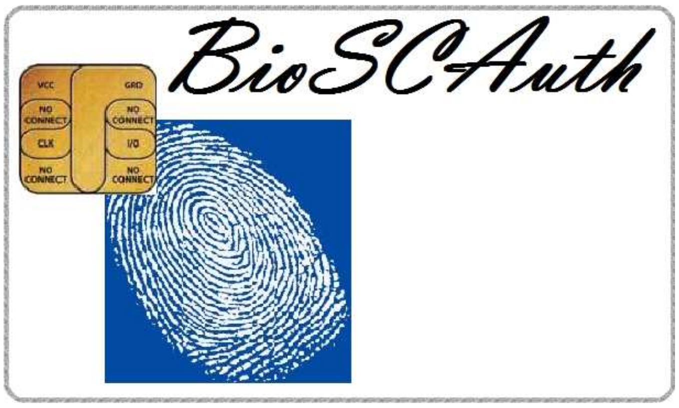
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



Smart Cards: Remote Authentication using Biometrics

Filipe Alexandre Ricardo Marques

Dissertação para obtenção do Grau de Mestre em

Engenharia de Redes de Comunicacões

Júri

Presidente:	Prof. Doutor Paulo Ferreira
Orientador:	Prof. Doutor Ricardo Chaves
Co-orientador:	Eng. Carlos Lima
Vogal:	Prof. Doutor Carlos Ribeiro

Outubro de 2011

Acknowledgments

I thank my family for all the support provided throughout all my life.

My parents for the emotional support given through motivation and values, indispensable for my personal development, which allowed me to consolidate attributes such as persistence and a good sense of responsibility that have been essential in moments of discouragement.

To my godmother and my sister for all the affection and motivation they gave me during the course of my personal evolution.

A big acknowledgment to all my friends for making my life a better life, giving me joy and making me laugh every day, and for always being there for me in tough times.

I would also like to thank my friend Fábio Constantino in particular, for his dedication in revising the English in my thesis, demonstrating great camaraderie and spirit of sacrifice, making his time available to help me in a crucial time.

A big acknowledgement is also in order for my supervisor, Ricardo Chaves, for all his help and accompaniment given throughout the development of my thesis, through his sound advice and guidance, always pointing me in the right direction when facing difficulties.

Last but not least, I would like to thank the company Zetes Burótica for all the material provided. I thank in particular, Eng. Carlos Lima and Eng. Nuno Boavida for being serviceable in the enlightenment of technical issues, making the resolution of some problems easier.

This thesis is dedicated to my parents,

Amadeu Figueiredo Marques,

and

Maria do Céu Pires Ricardo Marques,

for making my education one of their lives' most important priorities,

enduring sacrifices for which I will be eternally grateful.

My thesis is also dedicated to my late grandfather that throughout all

his life thought me many important values.

He was an example of integrity, and a great human being.

Abstract

This thesis proposes an architecture for a secure and reliable remote authentication system, based on local biometric signature validation. Biometric systems are a more secure and reliable authentication mechanism than those that use PIN and passwords. This authentication is based on something that is unique to each person and cannot be borrowed, ensuring the presence of the specific person during the authentication process. Additionally, biometric systems are a more elegant approach to authentication, eliminating the need for the user to memorize any secret information, protecting him against the theft of his secret.

Despite the advantages of biometric systems, they are not used in remote authentication applications, given that it is not viable to send the biometric template via remote connections.

The solution herein proposed is based on the utilization of Smart Cards in a PKI structure, in order to perform a remote authentication. The private key of the user is thus used by the Smart Card to sign a challenge sent by the remote server, proving his identity. This key is protected by a local biometric authentication, ensured by the Smart Card.

The proposed system uses available technology and aims to be compatible with existent solutions.

Keywords

Smart Cards, Java Cards, Biometrics, Fingerprints, Match-On-Card, Remote Authentication

Resumo

Esta tese propõe uma arquitectura para um sistema de autenticação remota seguro e fidedigno baseado numa validação biométrica local. Sistemas biométricos representam um mecanismo de autenticação mais seguro e fiável do que os que utilizam PIN e passwords. Esta autenticação é baseada em algo que é único para cada pessoa e não pode ser perdido ou emprestado, assegurando a presença física do utilizador durante o processo de autenticação. Adicionalmente, sistemas biométricos são uma abordagem mais elegante para a autenticação, eliminando a necessidade do utilizador de memorizar qualquer informação secreta, protegendo-o contra o roubo do seu segredo.

Apesar das vantagens dos sistemas biométricos, estes não são utilizados em aplicações de autenticação remota, dado não ser viável enviar o modelo biométrico do utilizador através de uma conexão remota.

A solução proposta nesta tese utiliza Smart Cards numa estrutura PKI, de forma a ser possível realizar uma autenticação remota. Como tal a chave privada do utilizador é utilizada pelo Smart Card para assinar um desafio enviado pelo servidor remoto, provando assim a sua identidade. Esta chave é protegida por uma autenticação biométrica local, assegurada pelo Smart Card.

O sistema proposto utiliza tecnologia disponível e tem como objectivo ser compatível com soluções existentes.

Palavras Chave

Smart Cards, Java Cards, Biometria, Impressão digital, Match-On-Card, Autenticação remota

Contents

List of Figures

List of Acronyms

MOC	Match-On-Card
GUI	Graphical User Interface
JNI	Java Native Interface
SCP	Secure Channel Password
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission
ITSEC	Information Technology Security Evaluation Criteria
CC	Common Criteria
EAL	Evaluation Assurance Level
PIN	Personal Identification Number
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
PHP	Hypertext Preprocessor
IDE	Integrated Development Environment
PKCS#11	Public-Key Cryptography Standards
PC/SC	Personal Computer/Smart Card
MUSCLE	Movement for the Use of Smart Cards in Linux Environment
RMI	Remote Method Invocation
OCF	OpenCard Framework
API	Application Programming Interface
JC-BioAPI	Java Card Biometric Application Programming Interface

SIO	Shareable Interface Object
PKI	Public Key Infrastructure
CA	Certification Authority
APDU	Application Protocol Data Unit
TPDU	Transport Protocol Data Unit
eID	Electronic Identity
e-service	Electronic Service
POS	Point Of Sale
ICC	Integrated Circuit Card
IAS-ECC	Identification Authentification Signature - European Citizen Card
IAS	Identification, Authentication, and Signature
SDK	Software Development Kit
JCDK	Java Card Development Kit
JCWDE	Java Card Workstation Development Environment
C-JCRE	C language Java Card Runtime Environment
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
CAD	Card Acceptance Device
CPU	Central Processing Unit
NPU	Numeric Coprocessor
SPU	Standard or Proprietary Use
GND	Ground
Vcc	Common-Collector Voltage
I/O	Input/Output
CLK	Clock
RST	Reset

RFU Reserved For Future Use

CAP Converted Applet

RAM Random-Access Memory

ROM Read-Only Memory

EEPROM Electrically-Erasable Programmable Read-Only Memory

RF Radio Frequency

ATR Answer To Reset

SCOS Smart Card Operating System

FMR False Match Rate

FNMR False Non Match Rate

FTE Failure To Enroll

FTA Failure to Acquire

JCRMI Java Card Remote Method Invocation

MF Master File

DF Dedicated File

EF Elementary File

FID File Identifier

AID Application Identifier

SFI Short File Identifier

RID Registered Application Provider Identifier

PIX Proprietary Application Identifier Extension

VB Visual Basic

JML Java Modeling Language

TLS Transport Layer Security

SSL Secure Sockets Layer

1 Introduction

Contents

The fast growth of networks and remote services available every day, trigger the spotlight over public networks such as the Internet. However, the expansion of this virtual world is not only composed of business and convenience opportunities, since such as in a real physical environment, it grows with an ample opportunity for profitable exploitation of vulnerabilities by malicious users. Therefore, the need for security mechanisms protecting the online services, and the information stored by them against informatics attacks emerges. Only with efficient and convenient security mechanisms it is possible to provide the necessary trust, for the sustained development of this emerging world.

The security in computational systems is concerned with the defense, among other areas such as natural disasters and predictable failures, against unauthorized activities. The unauthorized activities can affect the proper control of information access, or allow its alteration by unauthorized users, as well as compromise the adequate usage of the available resources. In a network environment, the information can be stored in a remote server or be in transit over a network link. Therefore, the illegal access to private information, and the possibility for an unauthorized user to edit or delete reserved data are some of the issues that the existent security mechanisms try to solve[45].

The Network security is mostly composed by the policies and measures taken in order to prevent and monitor unauthorized access, misuse, modification, or denial of network resources. The efficient implementation of an access control mechanism can avoid the unauthorized disclosure, destruction, alteration, or copy of private digital information. Since the access control mechanisms require a previous authentication of the user in order to identify who is trying to execute a specific operation or gain access to a particular resource, it is indispensable, among other measures, to ensure ever more secure and reliable user authentication mechanisms in remote scenarios[45].

This chapter is intended to introduce the remote authentication solution proposed on this thesis. There are three main aspects being covered by this chapter. The first section, 1.1, presents the motivation for the development of this thesis, and an overview of the developed research. An overview of the proposed solution, covering the goals that the solution proposes to achieve, and the main contributions to the current authentication systems, are presented in section 1.2. The final section of this chapter, 1.3, presents the structure of this document.

1.1 Background and Motivation

Password authentication schemes are the best-known, and still one of the most used authentication mechanisms in contemporary computer systems [4]. In these authentication schemes, the user has to present the correct secret for his specific identity. The first authentication system required a verification table, on the server side. This table had each user ID linked with his correspondent password. During the authentication, the remote server would consult the verification table to check if the received password is equal to that linked with the ID claimed. This solution is both not scalable and insecure, since the verification table grows with the increase of users, and the user's password is sent over an unsafe network link.

Lamport improved this system in 1981 at [21], by proposing a mechanism in which the password of the user is used to generate a sequence of secrets through hash chaining, such as used by the S/Key OTP (One-Time Passwords) application [16]. However, this system also required a verification table entry for each user, which is not scalable. Besides that, if the server was compromised, and the entry for a user was changed with malicious intent, all the security of the system would be compromised too.

In order to solve both problems, the lack of scalability from the verification table and the need for the server to keep sensitive data from the users, new authentication solutions were developed. On these newer solutions, the user authenticates locally on a trusted token, using his Personal Identification Number (PIN) or Password, having this token perform the remote authentication afterwards, usually using asymmetric cipher operations. An identity token, security token, access token, or simply token, is a physical device that performs or aids authentication. This token can be a secure storage device containing passwords, such as a bank card, remote garage door opener, or even a Smart Card [30]. Due to their design and architecture, Smart Cards can provide identification, authentication, secure data storage, and application processing [34]. These trusted devices can securely store crucial information such as user identification and private keys, and provide the necessary computation to perform security algorithms such as digital signing and cipher operations.

Many authentication mechanisms using Smart Cards and the user password have been developed [6, 7, 20, 22, 23, 29, 39]. Some of these mechanisms allow the user to choose his password freely, while others force the user to use a password generated during the registration process. Usually, this password is then used in the authentication operations carried out by the Smart Card, generating cryptograms that can be authenticated by the remote server.

However, the use of passwords and PIN in remote user authentication systems is not a perfectly reliable solution, since they cannot ensure the presence of the user being authenticated. This happens since most of the available mechanisms for the user to insert his password are not safe, allowing the theft of his secrets with simple techniques such as shoulder surfing. The passwords can even be susceptible to dictionary attacks, and are easily transmitted to a friend, lost, forged or stolen, thereby being unable to provide non-repudiation of the user.

Therefore, the development of the authentication mechanisms and techniques, over the years, has lead to the use of the human biometric characteristics to perform the authentication. The biometric authentication provides great benefits, making the authentication mechanisms more reliable and easy to use by the users, being a good approach to replace the password-based authentication systems [42]. Some of the above referred solutions, [6, 7, 22, 23] also use the user biometric verification in their systems, to ensure the presence of the user during the authentication, while not dismissing the password as a key component of the authentication.

In all the previous approaches, the local authentication of the user requires his password, forcing him to memorize it. This can be a problem mainly for elderly people, having memory problems. The basic issues with passwords can be easily understood, since they are related with the fact that a memorable password can often be guessed or searched by an attacker, and also with the fact that a long, random, or changing password is difficult to remember by the user [1, 23, 30]. A biometric authentication system, on the other hand, is based on a person's physiological and behavioral characteristics, being intrinsically related to a specific user. As such, biometric systems have good advantage on commodity for the users, removing the need for the user to memorize any additional information. Furthermore, since they are specific, unique and non transferable for each individual, they improve the security of the authentication, and can be used to assure that the user being authenticated corresponds to the person performing the authentication, since no one else can insert his biometric data [14].

In an authentication system that exclusively uses biometrics, it is not safe to send the biometric data of the user over a communication channel on an unsecure Network. There are also no security mechanisms for the remote server to trust in a remote biometric matcher. Therefore, the above referred systems using biometrics such as [22], use the user biometric data to generate random information useful for the cryptographic systems, but still requiring a user password for the generation of the cryptogram sent to the server. This password is generated by the server side, usually using signatures created with its private secret. At the authentication time, the server will then validate the user password through the sent cryptogram.

Even though the Smart Card performs a local user biometric authentication, ensuring his presence during this process, in the remote authentication part of the referred solutions, the biometric data is merely used as a random number generator, having the security and authenticity of the remote authentication procedure ensured with the user password.

1.2 Proposed Work Goals and Contributions

In this work we propose a biometric remote authentication system, which only requires the biometric data of the user in the authentication process, discarding the need for the use of passwords. This solution overcomes all the issues regarding the security of the biometric template of the user, and the confidence in a remote biometric matcher.

One of the goals of the proposed solution is to use existing technologies, thereby ensuring low development costs, and a high compatibility with existing remote authentication systems. This solution is therefore designed to be considered as a reliable possibility to integrate the future authentication mechanisms in already existent remote scenarios. The proposed solution is based on the use of Smart Cards, in which the private key and biometric template of the user are stored. The Smart Card of the user is afterwards deployed in a Public Key Infrastructure (PKI) [32].

There are similar solutions in which the Smart Card is deployed in a PKI, being able to perform the remote authentication, authenticating the user with his PIN. The Smart Card possesses the private key of the user, as well as a public digital certificate signed by a trusted Certification Authority (CA). The remote authentication process is based on a challenge response, in which the response corresponds to the digital signature of the challenge, signed with the private key of the user performed by the Smart Card. Not even the user has access to this private key, ensuring that only the Smart Card knows it. The user authentication is performed locally in the Smart Card, via his PIN.

One of the real applications using this remote authentication approach are the Electronic Identity (eID) cards. These cards are responsible for the cryptographic computations required for the identification and authentication of a citizen. An example of this solution is the Portuguese citizen card, where a Java Card stores the user information, fingerprint templates, PIN, private keys and public certificates signed by a CA. The applet installed to perform remote authentications follows the Identification Authentification Signature - European Citizen Card (IAS-ECC). These Smart Cards also possess the capability to manage biometric templates, and to perform Match-On-Card (MOC) operations.

The solution herein presented consists of replacing the local PIN authentication, in systems such as the Portuguese citizen card, for a local biometric authentication. This is achieved by only allowing a remote authentication after a biometric authentication is also performed. In this solution, an applet implementing the IAS standard is responsible for the remote authentication of the user. The IAS only has the private key released to sign the challenges from the remote server, after a successful user biometric authentication in the MOC applet.

The obtained results show that it is possible to use biometric authentication in remote scenarios, taking advantages of their potential as a more secure and easy to use authentication mechanism. It also eliminates the need for users to have different passwords in different systems.

1.3 Document Structure

This document consists of five additional chapters.

Chapter 1.3 presents the research performed during the development of this thesis, with an analysis of the current state of the art. It presents the evolution of the authentication mechanisms and the challenges of dealing with biometric authentication systems. It also covers the Smart Card technology, the benefits of using the Java Card Platform, and also presents an overview of existent solutions.

In Chapter 2.4 the proposed remote authentication system is described. The authentication scheme and the proposed architecture of the solution are presented, covering the needed Smart Card modules and Middleware functionalities.

The implementation of the prototype of the proposed solution is described in chapter 3.3. This section covers the challenges found in the different phases of the implementation, and the technologies used in the development of the proposed solution.

Chapter 4.4 contains the assessment of the proposed solution. Here, the developed test environment and results obtained during the evaluation process are described. The security analysis of the proposed solution and implemented prototype is also performed.

The main conclusions obtained with the developed work are presented in chapter 5.4. This chapter contains an overview of the developed work on this thesis, as well as the enumeration of the main contributions of the work herein presented.

2 State of the art

Contents

This chapter presents the research performed during the development of this thesis. The current state of the art covering existent technologies and techniques used on the development of the proposed solution are presented. This chapter also addresses the existing solutions providing features similar to those that serve the purpose of the proposed solution.

The proposed solution requires the development of a remote authentication system capable of using the user biometrics, taking advantage of the biometric authentication benefits. The state of the art describes the existing systems and technologies supporting the current authentication systems, in order to understand what can be exploited from the available solutions. Therefore, the research herein presented, covers the authentication mechanisms used in remote scenarios showing their benefits, but also analysing the problems that currently are preventing the secure and efficient use of biometrics in current remote authentication systems.

The biometric authentication is analysed, describing the required processes for the development of an authentication system using biometrics. The wide variety of biometric types is approached, giving a bigger emphasis to the fingerprint biometric, since it is the trait used in the implemented prototype. A remote authentication overview is also performed, describing the Public Key Infrastructure (PKI), used to manage and distribute the public keys of the users, and the asymmetric cipher, playing a key role in the signing of challenges.

Smart Cards are electronic devices used in current authentication systems. They are also used in the proposed solution, reason why they are described and analysed in this chapter. This technology is used in the proposed solution to solve the main problems regarding the use of biometrics in remote authentication scenarios, once they are secure and reliable devices capable of providing crucial biometric operations, such as the match on card. Their main advantages are presented, and the existent platforms discussed. It is given more emphasis to the Java Card platform, since it is the Smart Card platform used in the implemented prototype.

The existent solutions presented in this chapter are intended to show the current implemented systems, and their weaknesses. Therefore, they serve as a starting point to help understand what needs to be improved and guaranteed by the proposed solution, taking into account what is used in existent scenarios. The pros and cons of each solution are analysed, being given detail to the requirements defined by the proposed solution that are not supported by them.

The current chapter is organized in 4 sections. Section 2.1 describes the state of the art of the current technologies and mechanisms used by authentication in general. The Smart Card technology is presented in section 2.2, where the Smart Card structure, the main benefits of this technology, and the existent platforms with their added value for the proposed solution are analysed. Section 2.3 covers the research of the existent technologies, performing authentication using biometrics, where the main aspects that differentiate them from the presented solution are analysed. To finalize, the main conclusions described on the state of the art developed in this chapter are presented in section 2.4.

2.1 Authentication

This section presents the state of the art on the technologies used to authenticate the user in remote scenarios. The importance of the authentication in the current systems is described, given the rapid growth of insecure computer networks, leaving users vulnerable to attacks. The benefits of the current mechanisms are also discussed in order to understand what needs to be improved in future work.

The authentication of the user in remote systems has not always been a big concern, given that the networks were initially created in a trusted environment, where the goal of all users was just the proper operation of services. However, the development of networks, and the use of online services by a wide range of users on a variety of services, has led to the emergence of malicious users, with the intention of violating the privacy and security of other’s information. The authentication of a user before a remote server is a way for the server to know who is interacting with it, in order to personalize the interaction, but also to be sure that it is not giving information or access to a wrong user. Therefore, the development of secure and reliable authentication mechanisms has been a major concern for researchers in the area of computer security, in recent times.

2.1.1 Overview

The typical authentication process of the user in a remote scenario is performed by the user sending his identity, and proving it to the remote server. To prove the identity of the user, some information has to be sent to the remote server that only the user being authenticated is able to know or generate. Therefore, the authentication mechanisms are highly dependent on security functions related to the cipher of information and generation of random numbers, as well as hash functions.

Given the development of the authentication mechanisms there are different types of authentications. Therefore, the authentication factors are usually grouped into three categories related to what the user knows (e.g., passwords), what the user has (e.g., tokens), and who the user is (e.g., biometrics).The first, regarding what the user knows, deals with the need for the user to memorize his secret, and becomes less secure each time they are used since it produces more samples for the attackers to study. The second, is related with what the user has, forces the user to keep a physical token such as a Smart Card in order to perform the authentication and in some cases, the authentication system can be compromised if the user loses his token, and an attacker finds out how to use it. The third type of authentication takes into account what the user is, considering something that is part of his identity such as his biometrics. This authentication factor looks for unique features in each person, becoming a more difficult to copy or forge mechanism. However, if the security of a biometric template is compromised, it is not easily replaced. These three authentication factors can be used alone, or be grouped, in order to create a multi-factor authentication system where each authenticator factor has to be satisfied, in order for the user to be correctly authenticated [30].

2.1.2 Evolution of Authentication

The initial authentication mechanisms were based on the presentation of a password in plain text, to the remote server. In this context a password can be the definition of single words, phrases, or Personal Identification Number (PIN). After receiving the user password, the remote server compares it with the one stored in a password table, linked to the identification of the user. However, since the techniques of the attackers have been evolving, the need for the protection of the user password during the authentication process arose [21].

Lamport proposed a method of user password authentication over insecure channels, in 1981 at [21]. The main challenges that the author intended to address were the gain of access to the password stored on the system by an attacker, and the resilience to the interception of the communications between the user's terminal with the system, or against the observation of the execution of the password checking program. The author also had identified the problem of the inadvertent use of the user password, for example being guessed by an attacker, but described it as a problem that could not be prevented by any password protocol. The proposed authentication mechanism bases its security in the one-way encryption functions, and it is designed to prevent replay attacks. However, this solution uses a password table to ascertain the legitimacy of the user login, which can cause problems if the attackers are able to modify it. The use of that kind of tables has other inherent problems such as its growth with the increase of users on the system.

To improve the security of the existent remote authentication systems, the Hwang, Chen, and Laih have proposed a non-interactive password authentication system without password tables, in 1990 at [39]. The proposed authentication system is based on Shamir's ID-based signature [41], being easy to verify the authentication of a user by anyone inside the system. This system discards the use of password verification tables by using Smart Cards, and a public key cryptosystem [37], and at the same time ensures the resilience to the personification of a user even if an attacker is able to capture the messages sent during previous authentications. However, this system keeps forcing the user to use a password, and moreover, the user is forced to use a password that is assigned to him by the system. This password is used as a parameter in the cryptographic operations performed by the Smart Card during the remote authentication process.

Jan and Chen proposed a practical system to solve the authentication problems, existent at the time, with the concept of public key distribution, in 1998 at [20]. This system is based on the Diffie Hellman public key distribution system [17], and uses Smart Cards. It provides the possibility for users to change their passwords freely. However, the security of this system is strongly linked to the protection of the private key of the remote server, and continues to require the user password in the login process.

Hwang and Li have also proposed a remote user authentication scheme using Smart Cards in 2000 at [29]. However, this system is based on ElGamal's public key cryptosystem [12]. The scheme is divided in three phases, the registration where the user receives his password, the login in which the user inserts his password to the Smart Card in order to perform the remote authentication, and the authentication corresponding to the remote server validation of the cryptographic operations performed by the user's Smart Card. Although its security is based on a private secret of the remote server, this system could not withstand attacks by user impersonation.

In order to improve the security on the authentication mechanism proposed in [29], Lee, Ryu, and Yoo proposed a fingerprint-based remote user authentication scheme using Smart Cards, in 2002 at [22]. This system is also based on ElGamal's public key cryptosystems [12], however the authentication requires the user identification, password, biometrics, and Smart Card. This system was designed to resist replay attacks, and impersonation of the user. It uses two private secret on the server side, and the Smart Card of the user keeps the public information needed during the authentication process, and also provides the local biometric authentication of the user. The biometric map of the user fingerprint is also used to generate the random numbers needed for the ElGamal's operations. Even with all the improvements of this system, it cannot withstand masquerade attacks, since a legitimate user who registers a legal pair of identity and password can pass the fingerprint verification of his/her own Smart Card, and easily masquerade another valid pair of identity and password without knowing the two secret keys of the remote system [7].

Chen, Jan, and Tseng have also proposed in the same year, 2002, an efficient and practical solution to remote authentications using Smart Cards at [6]. This solution bases its security on the one-way hash functions. Its supports the mutual authentication between the user and the remote server, and the user can freely choose his password. However it uses timestamps in order to avoid replay attacks, which forces the clock synchronization between the entities using the system.

Lin and Lai have proposed a flexible biometric remote user authentication scheme, in 2004 at [7]. This system is an improvement of the authentication system proposed in [22], covering its vulnerability to masquerade attacks. Furthermore, this system enables the users to conveniently choose and change their passwords. This scheme is also based on the ElGamal's cryptosystem, fingerprint verification, and Smart Cards. However, despite all the benefits in security brought by this system, it is susceptible to the server spoofing attack [23].

[23] presents an authentication mechanism proposed in 2010 by Li and Hwang as an efficient biometric-based remote user authentication scheme, using Smart Cards. This solution bases the security of the authentication mechanism in one-way functions, biometric verification, and Smart Cards. This system still relies on user passwords; however enabling the users to change it. Another advantage comes from the fact that this system does not require synchronization of the system's clock.

As it can be seen by the presented evolution of the authentication mechanisms, at least since 1990 the benefits of Smart Card shave been widely adopted. Therefore, many e-commerce applications, network security protocols, and also remote authentication schemes have been designed to use Smart Cards [6].This happens since they can compute asymmetric cryptography operations, being able to authenticate the user remotely. The proposed solution improves the authentication mechanisms already developed, since it releases the user from the obligation of using a password during the authentication. This feature can be achieved by integrating the Smart Card of the user in a PKI [32], and combining the system with a local biometric authentication of the user. The PKI system and the biometric authentication mechanism are described in the following section.

2.1.3 PKI

The goal of a PKI is to enable secure, convenient, and efficient discovery of public keysused in asymmetric cryptographic systems [32]. The asymmetric cryptographic systems imply the use of a public and a private key for each user, unlike symmetric cryptographic systems where each connection between two users requires a shared private secret. In asymmetric cryptographic systems, the private key of a user is only known by him, while the public key, as the name suggests, can be known and used by any user of the system. The use of asymmetric ciphers in authentication systems requires fewer cryptographic keys shared by the users of the system comparing with those using symmetric ciphers, since for N different users it is just needed to share their respectively public keys, i.e. N keys. Although asymmetric cipher is a more effective cryptographic mechanism, providing non repudiation and requiring fewer keys shared among the users, it requires more computational power than symmetric cipher [45].Thus, asymmetric encryption is only used in the process of user authentication and symmetric key exchange and not in the cipher of data during communications.

The main issues regarding the use of asymmetric cipher in remote authentications are related with the restriction of access to the private key of the user only to himself, with the reliable public key distribution, and with the management of the life time of each key pair [45]. A Smart Card electronic device can assure the confinement of the private key of the user, as it will be shown later in section 2.2. The other two concerns can be solved using a PKI, and public certificates.

The user public key can be distributed by hand, by the user personally delivering his public key to the other entity. They can also be embedded in programs such as in the case of the web browsers. Another way to distribute the public key of the users is to use a certificate chain, where each public key is associated with a public digital certificate signed by a Certification Authority (CA) (Certification Authority). Each certificate has the user information, and his public key. If an entity trusts in the CA, will also trusts in the digital certificate signed by the CA, and thus can be sure from who is the public key stored inside. These digital certificates have the validation time of the public key, from which ceases to have effect. The chaining of certification composed by the CA, and all the software and hardware components used in the management and distribution of public keys and digital certificates, constitute the PKI [32, 45].

2.1.4 Biometric Authentication

”Biometric recognition is the use of distinctive anatomical and behavioral characteristics or identifiers, to automatically recognize a person” [25]. The anatomical properties are related to the shape of the body, while behavioral properties are related to the behavior of a person. A biometric authentication system can provide security, efficiency, reliability, and convenience for the users [25]. These advantages come from the fact that biometric characteristics are part of each individual, and specific to each one. In this sense, no one can share their biometrics with a friend, or even lose them. The biometrics are, thus, more difficult to abuse than traditional methods of identification and extremely difficult to guess, share, misplace, copy, or forge [33].

2.1.4.A Biometric Features

The main characteristics desired in a biometric feature are universality, distinctiveness, permanence, and collectability. The universality feature ensures that all people have the biometric, thereby avoiding that the biometric authentication can only serve one group of people. The distinctiveness is the property to ensure that for any two different biometric samples (of different people), there are no two equals. The permanence of a biometric is based on the fact that this feature is time-invariant and thus, an individual can be identified through a biometric sample collected a long time ago. The collectability ensures that biometrics can be taken from their carrier and quantified. The possibility to quantify the biometric traits ensures the ability for an automatic system to perform a biometric authentication of a person [33].

To evaluate biometrics, in order to enable the comparison between them, some characteristics should be taken into account: performance, acceptability and circumvention. Performance measures the accuracy, speed, and resource requirements to perform an authentication based on that biometric. Acceptability relies on the availability of the submission of the biometric collection, presented by the people to be authenticated. Circumvention is the characteristic that makes the comparison of biometrics possible, in terms of immunity to fraudulent methods and attacks [33].

2.1.4.B Authentication Flow

The first requirement of a biometric system is the enrollment phase. In this phase a new entity is registered in the system. Information about the identity of a person is stored associated with its biometric template in a remote data base, or in tokens such as Smart Cards. The enrolment phase is divided into the capture step, feature extraction step, template creation step, and data storage step [33]. In the capture step the capture of the user’s biometric characteristics is performed, where each type of biometric requires its own type of capture mechanism. Normally in this phase, a quality analysis is done in order to determine if the capture made has reached a certain level of quality. This analysis assures that the biometric capture can be used in the later stages, with a high degree of reliability, or determine that another capture has to be made [1].

After the capture step, comes the feature extraction, in which the capture previously done is analyzed. This step consists on the creation of the biometric representation, resultant from the computing of the biometric particularities. This step ends with the creation of the biometric template. The template is the final result of the biometric trait representation, and contains the extracted features in a specific format. The template is, thus, the raw material used in the matching phase of recognition [25].

In a recognition biometric system, two modes are possible: the identification and the verification. The identification is based on the identification of a person in the system, by comparing their biometric sample with all biometrics templates stored in the system database, until one matches. This process can take more time and be more inaccurate than verification, since the match process is done more than once. The verification module, on another hand, corresponds to the comparison between the biometric sample with the biometric template stored in the system data base, associated with the id claimed by the person [33].

Traditional authentication methods work for positive recognition only, while biometrics can be used for negative recognition too. This happens in the identification module, when it is possible to detect that a person is who it denies to be. This feature can prevent that multiple identities of the same person are registered in the system [33].

The recognition phase follows a protocol very similar to the enrolment. First a capture of the user biometric is done, followed by a feature extraction step. The difference is on the final steps, where the template creation and data base storage are replaced by the matching stage. The capture follows the same quality analysis, to provide a reliable biometric data to the feature extraction step. The match module depends on what module the biometric system is. If it is the verification module, the features extracted from the capture are compared with the template stored with the identification of the claimed user. If it is the identification module, the features extracted are compared with all the templates in the system, in order to determine who is the person, or if the person is not registered in the system [25].

2.1.4.C Error Analysis

The most common errors in biometric systems are derived from imperfect image capture conditions, changes in the users physiological or behavior traits, ambient conditions on the capture phase, or by the interactions between the user and the sensor [33]. These type of situations can introduce errors in the biometric data captured, due to the useful information lost in the process. Due to the need for computation and automation of the biometric authentication systems, each biometric must have its corresponding digital representation. Since a characteristic of biometrics is its individuality among each individual, typically the space needed to store the biometric information has to be big enough. However, in the process of a digital representation, there are limitations inherent to the technology, and to the algorithms used, which results in errors of limited representation. In some biometrics, such as face analysis, there are also errors associated with its variance, which can have influence on the applicability of this type of biometrics in some types of applications [25].

The degree of reliability of a biometric authentication system can be measured by the False Match Rate (FMR) and False Non Match Rate (FNMR) (False Non Match Rate) [33]. These measurements are associated with the final result of a biometric system, and with their assertiveness. In this sense, the FMR is the rate of biometric measurements and analysis saying that the biometrics of two different persons, belong to the same. By the other hand, the FNMR is the rate of biometric measurements and analyses saying that two biometric measurements from the same person belong to two different persons. The accuracy of the biometric systems also takes into account the Failure To Enroll (FTE) errors, and the Failure to Acquire (FTA) errors, coming from failures in detection, capture or processing of the biometric traits [25].

Due to the errors and limitations inherent to the biometric systems, the matching module cannot be based on a rigid scheme or algorithm. Thus, the result of the comparison between the template storage in the enrollment phase, and the biometric sample, is a matching score. This score is then compared with a threshold defined on the system, in order to determine the final result [33]. Because of the errors caused in the capture, features extraction, and matching process, biometrics technology cannot be used to establish an absolute identification of a person. However, the positive and negative identification error rate is very small, making the biometric systems sufficiency reliable to be used in the highest security systems in the world [19].

There are a large collection of biometric technologies types that can be used for authentication and identification of an individual, like facial feature, voice print, fingerprint, iris scan, retina scan, hand geometry, written signature, keystroke dynamics, lip movement, thermal face image, thermal hand image, gait style, body odor, DNA scan, ear geometry, finger geometry, palm geometry, vein pattern [9]. All these biometric features have their advantages, disadvantages and different types of error rate. However, the fingerprint is the most widely used biometric feature. In fact, some times, people confuse biometrics with fingerprint. This is because fingerprints have the distinct, and persist features of a biometric, and have been used for over 100 years [25]. With such time, and impact in society, fingerprint technology has reached a high maturity, and there are many low cost equipment available [25]. For these reasons, fingerprint biometrics is analyzed next.

2.1.4.D Fingerprints

Fingerprints are one of the most used body characteristics in biometric systems. They have the main features to be considered a very useful and reliable source of authentication, and are already a mature technology. The main biological principles of fingerprints are their individual epidermal ridges, and furrows in each person, and despite the existence of a wide range of characteristics to analyze in fingerprints, they are within limits that allow their classification. Another characteristic lays on their invariance, making the configurations and minutiae, permanent and unchanged in a person life, unless a person has a cut or a burn on his finger [19].

A biometric authentication system based on fingerprints basically follows the structure of a normal biometric system, where the enrollment is an important and indispensable phase. This phase can have errors associated with the capture of the biometric traits, caused by the different positions of the finger on the sensor, or by the low quality of the fingerprint image, among other problems surrounding biometrics capture. However, once this branch of biometrics has been studied for over 100 years, it has a high maturity. Besides the maturity in the analysis of this biometric, the low cost of their capture equipment, and the global knowledge about it, make fingerprints one of the most currently used biometrics [25].

The capture of a fingerprint can be performed in two main different ways. The goal is the same, scan an image of the finger to be analyzed by a computer. Therefore, the capture can be done through the scan of an image in which an inked finger is marked, or through the direct scan of the fingerprint by a sensor connected to a computer [42]. There are various types of fingerprint sensors. Among them are electronic field sensors, optical sensors, thermal sweeping sensors, and synthetic generators [3]. Each one of these capturing techniques has their own technologies, price, and error percentage, therefore providing different performances.

After the capture of the fingerprint, comes the feature extraction part, corresponding to the analysis of the image scanned. This analysis tries to extract useful information about the fingerprint, in order to get authentication data from it. This process is not always done in perfect conditions due to the errors in the previews capture phases [33]. Apart from this fact, it has to be considered the fact that the space to store the template is limited, creating the need to filter the information collected. Therefore, the distinctive information available in a fingerprint is not totally utilized sometimes [25].

The feature extraction and analysis phase consider two levels of information on each fingerprint image. In a first level the ridge line flow is considered, ranging between left and right loop, whorl, arch and tented arch. These features are not enough to reliably identify a person, however, they could be useful to eliminate some candidates and accelerate the match process. The second level of the analysis could get up to 150 different ridge local characteristics between bifurcation, ridge ending, dot, ridge crossing, etc. All these features are collected with their type and reference location, to be stored in a biometric template. This template is much more accurate than the result of the first analysis, and thus, may provide enough information to reliably identify a person [25].

2.1.4.E Remote Authentication Scenarios

Despite the fingerprint biometric systems being a secure and reliable system, comparing with the usual security systems based only in passwords, the consequences of a forged biometric are a bigger issue. In a password based system the password is changed and the problem is solved, on the other hand when a biometric system is compromised, the template forged cannot be used again. Forging a fingerprint is difficult, being necessary expertise and laboratory equipment to create a fake finger, and there are no economies of scale to repeat easily this attack [33]. Moreover, it is possible that some biometric sensors have protection mechanisms such as verification of blood pressure, conductivity, temperature, and detection of the human skin [24]. Beyond that, some systems use the combination of more than one biometric system in order to increase levels of security in the authentication systems [33].

Despite all the advantages in the use of biometric authentication, this approach is not usually used in remote authentication systems. The reason behind the reluctance in the use of biometrics in remote systems is related with the concern regarding the protection of the biometric template. As seen previously, once compromised, the biometric template may not be able to be used again. There are two main concerns for the usage of the biometrics in remote authentication systems, who captures the biometric data, and who perform the match operation.

The biometric capture has to be performed by a biometric sensor placed where the user is during the authentication process, once it has to have direct access to him. Therefore, the remote server has to trust in this entity as a reliable biometric sensor, which is one of the bigger concerns. Apart from this, it is necessary to perform the match of the user biometric data, against the previous enrolled template. If it is the remote server performing the biometric match, it raises the issue of having the biometric template being sent to the server over an insecure network link. Even if the template is sent over the protection of a secure channel, if the security of the channel is compromised in the future, the template is also compromised, and thus can never be used again[42]. The alternative would be for the server to trust in another entity to perform the local biometric authentication of the user, and then inform the server if the biometric authentication of the user was successfully performed. This option also requires the server to trust in a remote entity, responsible for performing the match operation. Moreover, the biometric sample captured of the biometrics of the user has to be fresh, meaning that it has to be done during the capture of the user's biometric information, ensuring that the biometric template used in the match operation was really captured in the moment of the authentication [Republic].

Some solutions have been studied lately, in order to try solving these issues, and allowing the use of biometrics in remote authentication systems. One of the solutions presented by Dodis, and Reyzin and Smith in [11], showed how to use biometric data to securely derive cryptographic keys, being then used for the purposes of authentication. These solutions are based on the generation of public information from the biometric template, bearing in mind that this is not exactly reproducible. This public information may be subject to modification by an attacker. In addition, errors associated with existing tolerance guaranteed by these systems, can generate a large number of false negatives. This solution maintains the need for the remote server to trust in an entity that captures the user's biometric data.

The proposed solution in this thesis is based on the use of a Smart Card, to perform the local biometric authentication of the user. The next section 2.2 presents the Smart Cards as a reliable source, able to perform the required security operations.

2.2 Smart Cards

This section has the purpose of resume an overview about the current state of the art of the Smart Cards. The main benefits and utility of the Smart Cards in current society are presented. It is also showed the main characteristics of the Smart Cards, as well as some operating systems and platforms developed to run on these electronic devices. It is given more detail on the Java Card platform, once it is the smart Card platform used in the implementation of the prototype of the proposed solution.

The Smart Card technology is used in the proposed solution, enabling the securely use of the biometric authentication in remote scenarios. In the proposed system, Smart Cards are responsible for storing the user biometric template, and for computing the match on card operation. This section describes the importance of these devices in current days, and the main benefits that makes them so popular and used. It is shown how Smart Cards are a very useful instrument in people lives, protecting against attacks on information security, being at the same time small, portable, and secure electronic devices [26].

2.2.1 Overview

In general, cards can be divided into cards with chips, and cards without chips such as the magnetic cards. The chip cards, or Integrated Circuit Card (ICC) can be considered memory cards or processor cards, depending on their chip structure and features. The processor cards have the ability to store information, and also to process it. This type of cards can be further subdivided into processor cards with or without coprocessors. The coprocessors are responsible for the execution of the asymmetric cryptographic algorithms. The cards with a chip usually communicate through the use of contact pads. However, these cards can contain an embedded antenna to perform the communication with the host side, instead of using the contact pads attached to the chip [5, 34]. The cards with antenna and contact pads can be called by combined or hybrid cards, depending on the amount of chips that they have. The combined cards contain one chip that can be accessed through either contact pads, or an embedded antenna, while hybrid cards contain two or more embedded chips, such that some communicate through the antenna and others through its contact pads.

The development and functionality of Smart Cards are strongly driven by international standards. Therefore, the International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) 7816 is the standard for contact Smart Cards, defining the physical characteristics of the integrated circuit cards, their dimensions and the location of the contact pads, as well as specifying the electronic signals and transmission protocols used in the communication with the host side. Likewise, the ISO/IEC 14443 is the standard for contactless Smart Cards, specifying their physical characteristics, the radio frequency power and signal interface used for the communication with the host side, as well as defining the initialization requirements, the anti-collision mechanisms, and the transmission protocol specifications.

Despite the existence of various types of cards, to be considered a Smart Card, a card has to respect some important requirements, such as those defined in [26]. By definition, a Smart Card is an electronic device that can participate in an automated electronic transaction, with security, and is not easily forged or copied. In this sense, a Smart Card must have memory and a microprocessor, to be able to store and compute data, and to assure the execution of some security algorithms and protocols. A Smart Card has also to be a tamper-resistant system, in order to resist to the physical attacks made directly to the chip [26]. This definition makes impossible for the magnetic stripe cards to be considered Smart Cards, once they do not have processing power, and they can be easily copied or forged [5]. Even some chip cards cannot be considered Smart Cards, if they are only memory cards without a microprocessor to ensure the requirements defined above [26].

2.2.2 Technology Specifications

A contact Smart Card can have six or eight visible contacts points: Common-Collector Voltage (Vcc), Reset (RST), Clock (CLK), Ground (GND), Standard or Proprietary Use (SPU), Input/Output (I/O) and 2 Reserved For Future Use (RFU). The Vcc is used to supply voltage to the chip, the RST is used to reset the microprocessor, the clock supplies the external clock signal to derive the internal signal of the chip, the GND is the reference zero voltage, the SPU is for standard or proprietary use and is optional, the I/O is used to transfer data and commands between Smart Card and the host side in an half-duplex mode, and the two RFU points are reserved for future use, being considered auxiliary points [34, 35].

The efficiency of a Smart Card is largely determined by its microcontroller. The microcontroller's chips are specially adapted for each Smart Card, in terms of electrical and physical parameters, such as the maximum current consumption, the range of allowed clock frequencies, and the allowable temperature range. Smart Card microcontrollers are especially hardened against attacks, including the detection of under voltage or over voltage conditions, and the detection of clock frequencies outside the specified range. Therefore, the microcontrollers usually incorporate light and temperature sensors, in order to recognize the attacks, and respond accordingly [34].

Typically, the internal structure of a Smart Card microcontroller is composed by the Central Processing Unit (CPU), the Random-Access Memory (RAM), the Read-Only Memory (ROM) and the Electrically-Erasable Programmable Read-Only Memory (EEPROM). The CPU can be an 8-bit, 16-bit or 32-bit processor, and the clock speeds up to 5MHz. If the card clock is multiplier by 2, 4 or 8, the speed up can reach 40MHz. Some Smart Cards are design to have a built-in coprocessor Numeric Coprocessor (NPU), for processing cryptography operations [5]. In terms of memory, the ROM is where the operating system is placed, while some operating system components and the running applications among with their persistent associated data are stored in the EEPROM. The RAM is used as working memory to hold data during computation, likewise it happens in computers. The ROM capacity of the Smart Card microcontrollers typically ranges from 16 KB to 400 KB, while the EEPROM ranges from 1 to 500 KB, and the RAM from 256 Bytes to 16 KB [34].

2.2.3 Communication

The communication with Smart Cards is defined in standards, in order to be assured the interoperability between different Smart Cards, and the terminals. The specifications for the communications with the contact Smart Cards are specified in the ISO/IEC 7816, while for the contactless Smart Cards the communication is defined by the ISO/IEC 14443 standard. Typically, in the contact Smart Cards, the communication is done over a half-duplex bit-serial link. To prevent collisions, the communication follows the master and slave principle, in which the Smart Card is the slave, while the terminal plays the master. The communication starts always on the master side. In the beginning of the communication, the terminal sends a reset signal, for which the card responds with an Answer To Reset (ATR), optionally followed by the protocol parameter selection. After the initialization, all the communication is based on the responses of the Smart Card to the commands sent by the terminal. The communication procedure for contactless Smart Cards is similar, however deal with more difficult conditions given the increased likelihood of interference, and data loss, in wireless communications [34].

The transport layer incorporates several protocols to transfer the Transport Protocol Data Unit (TPDU). The most used transport layer protocols in the contact Smart Cards are the T=0 (byte-oriented), and T=1 (block-oriented) protocols. The USB protocol, is widely used in the PC environment, and can achieve great improvements in the rate of communications, from the 115 kbps in T = 0 and T = 1, to 1.5Mbps in low-speed USB, or 12Mbps in full-speed USB. The most widely used transport protocols in contactless Smart Cards are the ISO/IEC 14443 Type A, and Type B. Both protocols are half-duplex, and strongly based on T=1. The main differences between them are based on modulation, coding and protocol initialization [34]. The communication with the contact Smart Card is done through a Card Acceptance Device (CAD). The CAD can be either a reader or a terminal. A reader is considered a slot, where the Smart Card is entered, able to perform the communication between the Smart Card and the host side by serial, parallel, or even through a USB port of a computer. On the other hand, a terminal integrates a Smart Card reader as one of its components, but also have the ability to process data [5]. The communication with the Smart Cards contactless are over the air, however they usually have a maximum distance of 10 cm from a terminal. Derived from the limitations on communication distance, they are considered proximity cards, and operate on the principle of inductive coupling via an Radio Frequency (RF) magnetic field with a frequency of 13.56 MHz, that is generated by the terminal or PCD (Proximity Coupling Device) [34].

The communication at the application layer is defined in the ISO/IEC 7816-4. The commands and responses exchanged between the Smart Card and the host side are encapsulated over Application Protocol Data Unit (APDU) messages. The APDU commands have their structure defined by a mandatory header followed by an optional body, while the APDU responses are constituted by an optional body followed by a mandatory trailer. The fields of the mandatory command header are CLA, INS, P1 and P2, which correspond respectively to the class of the instruction, the specification of the instruction to call in the context of the previously selected application, and two input parameters. The optional body of the command is composed by Lc, data field, and Le, in which the Lc corresponds to the data field length, while the Le corresponds to the length expected in the response. In the optional body of the response comes the data field, while the mandatory trailer response is composed by SW1 and SW2, representing the status word reporting the success of the execution, or a specific code with the meaning of the error occurred [5, 34].

2.2.4 Operating Systems

The operating system running on the Smart Card is an important requirement, determining the behavior of the card by providing a dedicated Application Programming Interface (API) specific for the chip; therefore, considering the hardware that is below it. The API determines the usability of the card by providing higher level functions used by the applications satisfying the user’s needs. However, some Smart Cards operating systems are not robust and flexible, making life difficult for the application developers. Due to this fact, some platforms and specifications have been created, providing an abstraction layer between the Smart Card native system, and the applications, through an common and useful API [26].

Likewise computer operating systems, the Smart Card operating systems have to provide the resource sharing management, and an environment in which the applications may run [26]. Therefore, the main principles of a Smart Card operating system are to provide the control of the communications between the Smart Card and the host side, ensure proper and effective execution of the received commands, manage the file system, manage and execute the program code of the applications previously installed, and handle all encryption operations [35].

The structure of the Smart Card file system is defined in the ISO/IEC 7816-4. It is based on a tree structure, with a root directory called Master File (MF). Each file system can only have one MF. From the MF derive various Dedicated File (DF) and/or Elementary File (EF), corresponding respectively to directories and binary files to store data [34]. Each DF can have DF or EF inside. The standard for file system names in Smart Cards requires that each EF, DF, or MF has a File Identifier (FID) of 2 Bytes. Each DF has a supplementary name, in addition to its FID, with a length between 1-16 Bytes, including an Application Identifier (AID) between 5-16 Bytes. The AID consists of the combination of the Registered Application Provider Identifier (RID) with the Proprietary Application Identifier Extension (PIX). Each EF has an additional Short File Identifier (SFI) with 5 bit in addition to the FID [34].

The two most relevant standards, to base the development of the Smart Cards operating systems are the ITSE EN 726-3, and the ISO/IEC 7816-4 [35]. Smart Card operating systems can be classified, according to the methods used to download and execute applications, in native operating systems, or interpreter-based operating systems [34]. The native Smart Card operating systems are able to execute applications written in the native machine language of the microprocessor. These systems do not need to use an interpreter or compiler to run the application code, therefore achieving a faster response time, once the applications run at the full speed supported by the processor. However, this gain in computational efficiency costs an increase of complexity, and raises the extension of code [35].

The interpreter-based Smart Card operating systems incorporate an interpreter, responsible for translate the applications into the machine language of the processor [34]. This interpreter allows developers to program in high-level programming languages, abstracting memory management concerns. However, these advantages come at the cost of performance, since the executed code does not run directly on top of the card microprocessor, being the overall execution time worsened relative to the performance of the interpreter on the card [26]. The implementation of a full compiler, and interpreter, inside a Smart Card requires a lot of memory, reason why others implementations are also considered. On implementation consists on the interpretation of a pseudo code similar to the machine code of the microprocessor. Nevertheless, others solutions can split the interpreter in two modules, one executed by the Smart Card, and the other by the host side. These modules split the complexity and the effort of the system, between the Smart Card and the host side, where the computation capacity is much bigger [5].

As mentioned earlier, the core code of the Smart Card operating system is placed on the ROM memory. In fact, there are some operating systems that keep the application code in the ROM memory too. This type of Smart Card operating systems are typically design to be single application oriented. In this sense, the application installed during the issue of the Smart Card, define exclusively its purpose [26]. However, the need to maintain a dynamic and updated system by the companies, led to the need for creation of smart cards capable of withstand multiple applications, and manage its installation and life cycle after the card has been issued [43].

There are different implementations of Smart Card operating systems. These operating systems have different goals, achieving different levels of complexity and security. Besides the developed Smart Card operating systems, also platforms to run on top of them were developed, in order to achieve portability. Therefore, the developed platforms were intended to execute applications running on a Smart Card produced by a specific manufacturer, in smart cards from other manufacturers [26]. A brief resume of some operating systems and platforms are described next.

The **MPCOS** (Gemplus)[1] is a multi-application payment operating systems, for Smart Cards [28]. This Smart Card system was not developed to independently provide an installation process of applications, reason why it is considered a native system [26]. However, their compatibility with the Java Card platform gives to developers the possibility to create and install their own applications after the card has been issued [28].

The **STARCOS** (Giesecke Devrient)[2] Smart Card Chip Operating System, was a native and monolithic operating system.[26] However, the new versions are developed to support several applications in the card, introducing the functionality of multi-application. They also support several hierarchical file structures, and various access controls [15]. Therefore, it is now possible to install more than one application in the EEPROM of a single STARCOS card. The STARCOS provides a Toolkit composed by tools and libraries (API) useful in the development of the applications. However, the definition of the security level of an application is a concern that is left to the designer. This operating system support also Java Card technology, with all inherent advantages of this platform [15].

MultOS (Multiple Operating System)[3] is an operating system for smart cards that provides the highest level of security. It has an Information Technology Security Evaluation Criteria (ITSEC) E6 certification, which corresponds to an Evaluation Assurance Level (EAL)7 from Common Criteria (CC). As the name suggests is a multi-application operating system, and is capable of handling with the deployment of applications after the card has been issued [26]. This system provides, beyond the features of a normal operating system, an API for application developers. Therefore, the installed applications run on top of the MultOS Virtual Machine. The development of applications has to be performed using a specific programming language, MEL (MULTOS Enabling

[1] http://www.gemalto.com/
[2] http://www.gdai.com/
[3] http://www.multos.com/

Language). MEL is a Smart Card - optimized language for MultOS systems, and is basically composed by assembly instructions and primitives [38]. To provide more flexibility and acceptance in the programmer's world, were created tools and platforms to compile C, Java, and Visual Basic applications into the MEL language. This improvement makes these languages compatible with MultOS, until a certain level [14]. The MultOS includes the specification and support for the download and deletion of applications, ensuring a certificated and secure process. Therefore, the applications have to be certificated by the MultOS CA before being installed in the MultOS system [18, 26].

Java Card technology[4] is an open source Java Card platform, able to be loaded into a wide range of Smart Cards from different manufactures. Unlike the systems previously discussed, the Java Card is not an operating system but a platform [26]. This platform allows programmers to develop applets in a constrained Java code language, due to the hardware limitations of the Smart Cards. Therefore, this language is simpler than the original Java, although keeping a subset of the popular Java language API [38]. The Java Card is a security and reliable platform, which makes it one of the most used technologies. This platform is embedded on top of the Smart Card operating systems, and is composed by the Java Card Runtime Environment providing an abstraction layer to the applications. The applets run on top of the Java Card Virtual Machine [5]. The Java Card platform is detailed in section 2.2.5.

GlobalPlatform[5] defines a set of specifications for multi-application Smart Card systems [26]. It also provides specific configurations and supporting documents for market and application purposes. The GlobalPlatform is agnostic regarding the underneath Smart Card architecture. The purpose of GlobalPlatform is to provide security, interoperability, flexibility, multiple suppliers, independence of technology vendors, and future-proofing. Due to this goals, the GlobalPlatformm provides an open and interoperable infrastructure for Smart Cards, devices, and systems [26]. Some Java Card implementations rely on the GlobalPlatform specifications for the secure management of applications creation, deployment and deletion.

Windows for Smart Card (WfSC)[6] was created by Microsoft to directly compete with the multi-applications operating systems available. For some sources like, [26], this system almost disappeared. One justification is based on the strong, secure, and efficient performance of the current available implementations, and standards. Although the project was never abandoned, and likewise it may disappear, it can become a success. Their design aims to achieve low resources cost, and to extend the pc environment into the Smart Card. Microsoft claims that WfSC

[4]http://www.oracle.com/
[5]http://www.globalplatform.org
[6]http://www.microsoft.com

will be less expensive (by a factor of 2 to 3) than either Java Card, or MULTOS [18]. WfSC is a customizable Smart Card Operating System (SCOS). Therefore, the decision to add specific components like runtime environment to run applications being installed, the EMV functionality, or cryptographic functions is left to the Smart Cards provider [18]. Apart from implementing the operating system features, this system also provides a high-level API for the applications. Therefore, this system allows for the development of applications in Visual Basic (VB) or C++, and ensures the execution of the compiled byte code in its on-card virtual machine [38].

Smart-Card.NET [7] is a multi-application platform on top of a specific operating system. Since this platform inherits the advantages of the .NET framework, the developers are allowed to develop applications in multiple programming languages like Charp, C++, VB, JCharp, or JavaScript [26]. The .NET platform is composed by an on-card and an off-card module. On the card is placed the .NET Operating System, and the Card Module Assembly, while out of card are placed the credentials Graphical User Interface (GUI) for the user authentication, and the mini-driver ".dll". The applications also run on top of a virtual machine, offering resources management and security, as well as a garbage collection mechanism [26].

BasicCard [8] is developed and controlled by ZeitControl. This operating system has a virtual machine responsible to run the applications installed after the card has been issued. These applications have to be compiled into a byte-code called P-Code [18]. The programmers have to develop their own applications in the ZeitControl Basic language. Furthermore, they can use the Windows-based development kit to install and download their applications. In the development of applications for this system it is not needed to be aware of the complexity to deal with APDU commands, once the platform provides an abstraction layer to these mechanisms. The platform also enables the use of the cryptographic functions, in order to introduce security and consistency to the developers applications [26]. Even with the development kit, this system is not really a multi-application card, since is just possible to install one application. BasicCard has three states, LOAD, TEST, and RUN. If it is currently in the RUN state, there are no more possibility to exchange or update applications [18].

2.2.5 Java Card

Today, Java Card is one of the most supported and respected Smart Card platforms in the world, when productivity and security are the main requirements [10]. Java Card is a Smart Card platform, on which applications written in a constrained Java programming language run. This platform provides the separation between the Smart Card system, and the applications, by implementing a Runtime Environment. The Runtime Environment supports the Smart Card

[7] http://www.gemalto.com
[8] http://www.zeitcontrol.de/

memory, communication, security, and application execution model [5]. This section contains an overview about this platform architecture, and presents its components.

2.2.5.A Platform Structure

The Java Card Runtime Environment (JCRE) runs on top of the Smart Card hardware, and native system. This module is intended to abstract the operating systems of the card, by implementing a common API layer. The abstraction of the native methods makes possible the interoperability amount various Smart Cards devices, and systems. The JCRE can be described as an on-card system module, and divided in three main layers. The first, and lower one, is composed by the Java Card Virtual Machine (JCVM), and the native system methods. The second corresponds to the system classes, and are composed by applets and transmission management, I/O network communication, applets installer, among others services. The highest layer, groups the installer, the framework classes (API), and industry specifications libraries [5].

The JCVM is split into two modules. One runs inside the card, the interpreter, and the other outside the card, the converter. With this separation is possible to remove from the card the effort of class loading, byte code verification, resolution and linking, and instructions optimization [5]. The converter is, thus, responsible for loading and processing the class files from a package, creating the Converted Applet (CAP) file, as well as creating the export file with the applet public API. On the other hand, the interpreter is responsible for the execution of the CAP file in the Smart Card environment [5].

The installer is a component responsible for receiving safety the CAP file of an applet, and proceeds with their installation in the Smart Card. The framework classes are packages, providing the API for the development of Smart Card applets. The industry specifications are a set of extra libraries, providing additional features directed towards to the purpose of the specific needs of an industry. JCRE system classes are the core of the operating system abstraction, and are responsible for managing applets, transactions, communications and other services. This classes, normally invoke directly native methods of the Smart Card [5].

Due to the resource constraints of Smart Cards, the implementation of the Java platform on these electronic devices is not equal to that used in normal computers. This implementation incorporates only a subset of features of the Java language, and some additional mechanisms that provide specific support for the Java applets inside the Smart Card [34]. These extra features, implemented by the Java Card Runtime Environment, consists on the definition of persistent and transient objects, the specification of atomic operations and transactions, the creation of the applet firewall, and the introduction of sharing mechanisms [5].

The objects can be persistent in memory, or be clean across CAD sections. In fact the space allocated to an object is reserved while it is referenced, so an applet should create just one transient object of each type during its lifetime, and save the object reference in a persistent field.

By default, all the objects instantiated are persistent. The atomicity in Java Cards ensures that any update to a single field, in a persistent object or class, is atomic. Java Card technology also supports a transactional model, in with a set of updates can be done inside a transaction. In this case, each and all of the updates are successfully executed, or all the fields return to their previous states [5].

The Java Card platform is a multi-application environment, so, the applet firewall divide into separate and protected spaces the contexts of each applet. Because of this mechanism, Java Card applets are executed in a sandbox, like applets in a web browser. To allow applets to share data across contexts, the Java Card technology has some shared mechanisms like privileges, entry point objects, global arrays, and shareable interfaces [5].

With the restriction policy imposed by the applet firewall, applets out of the context of other applets cannot have access to their methods or attributes, even if they are public. This security measure is intended to prevent attacks to the applets. However, in some cases it is useful for an applet to share its data and methods with applets from other contexts. One of the mechanisms available in Java Card technology to enable this exchange, overcoming the restrictions of the applet firewall, is the Shareable Interface. The object of a class implementing the shareable interface is called Shareable Interface Object (SIO). The applets wanting to call at least one of the methods of a SIO have to declare an object with the type of its interface, and get a reference to it, on the JCRE. Whenever an applet calls a method from a SIO, the JCRE switches the context of the applet to the context of the SIO. The result is then returned to the caller applet, after its context becomes activate again. With this mechanism, applets out of the context of the SIO are allowed to access the methods defined in its shareable interface.

During a CAD section, the JCRE has a typical Smart Card behavior with the host. The Java Cards are constantly waiting for APDU commands, notifying a specific applet to execute a specific instruction on the card. When a command arrives to the card, the applet is selected, to run the specific instruction called. After the applet has executed of the received command, the card sends the APDU response to the host [5]. The JCRE supports the communication over the T=0, T=1 and contactless transport protocols. It also supports the APDU format defined in the ISO/IEC 7816-4 standard [18].

Some limitations of the Java card version 2.2 were overcome in the new version 3.0. The most important implemented updates consist on the extension of almost all Java data types, except float and double, on the possibility for the use of multiple threads, on the extension of the API support with *java.lang* and *java.util* packages, on the direct handle of class files with all loading and linking process made inside the card, on the all new Java language syntax constructs such as *enums* and *generics*, on the enhanced for loops and auto boxing/unboxing, and on the introduction of the automatic garbage collection mechanism [31].

With all these new features, the Java Card technology can now handle with three application models: the classic applets from the previews 2.2 version, the extended applets that can proofed from all the new API, and the servlet applications using HyperText Transfer Protocol (HTTP) or HyperText Transfer Protocol Secure (HTTPS) [31]. Despite all this new advantages in the use of the updates available in the 3.0 version of the Java card, this new system has to run on a much more powerful and expensive Smart Card. Since that all security and internal structure are the same that on the 2.2 version, the use of version 3.0 has to be justified by the target application requirements.

2.2.5.B Applets

The development of an applet requires a set of conventions rules and steps, allowing the applets to run properly on the JCRE. In first place, it is necessary to carefully design the requirements of the system. After that, comes the implementation part with the decision of the technology to use. The implementation part is followed by the important phase of tests to the requirements and security of the system. The Java Card system security depends on the security of the applet, and on the security of the Smart Card platform. Therefore, in all development phases, the security of the system has to be an important issue [34].

Each applet is a persistent object in the Java Card. Therefore, unless the Java Card makes possible the deletion of applets, after installed the applet lives out the entire lifetime of the card [5]. In fact, the new version of the Java Card technology, 3.0, ensures the possibility for the deletion of applets, although in older versions be optional. Any applet is identified by its AID (see 2.2.4), defined in the registration step. All the applets have to extend from the *javacard.framework.Applet* class, to implement some of the crucial methods, such as the *Install*, the *Register*, the *Select*, the *Process*, and the *Deselect* methods [5].

The life-cycle of an applet starts with its installation and registration on the card. After that, each applet switches is state between inactive and active. This happens, at least in version 2.2, because the JCRE is an single thread environment, which means that at each moment just one applet can be active while the others stays inactive [5]. In the installation process, an instance of the applet is created, so, each applet must implement in their constructor all the necessary initializations. Due to the fact that not all implementations of the Java Card have a garbage collection available, it is useful to instantiate each application just one time in the life time if the card, in order to conserve resources. The register method should be the last instruction called by the installation method, once it allows the applet to be selected, and set to run, by the JCRE [5].

In a communication with a Java Card, the first APDU command has to be a *SELECT* command. Whenever the JCRE receives a *SELECT* command, it selects the specified applet in the command. Therefore, all the APDU commands following received by the JCRE are redirect internally to the previously selected applet. When a new *SELECT* command is received, the current selected applet is deselected, and the new one owns the context, by being selected. The select and deselect methods make an applet exchange from an active to an inactive state, and vice versa. When an applet is successfully selected, its textitprocess() method is executed, where the instructions and arguments are passed by the JCRE [5].

The development and installation of an applet implies a strong test process. Likewise in the normal Java programs, the developed Java Card source code has to be compiled in to class files. However, to be loaded into the Java Card, these class files have now to be compiled into a CAP file. Before the creation of the CAP file, the class files should be tested in a JCRE simulator. This simulator runs the applets on a normal Java Virtual Machine. This mechanism is a quickly source for testing the functionality of the developed code, but at same time a not efficient error reporter, once some runtime features of a real Java Card are not being tested. After this process, the applet can be converted in a CAP file, and tested in a Java Card emulator. The applet runs in a Java Card Virtual Machine emulator, in order to be tested before being installed in a real device. Therefore, the applet is soaked in a real environment of an Java Card, without being on a real Java Card. [5]. Only after all this tests, the application becomes ready to be loaded, and installed, into a real Java Card.

There are tools available to assist the programmers in the creation and development of applets for Java Cards. *Sun* has created the Java Card Development Kit (JCDK). This toolkit allows the programmers to write Java Card applets, and test them, without a physical Java Card, neither a Smart Card reader. The JCDK is composed by the simulator Java Card Workstation Development Environment (JCWDE), a Java Card converter tool, a Java Card verifier, an APDU tool, a CAP dump tool, a script generation tool, and some supporting extra libraries [27, 31, 40].

The simulator JCWDE can be integrated with a debugger and Integrated Development Environment (IDE), and it does not support some features of a real Java Card, such as the package installation, the applet instance creation, the firewalls, and the transactions. The C language Java Card Runtime Environment (C-JCRE) is a C programming language implementation of the Java Card API, JCVM, and JCRE, providing real code test conditions to the programmers. The Java Card converter tool is intended to generating the CAP files, while the Java Card verifier is intended to check the validity of CAP and export files. The APDU tool is used to send and receive APDU during tests, while the CAP dump tool is used to show the contents of CAP and EXP files. The script generation tool is the off-card installer, and basically converts CAP files into script files composed by APDU commands to be loaded to the Java Card. The supporting extra libraries are class and export files of the Java Card API implementation [31].

2.2.5.C Middleware

The development of an applet entails the development of some Middleware running on the host side, and which encapsulates the complexity of the communication with the Java Card, and with the Smart Card reader. The Middleware should also be robust and interoperable, in order to provide platform independence, and a useful API to the highest level applications. The software usually resides on a terminal such as a workstation, a Point Of Sale (POS), a mobile phone, or a host application [31].

There are several ways of implementing a Middleware. In order to develop an applet with the purpose of remote authentication, one of the best interfaces to provide is the Public-Key Cryptography Standards (PKCS#11). The PKCS#11: Cryptographic Token Interface Standard specifies an API implemented by devices holding cryptographic information, and performing cryptographic operations. The main purpose of the PKCS#11 standard interface is to encapsulate the complexity of the communication with the Java Card, and with the applet. This allows the implementation of the same standard methods, by different vendors in a wide variety of devices. The user applications use the Middleware providing this standard interfaces, having a common and logical view of a device, called cryptographic token.

The Middleware is usually developed using one of the follow API: the Java Card Remote Method Invocation (RMI) [31], the OpenCard Framework (OCF) API, the Personal Computer/Smart Card (PC/SC) or the Movement for the Use of Smart Cards in Linux Environment (MUSCLE). Currently, the more used ones are the OCF and PC/SC, for windows environment. The goal of all this libraries is to introduce high level API to encapsulate low level communication between the Java Card and the smart Card reader device [18].

PC/SC is an industry standard implemented by Microsoft defining an API for communication with Smart Cards. A wide number of the available device readers for PC already support these libraries. The central component of its architecture is the ICC Resource Manager, responsible for controlling all the interface devices and service providers. The OCF and PC/SC addresses similar concerns by providing similar functionalities. However, the main goal of PC/SC is to provide access for the cryptographic functionality in Smart Cards [18].

The OCF has appeared after the PC/SC from Microsoft, and took advantage of some already available features. This two technologies try to define a standard way to integrate Smart Cards with computers and terminal systems [13]. The main differences between these two approaches are based on the fact that OCF takes into account the access concurrency, the support for multiple applications, the portability, and the card issuer's transparency [18]. The OCF is composed by set of Java-based API, establishing a common communication base for different Smart Card readers, of different vendors. Furthermore, given that it is an open source library is easy to find the support documentation and code libraries. However, the OCF has no continuation since 2004, once the development team gave the project as finished. Therefore, despite being a very used technology and being stable, it has not updates since that time.

MUSCLE is an implementation of PC/SC for Linux. The PC/SC resource manager of the MUSCLE allows multiple-application to get access to the card, however just one at the time [18].

The Java Card RMI Client API uses a card-terminal and libraries such as the OCF, and has been only available after the Java Card Version 3.0. This technology allows the host application to get access to some of the applet's methods [31]. The Java Card Remote Method Invocation (JCRMI) provides an abstraction layer, in order to avoid the low-level communication using the APDU package. The stubs generated to the client, provide a way to interact with the remote object. In order to develop the interface for the JCRMI, can be used Java Modeling Language (JML). This language specify the remote interface using model fields, and serve as a contract to the communication [44]. The JCRMI client API uses the OCF for card management, and to assure the communications.

2.2.5.D Biometric Support

As seen previously in the section 2.1.4, a biometric authentication system is a secure and reliable mechanism to identify a person. Considering the safety and convenience of Smart Cards becomes interesting to join these two technologies, in order to develop a system with the advantages of both. A Smart Card, being a secure and reliable device to store private and sensitive data, can be seen as a platform for the secure storage of the biometric template. In order to protect the stored biometric template of the user, the Smart Card also implements the match on card operation. Therefore, no external entity can have access to the user biometric template. When a biometric authentication has to be performed, is the Smart Card executing the match of the user biometric data, captured in the authentication phase, against his biometric template, stored on the Smart Card during the enrollment process [9].

With the match on card module, the Smart Cards are able to perform local biometric authentications of the user. The biometric authentication involves an inicial enrolment process of the user biometric template. In systems using Smart Cards, the enrollment phase of the biometric template is stored in the Smart Card, instead of being used a central data base. Thus, when a user wants to perform an authentication before the system, the biometric data is not sent via the Intrnet. Typically, after the successful match inside the Smart Card a cryptographic key is released, and used to perform the remote authentication using cryptographic operations over a challenge sent by the remote server. Usually, in cryptosystems, the biometric authentication is performed locally, and is used as a key release mechanism. The cryptographic key is also stored inside the Smart Card as part of the users private data, and usually cannot be accessed by any external entity [33].

The existent matches on card modules are proprietary, once the mathematic algorithms supporting these solutions are also proprietary. This happens once the biometric technology is not yet very mature, and only relatively recently arises to the world of smart cards. In order to create interoperability in the use of biometrics at the Java Card applet level, the Java Card Forum [9] has proposed the Java Card Biometric Application Programming Interface (JC-BioAPI) [9]. Therefore, the applets inside the Java Card can use any match algorithm provided by any entity, without

[9] http://www.javacardforum.org/

having access to proprietary details, since it respects the JC-BioAPI. The Java Card technology already ensures the interoperability between Smart Card operating systems, thus, this API can be used in a wide range of different cards. This API supports the enrolment phase of the biometric template into the Java Card, as well as a secure biometric validation, avoiding that sensitive data to be exposed outside of the card. Besides this, the JC-BioAPI also supports multiple biometrics on a single card [9].

The JC-BioAPI is design to use limited amounts of the Java Card memory, and as few computational cycles as possible. This API provides the necessary abstraction for the development of the Java Card applets, and biometric algorithms, independently from each other. Therefore, the architecture of the JC-BioAPI consists on the server and client applets. The server applet is intended to manage the enrolment and configuration of the biometric on the Java Card, as well as to provide the shareable interfaces to the client applets. The client applets use the services of the server applets. Therefore, the client applets have to contact the server applet in order to obtain access to the biometric operations performed to the biometric template [9]. When the server applet is initialized, it creates an instance of a *BioTemplate*, that is responsible for providing the interfaces to the clients, and to the server. Once the match on card module is developed in the native system of the card, only this library can get access to the biometric template of the user [9].

Besides the use of the biometric template by the card there are others approaches such as the template on card. In the template of card approach, the template is release every time the authentication is request, in opposition to the match on card approach where the template never leaves the card [2]. The match on card approach can specifically protect the system against substitution or modification of the matcher, since nobody can get access to the biometric template of the user. Besides this, it protects against the tampering of templates, and once the verification decision is internally used by the card, the match result cannot be intercepted, or replaced from an outside entity [2].

2.3 Existent Solutions

This section presents authentication solutions currently implemented in the market and exploring the same or a similar concept than the one used by the proposed solution. Therefore, concrete solutions performing user remote authentication are explored, describing their benefits, and the main advantages that the proposed solution has compared to them.

The authentication evolution was described previously, where the clear preference for the use of Smart Cards and the tendency of most current systems to the use of biometrics can be noticed. Although some of the currently most used remote authentication solutions do not use the user biometrics, even though there is some reluctance to rely on the use of this technology in remote systems, some solutions already do it. This section describes two solutions currently used to perform user remote authentication, one using the user biometrics and the other only based on the user PIN. The presented solutions are the Portuguese citizen card, and a product implemented by *Gemalto*[10] that arose during the developed of this thesis.

The Portuguese citizen card is a solution that exists since 2007, used as the official ID document for Portuguese citizens. This Electronic Identity (eID) document incorporates a Smart Card enabling the users to communicate with their government administrations such as the Ministry of Interior, Finance, Health and Justice, as well as to perform remote authentications in common systems supporting this mechanism. This identification document is based on Identification, Authentication, and Signature (IAS) specifications, to among other things perform the user remote authentication, and is deployed in a PKI system. To identify and authenticate themselves, cardholders enter a secret pin code and the card then generates a digital signature to sign the challenge sent by an authenticator entity.This electronic document also includes biometric features, being able to store the user fingerprint biometric template, and perform his local biometric authentication [8].

The Smart Card solution was developed by *Gemalto*, while the Middleware was developed by *Zetes Burótica*[11] [8]. Although this solution is a reliable source to authenticate the users, it does not use the biometric features during the remote authentications. Therefore, this system cannot ensure that the person performing the authentication is really the user being authenticated, since the PIN authentication cannot ensure the presence of the user during the authentication. Non-repudiation is also susceptible of being questioned since the PIN of the user can be stolen or borrowed. The proposed solution on this thesis intends to enable solutions, such as the Portuguese citizen card, to take advantage of the biometric technology inside, adding the biometric authentication feature securely to the remote authentication.

The *Gemalto .Net Bio* solution was placed in the market during the period of development of this thesis.It is an innovative software solution that provides fingerprint biometric support for Gemalto .NET Smart Cards, and is integrated with Microsoft Windows XP, Vista and Windows 7. This solution enables fingerprint user authentication as an alternative or complement to Smart Card PIN verification. In short, it performs the requirements of the system proposed in this thesis. It also gives access to the digital certificates on the card, performs digital signatures, can be used to encrypt files and even to allow secure VPN access. This solution is compatible with the Smart Card security components of Windows and with the vast majority of fingerprint sensors.

However, this solution is exclusive for Windows environments, and .Net smart Card platforms. The main advantage of the proposed solution in this thesis relies on the fact that the proposed solution aims to be implemented in current existent solutions containing biometric authentication technology, but not using it in remote scenarios. Therefore, the proposed solution is generic and can be implemented in a wide range of existent solutions, improving their level of security and

[10] http://www.gemalto.com
[11] http://www.zetes.pt/

efficiency.

2.4 Conclusion

This section describes a brief summary of the presented state of the art. An overview of techniques and technologies supporting the development of the proposed solution on this thesis is presented.

The state of the art presents the motivation for the development of remote authentication mechanisms, given the growing need to protect online services and information access, from malicious users. This chapter presents the evolution of the authentication mechanisms during the development of the Internet, demonstrating the considerable adhesion to the use of Smart Cards as secure, and indispensable devices to create a reliable authentication process. In addition it is evident the gradual adoption of biometrics in the recent authentication mechanisms.

Nowadays a wide range of authentication systems based on something that the user knows, such as their passwords, still exist. This type of security can easily be transmitted to a friend, lost, forged, or stolen. Therefore, password systems cannot proof on the on-site presence of the user being authenticated. A biometric authentication system, on the other hand, is based on a person's physiological and behavioral characteristics, being intrinsically related to a specific user. The benefits of biometric authentication systems make this approach a good replacement for the PIN-based authentication systems, in order to provide a more reliable and secure authentication mechanism [42].

The substitution of password based authentication systems for biometric ones can be supported by the use of Smart Cards. The proposed solution suggests the deployment of the Smart Card in a PKI system, in order to enable the non-dependence on passwords.

Smart cards are an indispensable component in a reliable authentication system using biometrics. They are small, portable, and secure devices, capable of offering the protection of the user's private information, such as cryptographic private keys and biometric templates. The Java Card platform makes it possible to develop applications able to run on Smart Cards issued by different providers, being a source of interoperability and physical platform independence. Furthermore, they are able to perform critical operations such as digital signatures, and the biometric Match-On-Card (MOC), using the private information of the user. That way they can assure the exclusivity to the user's private data, since it never leaves the Smart Card. The existent solutions do not include all the new features brought by the proposed solution, since they are not able to completely release the user from using passwords. Therefore, the proposed solution is an asset in today's world, contemplating a solution to a remote authentication mechanism that can take advantage of existent technologies and from the development of authentication systems developed to date. The proposed solution is presented in the following chapter.

3
Proposed Solution

Contents

In this chapter, a mechanism to provide secure user remote authentication, using biometric data, is proposed. The presented solution is intended to be generic, and easy to deploy in current solutions.

One of the currently most used remote authentication systems is based on Smart Cards and asymmetric cipher. Smart Cards are simple and trustworthy devices. As stated in the previous chapter, they have storage capacities, an internal secure architecture to prevent the disclosure of private information, and also perform cryptographic operations. The Smart Card system is seen by the remote side as a trusted platform, being the security bases of the authentication process. In fact, the entity being remotely authenticated is the Smart Card itself, since the user private key is unique, and the Smart Card has exclusive and controlled access to it. In these systems, the Smart Card ensures to the remote server that the user performs a local authentication, using a private secret. However, in these systems the actual presence of the user being authenticated is not assured, only the knowledge of the Personal Identification Number (PIN) or password is verified. The secret can also be susceptible to dictionary attacks, or be easily stolen, guessed, or borrowed, by other people.

The biometric authentication can replace the user local authentication using PIN, since it has advantages in convenience and security. The biometric data are specific, unique and not transferable among users. As seen in the previous chapter, this authentication mechanism ensures the presence of the user being authenticated, and eliminates the need for the user to memorize his private secret. Thus, it improves the convenience of the users, as well as making life more difficult for the attackers.

The proposed solution appears as a consequence of the need to improve the existing remote authentication systems. This solution is based on the existence of remote authentication schemes, using Smart Cards providing asymmetric cipher operations, and assuring the user local authentication. However, the local user authentication is based on his biometric data. With this solution it is possible to have a remote user authentication system, using his biometrics, overcoming the main obstacles that currently are blocking the use of biometrics in remote authentication schemes.

To adequately describe the proposed solution, this chapter is divided into 3 sections. Section 3.1 describes the authentication scheme of the proposed solution, presenting the participating entities and the authentication flow. The following section 3.2 presents an overview of the proposed architecture and its components. This section has 2 subsections, describing the architecture and functionality of each component needed inside the Smart Card and the relationship between them, and presenting the impact and importance of the Middleware in the proposed solution, as well as the composing modules. This chapter is concluded by presenting the main advantages of the solution.

3.1 Authentication Scheme

The authentication scheme of the proposed solution is divided in two phases: i) the remote authentication between the Smart Card and the remote server; and ii) the local authentication between the Smart Card and the user being authenticated. Figure 3.1 depicts this authentication scheme.

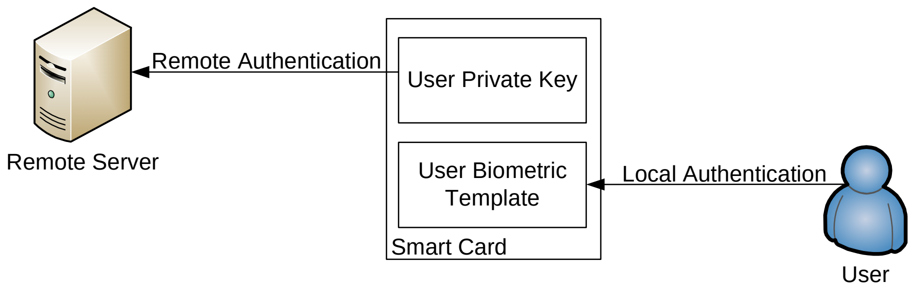


Figure 3.1: Authentication scheme of the proposed solution.

The remote authentication is performed by the Smart Card through asymmetric cipher, using the user private key that only the card knows. The remote server can authenticate the Smart Card using asymmetric cryptography, and trust the Smart Card to locally authenticate the user using the card's biometric Match-On-Card (MOC) operation to validate the biometric signature. This authentication scheme requires each Smart Card to securely store the private key of the user, as well as his biometric template.

For the remote authentication, the user private key is used to sign a challenge sent by the remote server to the Smart Card. The authentication is successfully executed, if the remote server validates that the challenge previously sent, was in fact signed using the private key of the user being authenticated.

The user public key is encapsulated in a public digital certificate sent to the remote server during the authentication process. The digital certificate containing the public key, along with other public information of the user, also contains its own digital signature. The signature of the public certificate is performed with the private key of a trusted Certification Authority (CA). To check the validity of the user public digital certificate, the server uses the public key of the CA.

After receiving the public digital certificate of the user, the server verifies if it actually belongs to the user being authenticated, and, if it does, obtains the user public key from it. After getting the user public key, the server sends the challenge to be signed to the Smart Card. The Smart Card receives this challenge, and signs it with the on chip private key, and sends the result to the remote server. The remote server validates the received signature, to check if the Smart Card properly signed the challenge, and with the correct private key. Figure 3.2 details the main components and interactions necessary for the remote authentication used in the proposed system.

After a successfully remote authentication has been completed, the server is assured that it

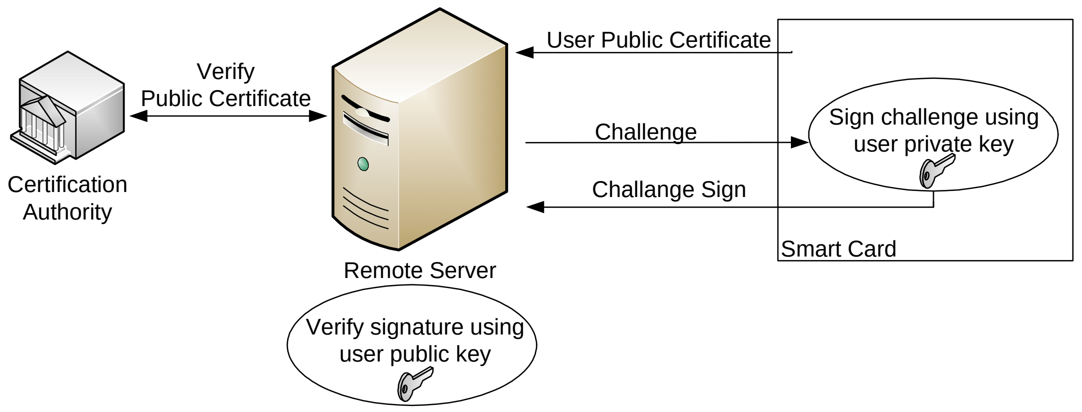


Figure 3.2: Scheme of remote authentication of the proposed solution.

is communicating with the Smart Card of the user being authenticated. However, the described remote authentication scheme is not enough to prove to the remote server, the presence of the user during the authentication process. The presence of the user is assured by the local biometric authentication, performed by the Smart Card.

The Smart Card ensures the presence of the user being authenticated by prompting his biometric signature when using his private key. The scheme of the biometric local authentication used in the proposed system is depicted in Figure 3.3.

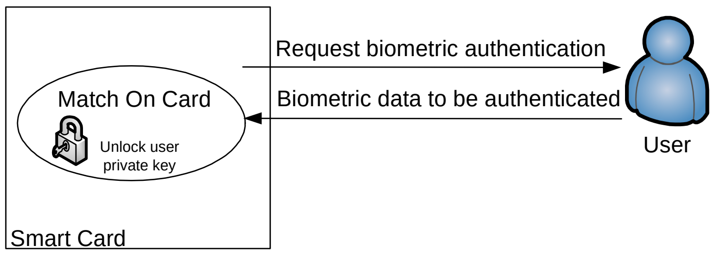

Match On Card
Unlock user
private key
Smart Card
Request biometric authentication
Biometric data to be authenticated
User

Figure 3.3: Scheme of biometric local authentication of the proposed solution.

The local biometric authentication is performed in the Smart Card, comparing his signature with the stored biometric template, using the MOC operation. The server trusts in the authentication system, by trusting in the CA and in the Smart Card. With this scheme, biometric data is never sent over unsecure networks, and the server does not have to trust in insecure biometric matchers.

The next section describes the proposed architecture to implement this authentication scheme, and provides a detailed explanation on each component.

3.2 Architecture

The architecture of the proposed solution depicts the needed structure to implementation the authentication scheme presented in previous section 3.1. The design of this architecture is intended to allow the deployment of the biometric authentication feature in existent systems using the PIN to locally authenticate the user.

Old standards and implemented solutions were designed to perform local authentication of the users only using PIN. New solutions can take advantages of the recent biometric Application Programming Interface (API) and developed standards, to simply embed in its initial architecture, the biometric authentication feature. However, the proposed solution is intended to be deployed in already existent solutions, using PIN. The goal is to create a solution that can easily, and with a low cost, add biometric authentication to these systems.

To assure the authentication scheme previously presented in section 3.1, the proposed solution requires the use of Smart Cards with in a Public Key Infrastructure (PKI), a Middleware to ensure the communication between the Smart Card and the user side applications. Figure 3.4 depicts the layers of the proposed architecture, and the components that compose it.

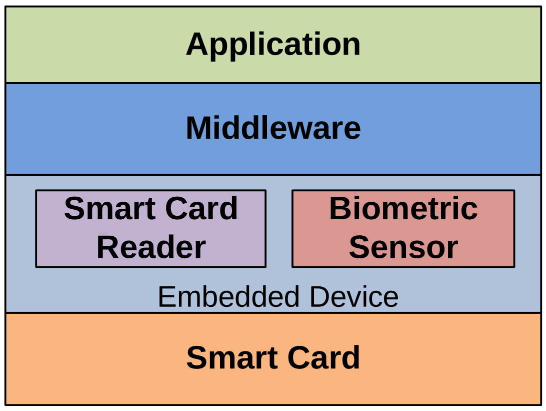


Figure 3.4: Architecture of the proposed solution.

The main requirements of the Smart Card layer consider its capability to store the private key of the user, to internally perform the biometric verification, and to execute the necessary cryptographic operations. The embedded device layer is intended to capture the biometric data of the user, and enable the communication with the Smart Card. The purpose of the Middleware layer is to ensure the communication with the Smart Card reader and the biometric sensor, and to provide an API to be used by the application layer.

In current existing solutions, the Smart Card layer is composed of a Smart Card with an authentication module inside. This authentication module already has the ability to satisfy the requirements of the remote authentication, storing securely the user private key of the user to perform the needed cryptographic operations, ensuring the local authentication of the user with his PIN. Some solutions even have the ability to perform user biometric authentication, using the MOC module. However, the module performing the biometric authentication has no connection with the remote authentication module, only allowing local biometric authentications.

In Smart Cards where modules can be deployed after the card has been issued, the module able to perform local biometric authentication can be easily loaded, independently of the already installed authentication module. In order to exploit the capability of the co-existence of these two modules, the proposed solution suggests the development of a proxy module. This module is capable of managing the remote authentication inside the Java Card, ensuring the biometric local authentication of the user in the MOC module, before the use of the user private key in the authentication module.

The Middleware used in existent implementations already has the capability to satisfy the remote authentication requirements, using the local authentication with PIN. It has the ability to communicate with the Smart Card, and to encapsulate the complexity of the local user authentication process. The Middleware of the proposed solution uses all the functionalities of the existent systems; however it also requires the capability to interact with the biometric sensor. This new feature allows the capture of the user biometric data being matched on the card during the biometric local authentication of the user. The Smart Card reader and the biometric sensor have to be a single embedded device, to ensure that the biometric data sent to the MOC module is captured in the moment of the authentication, and is protected from external entities.

The following sections detail the structure of the Smart Card and Middleware layers.

3.2.1 Smart Card Layer

The Smart Card layer in the proposed architecture, represents the lowest level layer as depicted in the Figure 3.5.

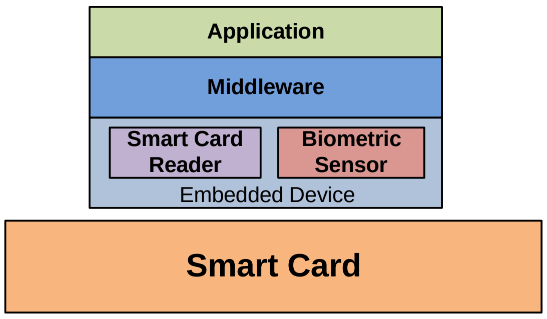


Figure 3.5: Smart Card layer of the proposed solution.

The Smart Card layer is responsible for storing the user information, and to process the critical operations of the authentication. The user information stored in the Smart Card can be public, protected, or private.

The public data can be easily accessed without security requirements, such as the certificate containing the public key of the user, the user name, age, and picture. The access to information such as the user address, or other private information, is protected by a biometric local authentication of the user. The information that is crucial to the correct operation of the authentication system, and that cannot be accessed by any external entity is also kept in the Smart Card. This private and sensitive information is also protected by the biometric local authentication of the user, but it can never leave the card, so it can only be used by operations performed inside it, being this way protected against external attacks. This data is mainly the private key of the user, used in remote authentication, and the biometric template.

In the existent solutions the main operations needed to assure the user authentication are divided in those used for the remote authentication, and those used in the local authentication. The remote authentication requires cryptographic operations such as digital signatures using the user private key. The local authentication requires the ability to perform biometric match operations in order to compare the biometric data of the user, captured in the authentication process, against the private biometric template of the user, stored in the Smart Card during the enrollment process. The structure of the already implemented solutions, performing remote authentication and assuring the user local authentication using the PIN is depicted in Figure 3.6.

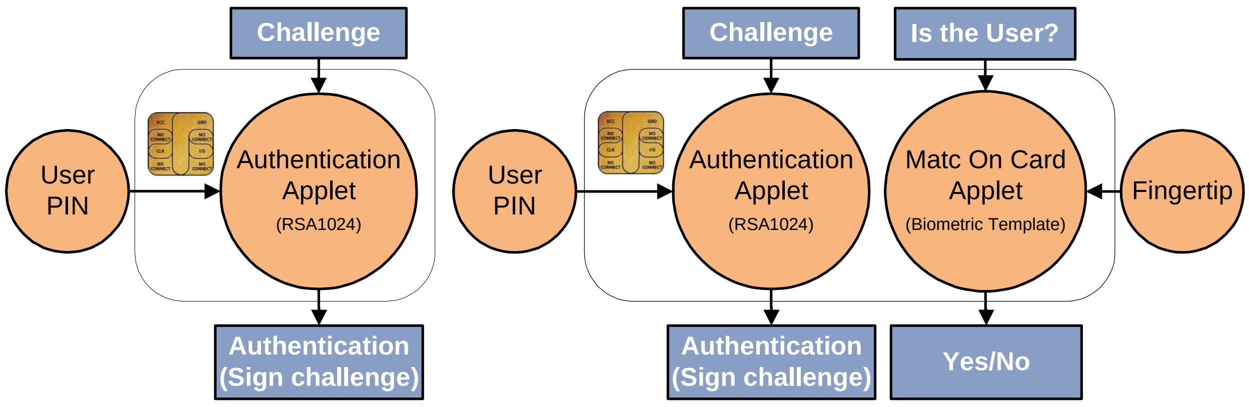

Challenge
User PIN
Authentication Applet (RSA1024)
Authentication (Sign challenge)
Challenge
Is the User?
User PIN
Authentication Applet (RSA1024)
Matc On Card Applet (Biometric Template)
Fingertip
Authentication (Sign challenge)
Yes/No

Figure 3.6: Smart Card architecture of existing solutions.

In the proposed solution, this Smart Card layer performs the remote authentication towards the remote server, and at the same time assures the local biometric authentication of the user. As seen in the Figure 3.6, the existing solutions require an authentication module, responsible for the remote authentication, which stores the data of the user and performs the cryptographic operations. This module uses the user PIN authentication, to unblock the user private key in the remote authentication process, before it signs the challenge sent by the remote server.

Some existent solutions already have the biometric MOC module, capable of executing the user biometric local authentication, by comparing the stored biometric template of the user against the user biometric data. However, this module can only be used in local authentications, when the user can be seen during the authentication process. This ensures that the biometric data being verified is derived from a local biometric capture during the local authentication.

Although these solutions apparently contain all the necessary technology to perform remote authentications using biometrics that does not occurs. The authentication and the biometric MOC modules do not communicate with each other. Thus, they cannot directly take advantage of each other's capabilities. The communication between these two modules does not exist in the current systems, due to security restrictions in the communication of different modules inside a Smart Card. Another reason for this comes from the fact that the older authentication modules were developed during a time where Smart Cards made local authentication solely by using PIN. Therefore, given that biometric standards are more recent; when these older authentication modules were developed they were not prepared to communicate with the biometric modules. Apart from these issues, there are some global security concerns that needed to be ensured by the system, and that are not ensured with the simple aggregation of these two modules, as will be analyzed ahead.

Using the new biometric standards and API for Smart Cards, the newest remote authentication solutions can contain a more evolved authentication module, integrating the ability for the local authentication to be performed using biometrics instead of PIN. The proposed solution, however, is intended to be totally compatible with existent solutions, to take advantage of the existent technologies and their benefits, since they are widely tested and implemented. Thus, the proposed solution aims to create an add-on, easily deployed in existent systems. The proposed solution is designed to be ported into existent remote authentication systems, with low cost and code alteration, adding the biometric local authentication feature to the authentication module already existent inside the Smart Card, by taking advantage of the MOC module.

To provide the ability of the authentication module to interact with the MOC module, the proposed solution requires the addition of a new module, the proxy module. The new structure proposed for the Smart Card layer is depicted in Figure 3.7, which includes the proposed proxy module.

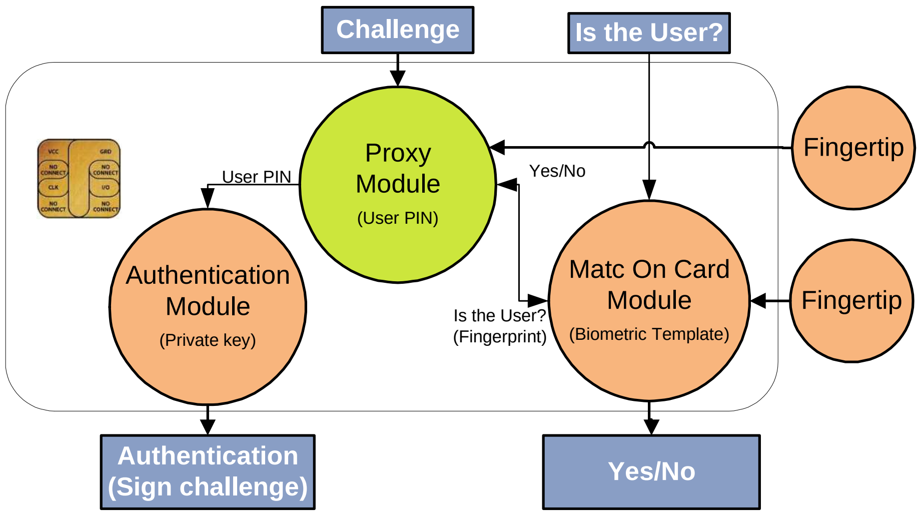


Figure 3.7: Smart Card architecture of the proposed solution.

The proxy module is responsible for assuring the communication between the authentication and MOC modules. The needed alterations to the existing modules are slim to none. The

MOC module has to provide a standard or pre-known biometric API, to be used internally by the other modules inside the Smart Card. Usually this module already comes with such an interface. Identically, the authentication module has to enable the communication with internal modules, by providing a public API to be used by the proxy. The alterations required in the authentication module do not change its internal structure or operating mechanism, since they merely consist on a publication of a public interface.

As shown in the Figure 3.7, the proxy module acts as a bridge between the authentication and MOC modules. This way, the authentication module can use the biometric authentication technology inside the Smart Card, to perform the user local authentication without resorting to a profound change in its structure.

However, if the authentication module keeps the communication channel with the external world outside the card, the migration of the system to ensure the local biometric authentication cannot be guaranteed. The user could keep using directly the authentication module, with his old PIN, which compromises the goal for the assurance of the biometric authentication. Therefore, it is possible to notice in the Figure 3.7, that the direct communication between the host side and the authentication module no longer exists. On another hand, the MOC module can maintain a communication channel with the external world. This happens, since the MOC module requires a biometric data enrollment of the user, and its normal behavior as a local matcher does not compromise the security of the system.

There are two main ways to prevent the direct interaction of the authentication module with the external world outside the card. One way to perform this is to change the status of the authentication module, and turn it into a built-in library on the Smart Card. This way, no entity outside the card can directly communicate with it. The other way, is for the proxy to change the user PIN directly in the authentication module. Thus, after the user PIN changes, only the proxy will know the PIN required by the authentication module to perform the local PIN authentication. If an external entity would try to communicate directly with the authentication module, it will not know the correct user PIN, and after three wrong PIN entries, the authentication module would block.

The proxy module is responsible for managing and controlling the execution flow of the authentication process, inside the Smart Card. With the deployment of the proxy depicted in Figure 3.7, the three modules (authentication, proxy, and MOC), are seen by the outside world of the Smart Card, as an unique module, capable of ensuring the authentication with the remote server, and the user local biometric authentication.

When a challenge is received by the application layer in the user side, it is sent through the Middleware to the Smart Card, over a specific command that it is received by the proxy module. The proxy module checks if it’s a command to be executed by the authentication module, and redirects it internally. The authentication module receives the command as if it was sent directly to it, by the host side, and normally executes the command. If it is the command requesting the signature of the remote challenge, the authentication module first checks if the user is already locally authenticated. If it is, the authentication module processes the digital signature of the remote challenge, using the user private key, and returns the result of the challenge signature to the host side. Otherwise, it throws the requiring user PIN exception, requesting the user local authentication. If this exception was thrown directly to the host side, a user local PIN authentication would take place. However, the proxy module catches this exception and throws the requiring biometric user authentication exception, to the upper layer.

Out of the Smart Card, the Middleware knows that a local biometric authentication is needed. This initializes the user local authentication process, after what the Middleware sends the sign command again. In this local biometric authentication, the proxy module is the front end of the MOC module too, since the biometric authentication process also passes through it. The Middleware has to send the user biometric data to be matched by the MOC module, to the proxy module. The proxy module internally calls the MOC module, sending the biometric data generated by the biometric reader. At this point, the MOC module verifies if this biometric data matches the stored user biometric template.

After a successful user biometric authentication, the proxy module will automatically perform a user PIN authentication, in the authentication module. Thus, when the signature command, to sign the challenge, is called again, the user is already authenticated in the authentication module. The signature command, to sign the challenge sent by the remote server, can now be called again. The proxy sends it to be executed by the authentication module, and the result is sent to the Middleware layer.

3.2.2 Middleware Layer

The user authentication, in a remote server, is usually performed in the context of an application. However, the Smart Card is the layer executing the critical and sensitive required operations to authenticate the user. The application layer does not communicate directly with the Smart Card, due to physical and complexity limitations, reason why the Middleware layer is necessary. Therefore, the Middleware layer is responsible for the outside management of the authentication process, and for providing a standard and easy interaction with the lower layers of the architecture, through a standard API.

As depicted in Figure 3.10, the Middleware is the software component placed in the user side, between the application and the Smart Card reader layers. This layer abstracts all the complexity of the external user authentication management, as well as the communication with the Smart Card reader and the biometric sensor.

The existent solutions, performing remote authentications using Smart Cards with PIN, already have a Middleware layer providing an API for the host side applications. This Middleware encap-

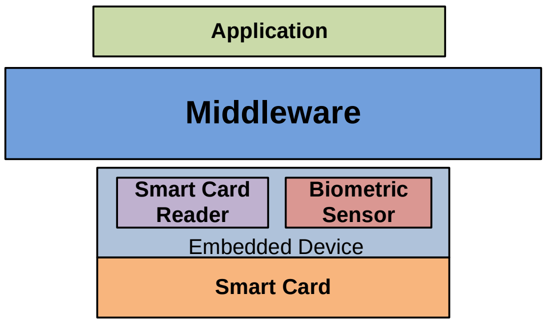


Figure 3.8: Middleware layer of the proposed solution.

sulates the command structure needed to be sent to the Smart Card, as well as the particularities of the Smart Card protocols and command formats. Therefore, a programmer developing a user application can use the standard Middleware API, not needing to worry with the complexity of directly interacting with the Smart Card, and its reader.

In addition to the requirements of the existent solutions, the Middleware of the proposed solution has to be prepared to interact with the biometric sensor, in order to manage the biometric local authentication of the user. The Middleware is the component that coordinates and guides the user in the host side, during the whole authentication process. As depicted in Figure 3.9, the Middleware architecture is divided in three main sections, the application API, the Smart Card reader library, and the biometric sensor library.

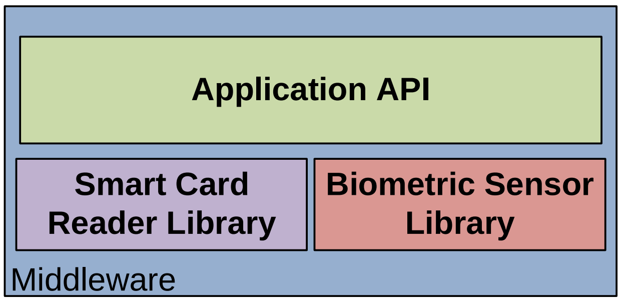


Figure 3.9: Middleware architecture of the proposed solution.

The API module represents the interface provided by the Middleware to the user applications, in the host side. This module can be generic, respecting Smart Card standards and recommendations such as the Public-Key Cryptography Standards (PKCS#11). The Smart Card reader libraries are responsible for the communication with the Smart Card. They encapsulate the complexity of choosing the Smart Card reader, as well as of creating the commands respecting standard formats and the Smart Card module requirements. The biometric sensor libraries are responsible for requesting the user to insert his biometrics in the sensor, as well as to request the sensor to collect the biometric data of the user, and to send them to the Smart Card, where they will be matched.

The requests of the server, during a remote authentication, pass through the Middleware, before reaching the Smart Card. Whenever a request is made by the remote server, the Middleware keeps it, and sends it to the proxy module in the Smart Card. If it is a private operation requiring local user authentication, the proxy requests the user biometric authentication. This authentication process is coordinated by the Middleware layer.

The Middleware prepares the biometric sensor to receive the biometric data of the user, and informs the user to put his biometric trait in the biometric sensor. After the sensor has captured the biometric signature of the user, and generated the data to be compared, it sends it to the proxy module, in order to perform the local biometric MOC. After a successful authentication of the user, making sure that the user being authenticated is authorizing the authentication, the state of the authentication module in the Smart Card is updated. The Middleware re-sends to the Proxy module the previous request, without the need for the user application or the remote server to repeat it. Therefore, the Middleware encapsulates all the process of the user local authentication, requesting the user biometrics and managing the required commands to be sent to the Smart Card.

The architecture layer between the Middleware and the Smart Card is the embedded device reader, as depicted in Figure 3.10. Here is represented one of the most important requirements of the proposed solution, the need for the Smart Card reader and the biometric sensor to be in a single embedded device.

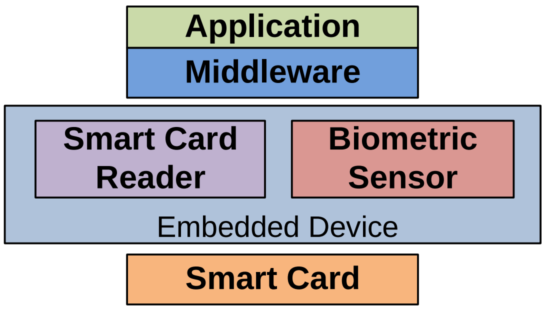


Figure 3.10: Readers layer of the proposed solution.

The purpose of this requirement is to assure a secure and exclusive environment between the biometric data captured by the biometric sensor, and the match process of this biometric data against the biometric template stored in the Smart Card.

3.3 Conclusion

In this chapter an overview of the proposed solution, and its architecture, is presented. The proposed solution is intended to be easily deployed in existent authentication solutions using Smart Cards, and PIN to locally authenticate the user, providing a remote authentication system using biometric signatures. The presented architecture has three main layers, the Middleware, the embedded device reader, and the Smart Card layer.

The Smart Card layer is responsible for storing the sensitive data of the user, and to protect it against unauthorized accesses, as well as to perform the needed cryptographic operations. This layer is also responsible for ensuring the local authentication of the user, and to perform the biometric MOC operation. The Middleware layer is responsible for simplifying the use of the authentication system, providing a standard API to the user applications, which encapsulates the complexity of the communication with the embedded reader device, and the commands exchanged with the Smart Card.

The main advantage of the proposed solution, compared with current remote authentication systems, is the assurance of a secure remote authentication based on the biometric authentication of the user. With biometric authentication the presence of the person being authenticated is assured, in the local authentication process. This solution also increases the convenience of the users during the authentication process, given it eliminates the need for the user to memorize any information, such as his PIN.

In the next chapter, a prototype for the proposed solution is presented. This prototype is based on the architecture described in this chapter, and aims to be a proof of concept of the proposed solution. Technologies used in the implementation of each component, and technical considerations are detailed.

4 Prototype Implementation

Contents

This chapter describes the implementation of the prototype based on the proposed solution described in the previous chapter. With this prototype a user holding a Smart Card can be remotely authenticated by his biometric signature.

The implemented prototype is a proof of concept of the proposed solution, demonstrating how generic the developed architecture is, and proving that secure remote authentication of users by using his biometric signature can be implemented. The developed prototype is compatible with solutions, taking advantage of existent technologies. As described by the architecture of the proposed solution, the authentication process requires the use of a Smart Card, and an embedded device reader capable of capturing the biometric information of the user.

The development of the prototype is based on the architecture proposed and depicted in Figure 3.4. The architecture is composed by three main layers, the Smart Card internal structure, the embedded reader device, and the Middleware. Therefore, the implementation of the prototype consists on the implementation of the Smart Card and Middleware layers, since the correct functionality of the embedded reader device is assumed and out of the scope of this thesis. The Smart Card layer is responsible for storing the private data of the user, and for computing the security operations needed during the remote authentication process, while the Middleware layer is responsible for providing a generic Application Programming Interface (API) easy to be used by the user applications, encapsulating the complexity on the use of the developed authentication solution.

The architecture scheme of the implemented prototype is depicted in Figure 4.1, giving an overview of the different layers of the architecture, with their internal modules and the main used technologies. The Smart Card layer was implemented using a Smart Card with Java Card technology, fitted with an embedded Match-On-Card (MOC) library provided by *Precise Biometrics*. Inside the Java Card three applets were developed, the Identification, Authentication, and Signature (IAS), the Proxy, and the BioServer, implementing the authentication, the proxy, and the MOC modules respectively, as defined in the proposed architecture depicted in Figure 3.6. The architecture of the Middleware layer, depicted in Figure 3.9, implies the development of three components. The API provided by the Middleware layer to be used by the user applications was implemented accordingly to the Public-Key Cryptography Standards (PKCS#11) standard, the communication with the Java Card through the embedded device is assured by the Personal Computer/Smart Card (PC/SC) libraries, while the management of the biometric data between the user and the Java Card is performed using the Precise Biometric Software Development Kit (SDK).

The usage of the implemented prototype requires the enrolment of the user private data into the Java Card. A process to load the initial information of the user, and private data to be used during the authentication process was also developed, which led to the need for the development of an application where the user inserts his personal data, and an application where the user loads his biometric template.

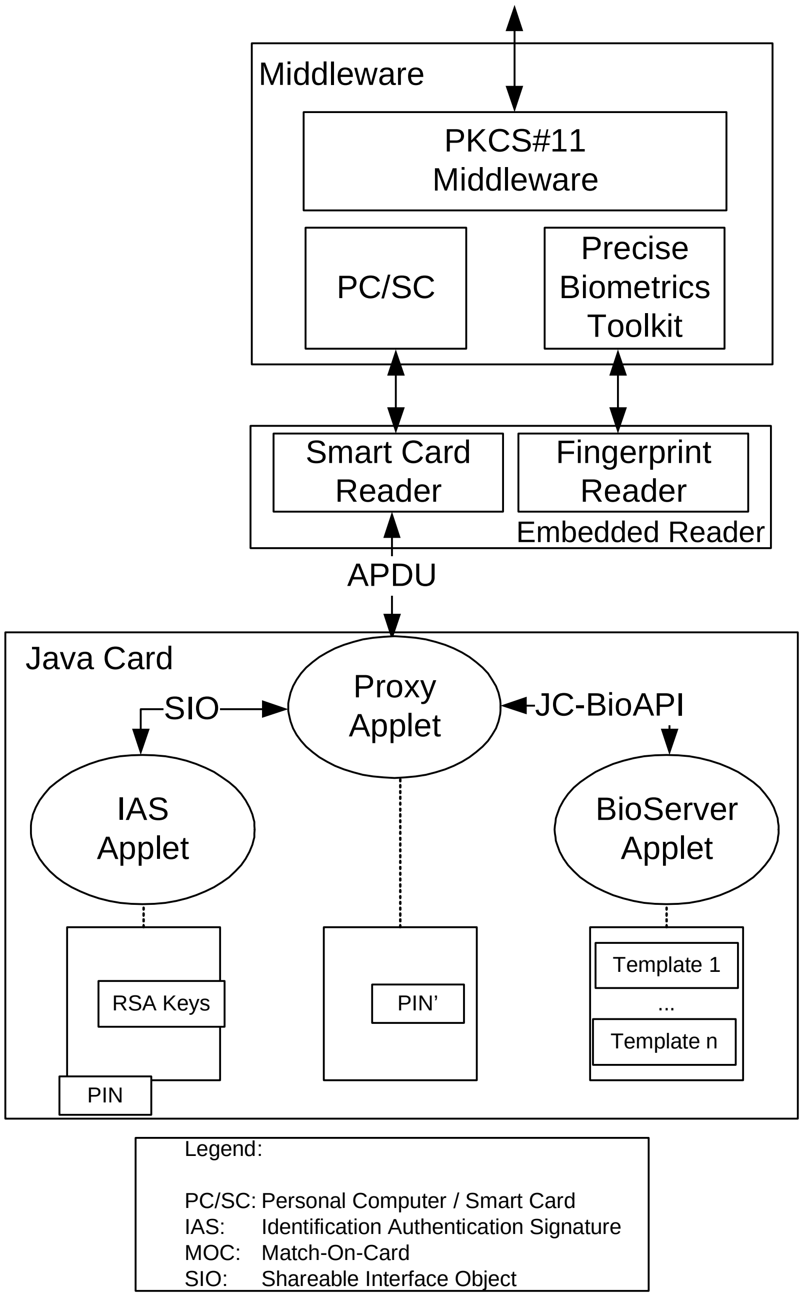


Figure 4.1: Architecture scheme of the prototype implementation.

The following sections detail the implementation and the development of each component of the prototype. Therefore, this chapter is divided in the main development phases, corresponding to the implemented software modules. The first section, 4.1, is intended to address the technologies chosen, as well as the implementation details of the internal required modules of the Smart Card layer. Section 4.2 describes the used technology and the implementation details of the developed Middleware components. The enrolment process and the developed applications including its structure and details are described in section 4.3. Section4.4 concludes the presentation of the implemented prototype.

4.1 Java Card Applets

The development of a prototype for the proposed solution implies the implementation of the Java Card layer components. This section is intended to present the Java Card components responsible for securely storing the user information, and performing the security operations needed for the remote and local authentication. The details, the main used technologies, and the major problems found during the implementation of the Smart Card layer are also described.

As described in the proposed architecture, the main components of the Smart Card layer are the authentication module, the MOC module, and the Proxy module. This architecture was designed to be easily deployed in existent remote authentication solutions using Smart Cards combined with Personal Identification Number (PIN). Therefore, it is necessary to use an already implemented remote authentication system to serve as the base of the prototype. The system being used has to have an authentication module assuring the remote authentication procedures, being able to securely store the user private key, and perform the digital signature of the challenge sent by the server, after authenticating the user locally with his PIN. Moreover, it also have also a MOC module, capable of executing the local biometric authentication of the user, and securely store his biometric template.

One of the most current used solutions satisfying the mentioned above requirements are the Electronic Identity (eID) cards. The Portuguese citizen card, for instance, carries an authentication module performing the remote authentication, and a MOC module assuring the local biometric authentication of the user. The Smart Card platform used on this card is the Java Card, and its internal structure contains, among others, the modules depicted in Figure 3.6.

In the Portuguese citizen card, the authentication module was designed according to the Identification Authentification Signature - European Citizen Card (IAS-ECC) recommendation [8]. Therefore, the authentication module is an IAS Java Card applet developed by *Gemalto*, providing all the necessary requirements to integrate the eID card in a public key infrastructure system, able to remotely identify the user. It is also useful to store information about the card holder, and other sensitive data. The MOC module in the Portuguese citizen card is also a Java Card applet. This applet was developed by the *Precise Biometrics*, and is used to perform the biometric local authentication of the user, by matching his biometric fingerprint template.

The eID cards, as is the case of the Portuguese card, are equipped with all the necessary technology required by the proposed architecture, since they can perform the remote authentication of the user with his PIN, and are able to execute his local biometric authentication. Therefore, the implementation of the prototype for the proposed solution is based on the deployment of a proxy applet on an eID card, as the Portuguese citizen card. The deployment of the proxy applet, along with the necessary alterations defined in the proposed solution chapter, makes it possible to adapt the eID card, making it capable of executing the remote authentication of the user, ensuring

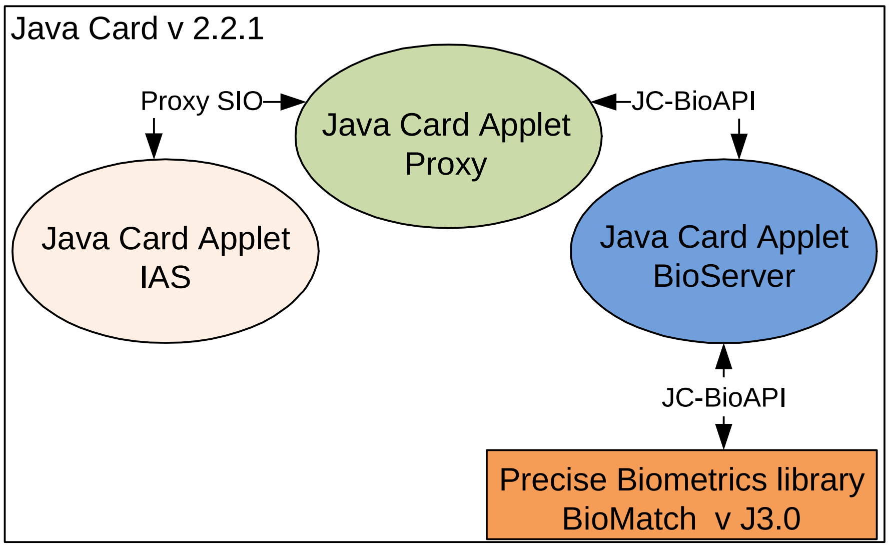


Figure 4.2: Java Card structure of the implemented prototype.

his local biometric authentication first.

The implementation of the proposed solution is based on the eID card technology; however, since all the software of these solutions is proprietary, it was not possible to have access to any existent versions. Consequently, to develop the prototype of the proposed solution it was first created a replica of an eID card, implementing both, the IAS and the BioServer applets. Only after this replica had been created, was the Proxy applet developed and loaded into the Java Card. The design of the developed software inside the Java Card is depicted in Figure 4.2.

Following the Portuguese citizen card example, a Smart Card with the Java Card platform was used. The Java Card is a secure and reliable solution, given that it provides a trusted environment for applications that run on. Multiple applications can be deployed on a single Java Card, and new ones can be added to it, even after it has been issued to the end user. This platform also assures that the applets written in the Java Card programming language can be executed, securely, on Java Cards from different vendors. Therefore, the Java Card platform assures the security and reliability needed by solution.

The Java Card used was supplied by the *Precise Biometrics*, using Java Card version 2.2.1, and a biometric MOC library, called BioMatch vJ3.0. This library is able to provide the secure storage of biometric fingerprint templates, and perform the biometric fingerprint match operation inside the card. The MOC algorithm of the BioMatch library is proprietary, and it is only possible to access it through internal applets using the Java Card Biometric Application Programming Interface (JC-BioAPI) .

The used Java Card contains a BioManager applet, providing an external interface to be used by the host side through Application Protocol Data Unit (APDU) commands, and also providing an internal proprietary interface to be used by internal applets through a Shareable Interface Object

(SIO). The BioManager applet in the used Java Card is intended to execute the management of the biometric data inside the card, by being able to communicate with the BioMatch library inside. Since this BioManager is proprietary, not being possible to get access to its code, and since its internal API does not implement the JC-BioAPI standard recommendation, a new applet, named BioServer, was developed.

The developed BioServer applet takes advantage of the available BioMatch library inside the Java Card, in order to use the proprietary MOC algorithm from *Precise Biometrics*. It implements an external API respecting the same API provided by the Precise Biometrics BioManager applet; however its internal interface implements the JC-BioAPI. Therefore, any internal applet, as is the case of the Proxy applet, can communicate with it using the JC-BioAPI. Even though fingerprints are used as the biometric authentication mechanism, the generic architecture proposed in this thesis, and the wide coverage of biometrics by the JC-BioAPI, allows for the use of other biometric technologies. The fingerprint was used since it is the key technology provided by the Pricise Biometrics, and since it is one of the most used and accepted biometrics in current days by the Portuguese society. A manual with the implementation overview of the developed BioServer applet is attached in A.

The IAS applet present in the Portuguese citizen card is also proprietary, and it was not possible to get access to its code, or any trial version. Therefore, a Java Card applet ensuring the required operations, and designed according to the IAS-ECC standard was also developed. The IAS applet was developed in a way that it can assure the crucial methods required for the remote authentication herein proposed, such as cryptographic and storage operations. Therefore, an applet was developed with the external interface defined on the IAS-ECC standard, able to securely store the user private key, and implementing a binary file system according to the International Organization for Standardization / International Electrotechnical Commission (ISO/IEC) 7816, to store the public certificates and personal information of the user. In order to equip the IAS with all the requirements that the system imposes, an internal interface was added, given it the capability to communicate with the Proxy applet. A manual with the implementation overview of the developed IAS applet is attached in B.

With the development and loading of the developed IAS and BioServer applets, the Java Card used is able to emulate the Portuguese citizen card, and become a prototype base of current solutions. It can be used to perform remote authentications with the user private key, protected by a local PIN authentication, and can also be used to locally authenticate the user through his biometric signature, independent of the IAS applet. To create the prototype of the proposed solution it is now necessary to develop the Proxy applet according to the specifications defined in the proposed solution, and load it into the Java Card.

The developed Proxy applet is intended to manage the authentication process inside the Java Card, ensuring the remote authentication of the user with his private key using the IAS applet, just after a successful biometric local authentication of the user at the BioServer applet. The Proxy applet works as the front end of the entire system inside the card. When a command arrives to the Proxy applet, it first verifies if it is a specific command to be processed directly by it, or by the BioServer applet. If the command is specific for the Proxy, it executes it, and returns the response; otherwise, if the command is specific for the BioServer applet, it redirects it to the BioServer applet. If the command it is not specific for the Proxy nor to the BioServer applet, the Proxy applet forwards it to the IAS applet.

During the authentication process only the Proxy applet is called by the Middleware, where the three applets are seen from the outside as one single applet. To communicate with the BioServer and the IAS, the Proxy applet uses the JC-BioAPI and the IAS SIO respectively. A manual with the implementation overview of the developed Proxy applet, and the internal interface used by the IAS applet, is present in appendix C.

The three Java Card applets were developed using the Java Card Development Kit (JCDK) of the *Oracle*. The JCDK offers the needed Java Card libraries for the development of the applets, and the needed tools for the compilation, and testing of its code. It also provides the tools for the generation of the applet's installation byte-code, needed to be loaded into the Java Card. The Integrated Development Environment (IDE) used to aid in the development of the software, and the integration with the JCDK, was *Eclipse*, with the JCDK plug-in installed.

The Java Card technology does not define the load mechanism of the applets into a Java Card. Therefore, an application implementing the Global Platform recommendation was used to load the developed applets into the Java Card. This Global Platform implementation is capable of interacting with the Applet manager of the Java Card, and after performing the needed authentication over the required Secure Channel Password (SCP), it sends to the Java Card the JCDK generated code, needed to install the applet. The APDU commands are sent directly to the manager applet of the card, following the structure defined by the Global Platform. A manual with a tutorial for the installation of a Java Card applet, using the Global Platform implementation, is attached in D.

Using the Portuguese citizen card as the existent solution to base the development of the prototype, demonstrates that the proposed solution can be deployed in large-scale solutions, allowing the use of biometrics in remote authentications. The next section describes the development of the Middleware layer and its particularities.

4.2 Middleware

The Smart Card layer implements the crucial operations of the authentication process. However, the implementation of the prototype implies the development of a Middleware to provide a standard interface to be used by the applications. This section presents the details and technologies used in the implementation of the developed Middleware.

The Middleware layer of the proposed solution's architecture aims to provide an API for the user applications, in order to encapsulate the complexity of the communications with the Smart Card reader, and the biometric sensor. Therefore, the Middleware has to provide the necessary requirements to enable an easy integration of the developed solution in existing systems and applications, in a way that they can benefit from the authentication mechanism implemented in the Smart Card layer. One of the main functions of the Middleware layer is the management of the local biometric authentication process, transparently to the applications using it.

The layout architecture of the developed Middleware follows the specifications defined in the proposed solution chapter. The main structure of the Middleware is divided in three main components (depicted in Figure 3.9), the public API provided for the user applications, the library to deal with the Smart Card communication, and the libraries to deal with the biometric capture and match.

Existing solutions, as the Portuguese citizen card, already have a Middleware providing the required features to easily use of the security authentication mechanism provided by the Java Card. However, the Middleware of the current solutions is merely prepared to deal with cryptographic operations requiring the user local authentication to be performed with PIN. On these Middleware implementations, when the card requests the user PIN authentication, the Middleware manages the local authentication process by requesting the user to insert his PIN, and sending it to the Smart Card encapsulated in the *Verify* APDU command. The requirements of the proposed system imply the addition of the capacity to deal with biometric authentication of the user. The user has to be informed by the Middleware that he has to proceed with a local biometric authentication, by putting his fingerprint in the biometric sensor. Therefore, the Middleware has to manage all the process of the biometric authentication, encapsulating its complexity to the user applications.

The *Zetes* company developed the Middleware to interact with the Portuguese citizen card. Given that this Middleware is proprietary, and it was not possible to have access to it, a Middleware to interact with the developed Java Card solution was developed, including the necessary components for the execution of the biometric authentication of the user, when requested by the Java Card.

The Middleware code was developed using C++ programming language, since it contains some language extensions over C, which makes object oriented programming more convenient. Furthermore, its code is easy to export to *.dll* or *.so* libraries files. Once the performance of a program developed in C++ is close to performance obtained by a program developed in C, many software libraries to interact with Smart Cards are also developed in C++ language. Figure 4.3 depicts the main modules that constitute the developed Middleware.

The API provided by the developed Middleware is the PKCS#11. The PKCS#11 is an open source standard API for cryptographic tokens as Smart Cards, being supported by open source

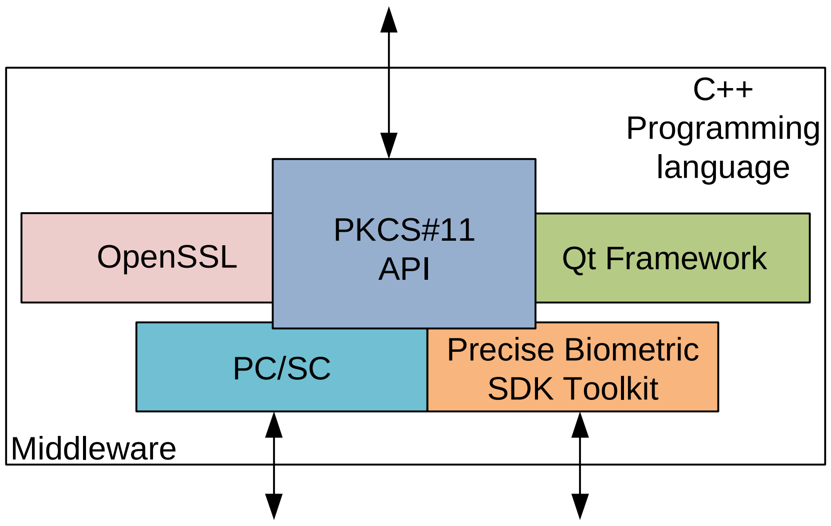


Figure 4.3: Middleware structure of the implemented prototype.

applications as the web browser *Mozilla Firefox*, in a wide variety of operating systems. Since the PKCS#11 standard interface has a wide range of methods, only the necessary methods were implemented. However, to use the cryptographic methods necessary to the interaction with the Smart Card layer, it was necessary to implement almost every function related with the general purpose usage, the slot and token management, the session management, and the object management. A list with all the implemented functions of the PKCS#11 is available at E.

The interaction with the user has forced the development of two window applications. One to request the user to input his PIN, and the other to request the user to insert is finger in the biometric sensor. Both windows were developed in order to be possible to test the IAS applet alone, requesting the user PIN before the proxy applet was installed, and requesting the biometric local authentication after. The applets were developed using the Qt framework developed by *Nokia*. Since it uses c++ programming language, and is a cross platform, it becomes easy to use in a wide variety of operating systems, and is easily integrated with the developed PKCS#11 Middleware.

The PKCS#11 Middleware needs to read the user public certificate, which lead to the need to use the *OpenSSL* software library. This software is open source, and is also implemented to be used in a wide range of operating systems. In the current Middleware implementation it provides the possibility for PKCS#11 to access the variables and components of the user public certificate. Therefore, the Java Card stores the user public certificate as a binary file, and does not need to keep separately all its components and variables.

The communication with the Smart Card reader was performed using the PC/SC software libraries. This software was chosen since it can be used it in a wide range of operating systems, and is compatible with a wide range of Smart Card readers from different manufactures, by respecting the main standards in Smart Card communications.

The interaction with the biometric sensor component in the embedded reader device, implied the use of the Precise Biometric SDK toolkit. This toolkit was used to capture the user fingerprint image, and convert this image into biometric data to be matched against the biometric template of the user, stored in the Java Card. This SDK is optimized to communicate with the used biometric sensor, and is easy to use.

The embedded device used for the communication with the Smart Card and the capture of the biometric data of the user was the Precise Biometric 250. This embedded device has the capacity to read and communicate with Smart Cards, as well as to capture the biometric fingerprint of the user. It is a requirement of the proposed architecture the usage of an embedded device to capture the user biometrics and to communicate with the Smart Card, in order to ensure a protected environment for the biometric template.

The developed Middleware provides an easy way of using the developed prototype, by the user applications. The next section presents the enrolment process, needed for the authentication system to receive the user private data.

4.3 Enrollment Process

The implementation of a prototype for the proposed solution consists of the development of the Smart Card and Middleware layers. Nevertheless, the development of these layers is generic, being necessary to personalize each Java Card for each user, through an enrolment process. This section describes the details of the developed enrolment process, including the implemented applications and the used technologies.

The proposed architecture includes the necessary modules to run on the host side, allowing an easy interaction with the authentication system. In addition to the developed applets and Middleware, an enrolment process was developed, in order to initialize and personalize the user Smart Card. The enrolment process consists on the initialization of the IAS applet with the user personal information and private keys, and in the loading of the user biometric template into the BioServer applet.

The enrolment process of the user, into the Java Card is divided in three main phases. First, the user private keys and public certificates are generated. Second follows the phase where the user name, id, photo, private keys and public certificates are loaded into the IAS applet, on the Java Card. The latest phase consists on capturing the user biometric template, and afterwards, loading it into the BioServer applet, inside the Java Card. In the end of this process, the Java Card is personalized with the user information, and is ready to be used in a real authentication scenario. The three main modules needed for the enrolment process are depicted in Figure 4.4, where the main components involved in the enrolment stage are depicted.

The first phase of the enrolment process consists on the generation of the user private keys

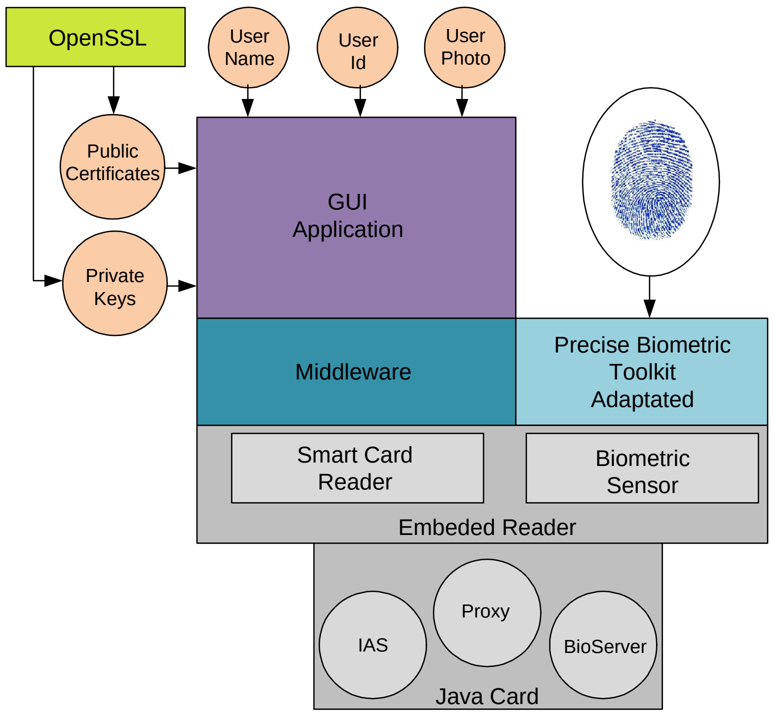


Figure 4.4: Enrolment process scheme.

and public certificates using OpenSSL software. OpenSSL is an Open Source toolkit implementing the Secure Sockets Layer (SSL) v2/v3 and Transport Layer Security (TLS) v1 protocols, as well as a full-strength general purpose cryptography library. It can be used in a wide range of operating systems, being easily handled and well documented. Any other technology or method can be used to generate this security information, as long as it is ensured that no external entity can access it.

After the generation of the security data, the private information of the user such as name, id, photo, private keys, and public certificates in the IAS applet, are inputted through an application developed in JAVA. This application has a Graphical User Interface (GUI) developed to guide the user during the enrolment process. This interface is a user friendly way to request all the needed information to the user.

The JAVA application behind the GUI validates the entire data insert by the user, in order to eliminate the error during the enrolment process. Its main function is to manage the execution flow of the commands sent to the IAS applet into the Java Card, transparently to the user. Thus, it assures the creation and space allocation of the files storing the user data inside the IAS applet. In order to encapsulate the communication with the Smart Card reader, and the low level specifications during the communication with the Java Card, a specific Middleware was developed. The JAVA application communicates with the Middleware through the Java Native Interface (JNI) technology. The Middleware provides a JNI, and is developed using C++ programming language. In the Middleware, the communication with the Smart Card reader is also assured by the use of

PC/SC libraries.

The third phase of the enrolment process is responsible for the loading of the user biometric template into the BioServer applet. The Toolkit of the precise biometrics already has a biometric template enrolment example. Since the developed BioServer applet follows exactly the same external interface as the BioManager applet, the Precise Biometric Toolkit was used, with some adaptations. The adaptations required on the Precise Biometric Toolkit consist on the alteration of the destination address in the commands sent, in order to redirect them to the BioServer instead of the BioManager.

In short, the enrolment process required the development of an application to initialize and load the user private data into the IAS applet, and an adaptation of the existing precise biometric toolkit example to load the user biometric template into the BioServer applet. The application developed to perform the enrolment of the user information into the IAS applet enables the Java Card to be used as the source of the remote authentication system, since it is loaded with the user private and crucial information. This application is easy to use, enabling the loading of the user information without the need to know the details, and the complexity of the enrolment process specifications. Since it validates the data, and formats the commands to be sent to the Java Card, it encapsulates all the complexity of the process. The biometric data is enrolled by using tested software, adapted to communicate directly with the developed BioServer applet, in this way assuring an easy and reliable loading process.

4.4 Conclusion

This chapter describes the development of a prototype for the proposed solution. The solution presented on this thesis is an authentication mechanism capable of assuring the biometric authentication of the user in a remote authentication scenario. The developed prototype follows the specifications and architecture described in the previous chapter, proving all the functionalities for the proposed solution.

The architecture of the solution implies the implementation of two main layers. The first one is the Smart Card layer, responsible for the storage of the private data of the user, and for the computation of the security operations needed in the remote and local authentication process. The second is the Middleware layer, responsible for providing an abstraction for the complexity of the interaction with the Smart card layer and with the Smart Card reader and biometric sensor.

The Smart Card layer was developed using a Java Card. Three applets were developed to run on the Java Card: the IAS applet, the Proxy applet, and the BioServer applet. These three applets are the security source of the proposed system, providing the needed storage and security processing operations. Even though three different applets exist, they are seen by the outside entities as one single applet, capable of performing the user remote biometric authentication.

The Middleware layer implements a generic API used by the applications that need the provided authentication mechanism. The implemented API was the PKCS#11, allowing this solution to be used in a wide variety of operating systems. The Middleware provides a known API encapsulating all the complexity of the communication with the Smart Card reader and the biometric sensor. It also encapsulates the complexity of the communication with the applets inside the Java Card, respecting the correct execution flow of the authentication process.

An enrolment process was also designed to personalize and initialize the Java Card with the user private information. The enrolment process lead to the need for the development of a Java application with a GUI, for the insertion of the private and personal data in the IAS applet. The adaptation of the Precise Biometrics SDK Toolkit was also performed in order to load the user biometric template into the BioServer applet.

The implementation of the prototype was based on the existing eID cards, as is the case of the Portuguese citizen card. This work also shows the compatibility of the proposed solution with existing technologies. The proposed eID card is able to provide the remote authentication of the user, ensuring his local biometric authentication first. The developed prototype shows that it is possible to add the local biometric authentication feature, to existent solutions, without a significant cost. The proposed solution assures to the remote server the presence of the user during the authentication process, and removes the user need to remember any secret information.

The security analysis of the proposed solution and of the implemented prototype is performed in the next chapter.

5

Assessment of the Solution

Contents

This chapter performs an assessment of the solution proposed in this thesis. An analysis of the proposed solution and the results of the tests performed to the implemented prototype are herein performed.

The proposed solution claims to be the architecture of a secure authentication mechanism, able to authenticate a remote user using his biometric signature. The implemented prototype of the proposed solution shows that the proposed architecture can be implemented in using the existent technologies, and deployed in existing solutions. Therefore, this section describes the performed analysis to the security of the developed solution, and the functionality of the implemented prototype in a real scenario.

The analysis of the implemented prototype is performed by the results obtained during the test of the developed solution in a real scenario. The test of the implemented prototype was performed with a web application authenticating the user towards a remote server. A web server was developed and used as the remote server, while the used user application was a web browser. The developed Middleware, capable of communicating with the Java Card, was installed in the browser. The analysis to the security and functionality of the proposed solution is detailed in the following sections. This chapter is divided in 4 sections. The first section, 5.1, details the developed test environment in which the implemented prototype was analyzed, including the used technologies. Section, 5.2, presents the performed tests to the implemented prototype, and the analysis of the obtained results. The security analysis of the developed solution, and of the implemented prototype, is performed in section 5.3. The final section, 5.4, concludes this chapter with some final considerations to the performed analyses.

5.1 Test Environment

The tests performed to the prototype, implementing the proposed solution, were performed using an environment test developed to represent a real authentication scenario. This section describes the developed authentication scenario, and the technologies used in the creation of the environment test.

The developed prototype is basically composed by the Java Card applets, and by the Middleware, implementing the proposed solution. The proposed solution was designed as a remote authentication mechanism capable of using the user biometric signature, taking advantage of the existing technologies. Therefore, the implemented scenario to test the developed prototype, aims to demonstrate the functionality of the proposed solution by requiring a remote authentication of the user, based on his biometric authentication. The authentication of the user in the environment test is performed by the developed prototype, assuring at the same time his local biometric authentication.

The environment test showed that the proposed architecture can be implemented in existent solutions, and can be used by current systems. A web environment was developed, being one of the most used systems requiring user remote authentication. The developed web site was deployed on an apache web server. The web server represents the remote authentication server, to whom the user wants to authenticate. Nevertheless, the authentication will be established between the web server, and the web browser. Thus, the web browser represents the user application using the developed Middleware to interact with the developed authentication system, present in the Java Card.

The authentication between the application of the user, and the remote server, consists on the establishment of a HyperText Transfer Protocol Secure (HTTPS) connection in which the security operations required in the user authentication process are executed by the developed prototype. When the web browser needs to sign the challenge received from the remote server, to authenticate the user, it uses the installed Middleware to interact with the user Java Card, possessing the user private key. Figure 5.1 represents the scheme of the developed environment test.

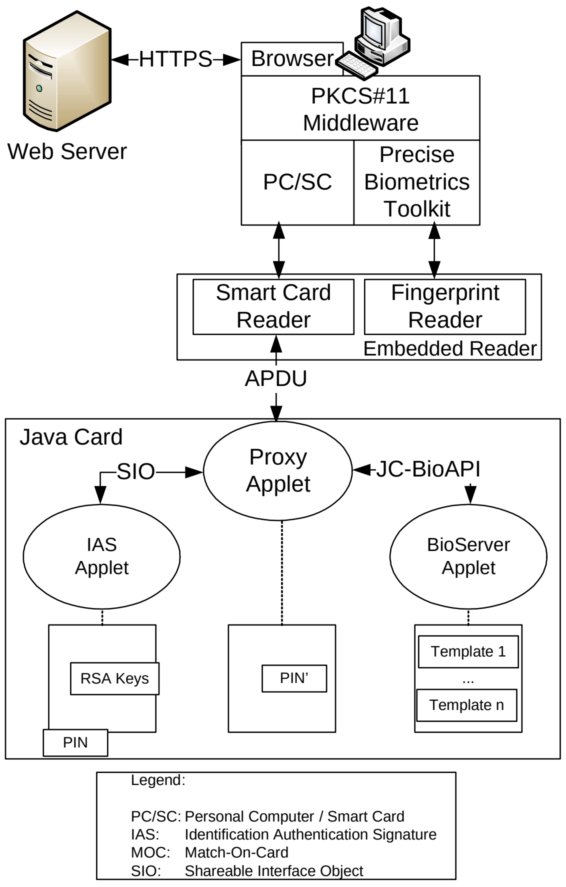


Figure 5.1: Environment test.

The request for the signature of the challenge is required during the establishment of the Transport Layer Security (TLS) handshake of the HTTPS connection, once the web server requires the client authentication. During this authentication phase the user application has to send the user public certificate apart from having to sign the challenge. Therefore, the web browser requests the user public certificate, and the signature operation, to the user Java Card, through the Public-Key Cryptography Standards (PKCS#11) Middleware.

The Middleware sends the sequence of the Application Protocol Data Unit (APDU) commands directly to the Proxy applet, since this is the front end of the authentication system implemented in the Smart Card layer. The Identification, Authentication, and Signature (IAS) applet is responsible for the storage of all the private information of the user and the performance of the security operations required in remote authentication. Therefore, the public digital certificate and the entire signature operation requests are redirected by the Proxy applet to the IAS applet. Whenever the IAS applet uses the user private key to perform the signature of a challenge, a local biometric authentication of the user is required. The Proxy applet manages the user local authentication inside the Java Card, by authenticating the user in the IAS applet after a successfully local biometric authentication. Out of the card, the Middleware is the entity responsible for managing the biometric authentication.

The developed PKCS#11 Middleware asks the user to put his finger on the biometric sensor, and sends his biometric fingerprint template to the Proxy applet. Once again, the Proxy applet serves as a front end of the Java Card system, this time proxing the BioServer applet. Once the BioServer applet possesses the user biometric template and is able to use the BioMatch library, it is the entity responsible for performing the user biometric authentication, against the biometric fingerprint template received from the Proxy applet. If a successful biometric authentication is reported by the BioServer, the Proxy applet authenticates the user in the IAS applet using his Personal Identification Number (PIN), and informs the Middleware that the user is locally authenticated in the Java Card system.

The signature of the challenge sent by the server is requested again by Middleware to the Java Card, and the result is provided to the web browser, to be sent to the web server in order to conclude the user remote authentication. This process assures that the user performs a local biometric authentication, every time that a signature with his private key is performed.

The technology used in the development of the environment test is depicted in Figure 5.2.

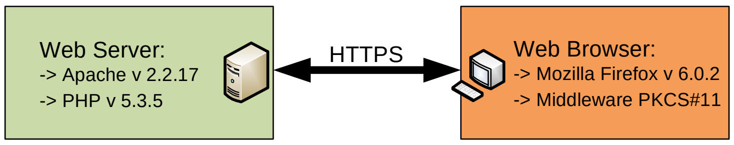


Figure 5.2: Technology used in the environment test.

The used web server was an apache version 2.2.17, with the Hypertext Preprocessor (PHP) version 5.3.5. The web site developed, presents the user name and id after a successful remote authentication. The user name and id are requested to the web server using PHP, and are obtained from the user public certificate, available after a successful HTTPS connection. The web browser used was the Mozilla Firefox version 6.0.2, running the developed PKCS#11 Middleware.

5.2 Test Results and Analysis

The tests performed on the developed prototype of the proposed solution, using the environment test presented in the previous section, are detailed in this section. This section also aims to present the obtained results, and analyze them based on the goals of the proposed solution.

The proposed solution consists of a remote authentication system using the user biometric signature, being at the same time compatible with the existent technologies. The proposed solution was developed in order to ensure the presence of the user during the authentication process, and to remove from the user the need for him to memorize any additional information. Its design also has in account the protection of the user biometric template used in the biometric authentication process. The main goal of the developed environment test is to provide a real scenario to test the implemented prototype, demonstrating the functionality of the proposed solution and its compatibility with existent systems.

The enrolment process was applied on the Java Card, where the user loaded his name, id, photo, and the generated private keys and public certificates, using the developed java Graphical User Interface (GUI) application. After the registration of the user private information needed for the remote authentication in the IAS applet, the user biometric template was loaded into the Java Card BioServer applet, through the Precise Biometrics Software Development Kit (SDK) toolkit with the alterations presented in the enrolment process described in section 4.3. The enrolment process allowed the personalization of the user Java Card, thus initializing the authentication system with all the requirements of the proposed solution specific for the user.

All the created tests were performed using the developed environment test. The user, using the web browser with the PKCS#11 Middleware installed, opens the login web page. The web page is accessed by HyperText Transfer Protocol (HTTP), and contains a button for the user to perform his login. The HTTPS connection is required by the web server during the login, and the user has to perform his biometric authentication to open the web page containing his name and id. The first test was performed by the user that previously had performed the enrolment on the Java Card. With the correct user using the authentication system the authentication was performed successfully, proving that the developed system works and can be used in real scenarios with existent technologies.

The second test consisted on the confirmation of the correct functionality of the biometric authentication, performed locally by the BioServer applet in the Java Card of the user. The local biometric authentication of the user has to reject all other users, aside from the one who performed the enrolment on the Java Card. Therefore, a login was performed in the web page used for testing, by a user not enrolled on the Java Card. The biometric authentication of the user failed and the HTTPS connection was cancelled, being the reason why the user web page was not made available by the web server. The web browser showed an error page, reporting an error on a PKCS#11 interaction command.

The third test was made in order to test if the biometric authentication of the user is able to distinguish the finger inserted by the user enrolled on the authentication system. Therefore, a login was performed in the web page used for test by the user enrolled on the Java Card, but using a different finger from the one used in the enrolment to generate the biometric template. The biometric authentication of the user has also failed, and the web browser showed an error page reporting an error on the PKCS#11 interaction command.

The fourth and most complex test was performed with the purpose of verifying if the PIN used by the IAS applet on the user Java Card, to locally authenticate the user, was successfully changed by the Proxy applet, for one randomly chosen. The change of the PIN in the IAS applet prevents the user from using it directly, by forcing them to use the Proxy applet, responsible for managing the authentication process inside the Java Card. Figure 5.3 depicts the purpose of the fourth test.

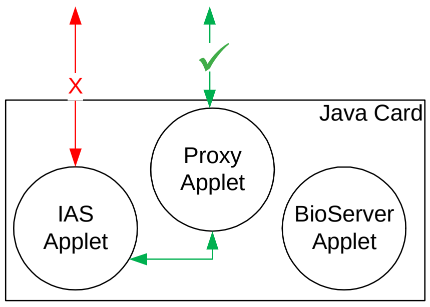


Figure 5.3: Test the direct access to the IAS applet.

This test implied the alteration of the used Middleware. It must be remembered that the Middleware is not a component of the proposed authentication system responsible for providing security. Rather, it is a means to achieve a good abstraction for user applications, encapsulating the complexity of the communications with the user Java Card. Therefore, a user can use the developed Middleware, or develop his own Middleware, in order to interact with the authentication system. The alteration performed to the Middleware, enabling it to be used on this test, consisted on the redirection of the challenge signature request directly to the IAS applet, instead of passing through the Proxy applet. Therefore, when the IAS applet intends to use the user private key to perform the signature, the PIN is requested to locally authenticate the user, instead of performing a biometric authentication. The user then inserts his PIN on the popup window prompted by the Middleware. Since the PIN was correctly changed by the Proxy applet, the authentication of the user has failed, and the HTTPS connection was cancelled, having the web browser show an error page reporting an error on a PKCS#11 interaction command.

The tests performed to the implemented prototype showed that the proposed solution works correctly and prevents unauthorized users from using the authentication system to authenticate themselves in the remote servers. A security analysis of the proposed solution is performed in the following section, where an indepth discussion of the security of the biometric template during its usage cycle is performed.

5.3 Security Analyses

This section aims to present a security analysis for the proposed solution and for the implemented prototype. The main components that represent the focus of the security analysis are detailed and analyzed. The security analysis aims to present the main concerns in the usage of the proposed solution, as well as the consequences of the breaches of some security assumptions.

The analysis performed in this section aims to analyze the possible points of failure of the proposed solution, as well as the severity related to each one. This security analysis is important in order to define which limitations the proposed solution has, and to specify the precautions needed in the developed prototype. This analysis can also be used to define the areas needing better improvements, in order to assure the global security of the system.

The scope of the security analysis herein proposed covers the Java Card system, responsible for the storage of the user's private information such as the private keys and biometric templates, as well as for the performance of the crucial operations needed during the authentication process. The security of the biometric template during its usage cycle is also analyzed, since it is one of the major concerns in the use of biometrics on current remote authentication systems. Finally, the necessary trust between the entities involved in the user authentication is also examined. The consequences of a breach of confidence due to the compromise of one of these entities are described, and the problems surrounding the introduction of the proposed solution in an already implemented system are detailed.

The Java Card is the source of all the security functionality in the proposed authentication system, once it possesses the user private keys and biometric template. Given that, the Java Card is a secure platform, and the Smart Cards a tamper resistant system, the private information of the user is securely protected against physical attacks made directly to the user Java Card. Even if the user loses his Java Card, or if it is stolen, an attacker cannot have access to the private information kept inside it. The attacks made by software to the user Java Card are also not able to get the user private information, since this data is kept in Java Card applets, responsible for assuring that the private data never leaves the card. To ensure this, the applets perform the security operations needing the private information of the user.

The IAS applet ensures the storage and usage of the user private key, and the BioServer applet assures the storage and correct usage of the user biometric template. The operations requiring the usage of the user private key, as is the case of the signing of the challenge sent by the remote server, always require the user local authentication. As explained in the proposed solution chapter and validated in the previous section 5.2, the Proxy applet manages the authentication of the user inside the Java Card, forcing him to perform a biometric authentication, which ensures his presence during the authentication process. Therefore, if a malicious person has access to the user Java Card it cannot use it to perform a remote authentication, since he does not possess the biometric signature of the user.

After the overview on the security of the Java Card system, it is important to consider the weaknesses of dealing with the biometric templates, in this security analysis. The security of the fingerprint template of the user has to be analyzed from the moment in which it is captured by the user's finger on the biometric sensor, until it reaches the BioServer applet, inside the user Java Card, to be matched against the template previously stored during the enrolment process. The intervenient entities on this process are the user, the embedded device composed by the biometric sensor and the Smart Card reader, and the Java Card of the user.

The most important considerations regarding the security of the biometric template are related to the fact that the template, used during the authentication process by the Java Card, has to be ”fresh”, which means that it has to be captured in the moment of the authentication, avoiding this way the use of captured biometric templates by possible attackers. Apart from that, it has to be assured a secure and exclusive environment to the biometric template, assuring that no external entity has access to the user biometric data sent during the local biometric authentication of the user.

To assure the protection of the biometric template specified above, the embedded device has to be able to manage the biometric authentication of the user, without the use of any external entity, providing a secure and exclusive environment for the user biometric template. Since this requirement is vital for the functionality of the presented system, it is important that the Java Card authenticates the embedded device, assuring that it was validated by an external entity responsible for verifying if it respects the requirements of the system. Therefore, the embedded device is certificated by the Certification Authority (CA). It possesses a private key and a public digital certificate with its public key, signed by the CA.

The Middleware sends a command to the embedded device that triggers the capture of the biometric data and its validation. After that, the biometric authentication of the user starts inside the embedded reader device, with the establishment of a secure channel with the Proxy applet inside the Java Card. The embedded device is public, and susceptible of being attacked by any person. If the embedded device or his private key were compromised, all the security of the system is compromised, since the protected and exclusive environment of the biometric template is violated, and can then be forged. This protected environment is essential to assure that the

Match-On-Card (MOC) operation is always performed with a fresh template. As depicted in Figure 5.4, the protected environment assures that no external entity has access to the biometric template.

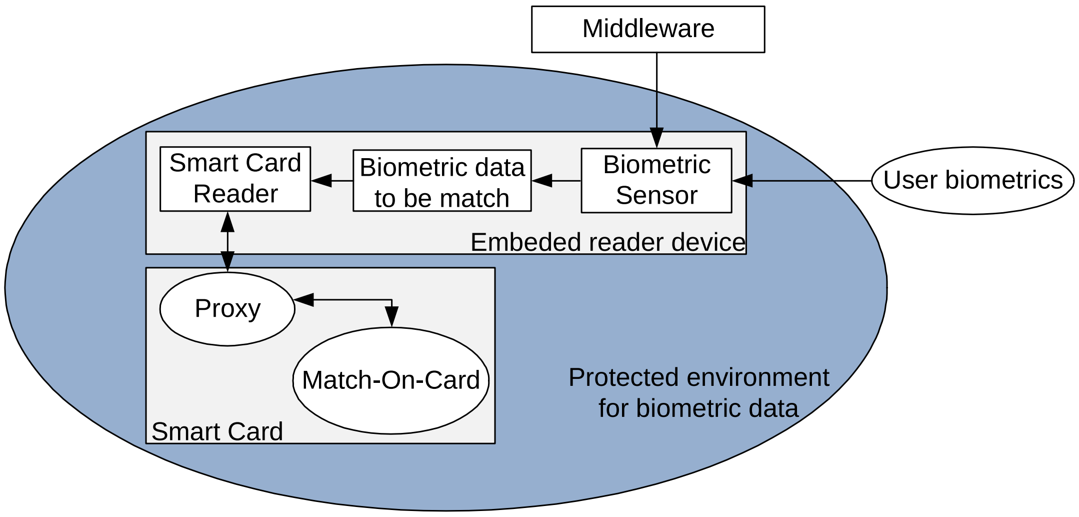


Figure 5.4: Protected environment for biometric data in the biometric match process.

The embedded reader device receives the request for a local biometric authentication from the Middleware, and triggers the procedure flow to perform the local authentication of the user in the Java Card. Therefore, it captures the biometrics of the user, and generates the biometric data to be match by the BioServer applet. It must be the embedded device the one who sends the generated biometric data directly to the Proxy applet inside the Java Card, using the secure channel previously established. The biometric sensor will send it, through the Smart Card reader component in a specific format to be matched inside the Java Card.

The Proxy applet in the Java Card trusts in the biometric data coming to be matched, once the embedded device is certificated and authenticated. The template received is necessarily ”fresh”, since it was captured by the certificated reader device, in the moment of the local authentication. Therefore, the biometric data of the user used to be match against the user template, never had contact with any external entity, being assured its confidentiality and protection. This scheme prevents the local authentication system from accepting old templates, or templates generated by external and unreliable sources, also protecting the system against attempts to copy the real template. To spoof the system a falsification of the biometrics of the user is needed, since it cannot be spoofed simply by presenting a faked/stolen biometric template. The scope of this thesis was to develop the solution to be installed on the Java Card system. The security of the biometric template is assured by the embedded device, and is an assumption required for the correct and secure functionality of the proposed system.

Until now, only the analysis of the Java Card system, and an evaluation of the cares needed with the management of the biometric template were performed. Therefore, it is also necessary to analyze the security perspective of the remote server. First, the server has to trust in the CA during the authentication of the user Java Card, to be sure that it is communicating with the Java Card of the correct user. This trust is very important, since it is the private key of the user that assures that the Java Card being used is the correct one, and possesses the user biometric template required to perform the local biometric authentication of the user. If this trust is compromised by an attacker discovering the private key of the CA, all the security of the system is compromised, since the would be able to sign his public certificate, as if it was a different and valid user.

The server also has to trust in the Java Card system, after it knows that it belongs to the correct user and it is validated by the CA. This trust assures to the server that the user presence was verified with his local biometric authentication, executed after following all the required precautions specified in the proposed solution.

The entire system proposed in this thesis is able to be implemented in existing solutions, as was shown in the previous chapters. However, if this system is deployed in a solution already in operation, where the user local authentication is assured by his PIN, the server may be unable to be sure if the Java Card used to authenticate the user already has the Proxy applet installed or not. Therefore, the server may be unable to be sure if the user being authenticated with his correct private key had performed a biometric or a PIN local authentication. One solution for this situation is to add additional information on the public certificate of the new users using the proposed solution installed on their Java Card. Thus, the server could verify the user public certificate to check if the Java Card being used to authenticate the user already has the Proxy applet installed, and thus performs a biometric authentication of the user. Another solution would be to assign a different level of IDs to the users whose Java Card already had the proposed solution installed. This last solution can be impossible to apply if the ID of the users cannot be changed, or is pre-established.

The analysis performed in this section shows the security of the proposed solution, and the requirements to use the implemented prototype. An observation was made of the security provided by the Java Card system, and the cares to have in mind when using this system in a real scenario. The next section presents the main conclusions of the analysis here in performed.

5.4 Conclusion

This section presents the conclusion about the assessment of the proposed solution. A brief overview of the chapter content is presented, summarizing the developed test environment, the results obtained during the test of the implemented prototype, and the main conclusions on the security analysis of the proposed solution.

The assessment of the solution was performed in order to determine the security and reliability of the proposed authentication system. The proposed solution is intended to perform the remote authentication of a user, using his biometric signature. Therefore, a prototype was implemented respecting the architecture defined in the proposed system, and its behavior tested in a real remote authentication scenario. The scenario implied the authentication of the user in a remote server, using his Java Card with the prototype installed, in order to ensure his local biometric authentication.

The developed test environment is composed of a web server requiring the user authentication through an HTTPS connection, established via the user's web browser. The the PKCS#11 Middlewar was installed in the web browser, which is responsible for redirecting the requests for the security operations related with the user authentication phase of the HTTPS, to the user Java Card.

The environment test was used to test the developed prototype. The realized tests allow us to conclude that the proposed solution works in existent systems, and can be implemented using current technology. The authentication of the correct user occurred successfully, while all the attempts by different users failed. The alteration of the PIN in the IAS applet was also tested, being verified that the user cannot use his PIN to directly authenticate himself in the IAS applet during the remote authentication. Therefore, it assures the correctness in the use of the Proxy applet and performed the biometric authentication.

The security analysis shows that the Java Card system containing the developed solution is secure, and that its requirements assure the safety of the user private data. The user biometric data used in the local biometric authentication is one of the major concerns during the authentication process, reason why an authenticated embedded device has to be used, capable of assuring a secure and exclusive environment to the biometric data.

In summary the developed solution is a secure remote authentication system, ensuring the presence of the user during the authentication by using biometric authentication. The assessment of the solution has positively demonstrated the functionality of the developed system in current authentication scenarios, and its reliability.

6 Conclusions

Contents

The growth of Internet usage increases the number of users using Electronic Service (e-service), which leads to the need for the development of more efficient security mechanisms, capable of assuring the protection of the user's information and identity, being at the same time adjusted to the needs of the market. In this sense, remote authentication mechanisms are becoming a very important issue, since they are a basic requirement on security systems, being used to identify people and, to assure that an entity is in fact who it claims to be.

The research on the authentication technology area has led to the need of developing a biometric authentication system, where the user is identified by something that is part of him, such as his fingerprint. This technology can be easily used by users, since they always carry their behavior and physical characteristics, and is also reliable, since it assures the presence of the user being authenticated. However, the biometric authentication is not used in current remote authentication systems, since it is not secure to send the biometric template of the user. Therefore, the goal of this thesis was to securely integrate the biometric authentication on existent remote authentication systems using Smart Cards with Personal Identification Number (PIN)s, through a solution easy to deploy and with low integration costs.

The proposed solution, the implemented prototype, their analysis, and the current state of the art were presented in the previews sections. The main conclusions of the work performed on this thesis is presented in this chapter. This chapter is divided in three sections. Section 6.1 presents an overview of the proposed solution, and the results obtained during the testing of the implemented prototype. Section 6.2 specifies the main contributions of the developed work, the convenience for the users, and the added security advantages provided to the current authentication systems. Future developments and a proposal for an important feature that should be assured on a large scale implementation of the proposed solution are described in the future work, presented in section 6.3.

6.1 Overview

The main goal of this thesis was to develop an authentication system capable of authenticating a remote user using his biometric signature. The authentication system was designed to be used in current systems, and to be deployed with existent technologies. The architecture of the proposed solution is based on the use of a Smart Card, already possessing an authentication module inside capable of authenticating the user remotely, using asymmetric cipher. Usually, the module responsible for the remote authentication assures the local authentication of the user through a PIN authentication, after which the user's private key is used to sign the challenge sent by the remote server.

The Smart Card also has to possess the ability to process the biometric template on the card itself with the Match-On-Card (MOC) operation, being able to perform a local biometric authentication of the user. Therefore, for a Smart Card to be used as the base of the proposed solution, the remote authentication module and the biometric authentication module have to combined, being independent between each other. The solution proposed on this thesis suggests the deployment of a Proxy module in the Smart Card. The Proxy module becomes the one responsible for managing the user authentication, inside the card, ensuring the release of the user private key in the remote authentication module, just after a successful biometric authentication of the user performed by the MOC module.

In order to prove the functionality of the proposed solution, a prototype was implemented following the defined architecture. The prototype was implemented over a solution based on Electronic Identity (eID) cards, as is the case of the Portuguese citizen card, to prove that it can be deployed in existent solutions. An Identification, Authentication, and Signature (IAS) and BioServer applets, corresponding to the authentication and MOC modules respectively, were developed, and deployed on a Java Card. After that, a Proxy applet was developed and loaded to the Java Card already containing the developed IAS and BioServer applet.

The test of the implemented prototype was performed using a web server, and a web browser, where the implemented Middleware was installed. The developed Middleware provides a Public-Key Cryptography Standards (PKCS#11) standards interface, and is responsible for providing the necessary abstraction of the communication with the Smart Card reader and biometric sensor. The environment test is composed by an embedded reader device, having the ability to perform the communication with the Java Card and the ability to capture the fingerprint of the user. The user authentication was performed during the user authentication phase of the Transport Layer Security (TLS) protocol, on the HyperText Transfer Protocol Secure (HTTPS) connection required by the web server. The web browser uses the developed Middleware to interact with the Java Card that is responsible for signing the challenge sent by the remote server, first ensuring the local biometric authentication of the user.

The tests performed on the developed prototype have shown that the implemented applets, installed on the Java Card, are secure and reliable, not being possible for a user to use someone elses Java Card, or to perform the remote authentication directly using the IAS applet. Therefore, the user is forced to communicate with the Proxy applet performing a local biometric authentication before the release of his private key.

However, for the secure functionality of the proposed system it is assumed that an embedded device, capable of managing the biometric authentication process with the Java Card, is used. Therefore, the capture of the fingerprint image and the generation of the biometric data used in the match process has to be performed on an embedded device. Furthermore, the embedded device has to send the biometric data generated, directly to the the Java Card of the user. This mechanism assures a protected and exclusive environment to the user biometric data, assuring that the used biometrics are always collected in the moment of the authentication, and that no external entity can gain access to it.

The implemented prototype has also shown that the proposed solution can be implemented using existent technology, and that it works with current authentication systems. However, if the proposed solution is intended to be deployed in a current implemented system, it has to be taken in consideration if the user is using a Smart Card with the Proxy already installed, or is still using a Smart Card that only has a PIN based authentication.

6.2 Contributions

The main contributions of the authentication solution presented on this thesis are summarized in the following sections. The added value of the proposed system is described in two perspective points, one from the user local authentication point of view, and the other from the remote server or service using the developed solution to authenticate the user.

User advantages during local authentication:

- The current remote authentication systems perform the local authentication of the user based on PIN. The proposed solution removes the need for the user to memorize any information such as his PIN, by performing the local authentication using his biometric signature. The user can never forget the necessary credentials required for the authentication system, since his local authentication is performed by something that is intrinsically related with him, and thus, always with him. This can be particularly useful.

- From a security perspective, the proposed solution is more efficient against theft, copy, or forgery of the user authentication credentials. The theft of the user biometrics is significantly more difficult than the PIN, making it more difficult for malicious users to get access to the credentials needed for the user authentication. Therefore, easy techniques such as the shoulder surfing cannot be used. The copy or counterfeiting of the users biometrics requires expertise and laboratory equipment, being harder to forge than the traditional PIN. The biometric template of the user, used to be matched with the previous stored template, is impossible to guess, and the use of an embedded device makes it even harder to obtain.

- The proposed solution is based on a mechanism that does not require any security enrolment on the server side, thus, it can be used to authenticate the user in a wide variety of servers and services. Furthermore, the user can always use the same credentials to be authenticated in all the systems being compatible with the proposed solution. Therefore, it is not necessary to have several authentication credentials, one for each service, making the interaction with the authentication system easier for all the various services that the user uses.

Remote server or service advantages:

- The remote services can ensure the presence of the user being authenticated by using the proposed authentication solution. The biometric traits of the user cannot be borrowed by anyone; thereby it is assured that no one besides the user can use its authentication credentials. This can be useful to prevent the repudiation of an authentication, or to assure that it is in fact the correct user being authenticated, providing an accurate control.
- The proposed system is easy to deploy in existent solutions, and can be used in current systems. Thereby, it takes a low cost and effort to adopt it on an authentication system being already in use, since it respects the requirements defined in the proposed solution in 3.2. Besides that, the majority of the current systems used for the remote authentication of the users can use the proposed solution, merely requiring the installation of the developed Middleware.
- The enrolment of the user biometric traits is performed into the Smart Card of the user, not being necessary to have a data base for each user, responsible for storing the user information necessary in the authentication. Thereby, the server just needs to trust in the Certification Authority (CA). Nevertheless, because the users do not need to constantly change the PIN, the maintenance of the authentication system is lower, reducing the need for constant alteration of the user's secret information.

6.3 Future work

The proposed solution defines the requirements needed for the development of a secure authentication system, using the user's biometrics. However, the protection of the biometric data, captured from the user to be matched inside the Smart Card, until it reaches the Smart Card, is an assumption of the proposal. The embedded device used during the test of the proposed solution does not generate internally the biometric data, requiring the Middleware to perform this computation. Therefore, the development of embedded device reader with the capacity to simultaneous capturing the image of the biometrics of the user being authenticated, and afterward generate the biometric data in a format that is prepared to be matched inside the Smart Card is suggested. Thus, it does not leave the reader unless it needs to be sent to the MOC module inside the user's Smart Card.

Bibliography

[1] Anil K. Jain, P. F. and Ross, A. A. (2007). Handbook of Biometrics. Springer, London, UK.

[2] Bistarelli, S., Santini, F., and Vaccarelli, A. (2006). An asymmetric fingerprint matching algorithm for java card tm. Pattern Analysis and Applications, 9(4):359–376.

[3] Cappelli, R., Ferrara, M., Franco, A., and Maltoni, D. (2007). Fingerprint verification competition 2006. Biometric Technology Today, 15(7-8):7–9.

[4] Chang, C.-C. and Wu, T.-C. (1991). Remote password authentication with smart cards. IEE Proceedings E Computers and Digital Techniques, 138(3):165.

[5] Chen, Z. (2004). Java Card Technology for Smart Cards: Architecture and Programmer's Guide. Addison-wesley, Boston, USA.

[6] Chien, H.-y. and Jan, J.-k. (2002). An Efficient and Practical Solution to Remote Authentication: Smart Card. Computers & Security, 21(4):372–375.

[7] Chu-Hsing Lin, Y.-Y. L. (2004). A flexible biometrics remote user authentication scheme. IEEE Computer Standards & Interfaces, 27(1):19–23.

[8] company, G. (2010). Portuguese Citizen's Card.

[9] Consortium, B. and Group, W. (2002). Java Card: Biometric API White Paper. Architecture, (August):02–0016.

[10] D, D., Gomes, B. G., Nova, L., and Brazil, N. R. N. (2006). Automation of Java Card component development using the B method. Engineering, 2.

[11] Dodis, Y., Reyzin, L., and Smith, A. (2008). Fuzzy Extractors : How to Generate Strong Keys from Biometrics and Other Noisy Data. Artificial Intelligence.

[12] Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4):469–472.

[13] Fodor, O. and Hassler, V. (1999). Javacard and opencard framework: a tutorial. IEEE Information Systems, pages 13–22.

[14] France-massey, T. (2005). Multos - the high security smart card os. Development, 44(September):0–4.

[15] French, H. P., Brennan, a., White, B., and Cusack, T. (2010). Manual therapy for osteoarthritis of the hip or knee - a systematic review. Manual therapy.

[16] Haller, N. (1994). The S/KEY One-Time Password System. Proceedings of the ISOC Symposium on Network and Distributed System Security.

[17] Hellman, E. (1976). New Directions in Cryptography. IEEE Trans. Inform. Theory IT 22, (6):644–654.

[18] Husemann, D. (2001). Standards in the smart card world. Computer Networks, 36(4):473–487.

[19] Jain, A., Pankanti, S., and Bolle, R. (1997). An identity-authentication system using fingerprints. Proceedings of the IEEE, 85(9):1365–1388.

[20] Jan, J.-k. and Chen, Y.-y. (1998). Paramita wisdom: password authentication scheme without verication tables. Journal of Systems and Software, 42:45–57.

[21] Lamport, L. (1981). Password authentication with insecure communication. Communications of the ACM, 24(11):770–772.

[22] Lee, J., Ryu, S., and Yoo, K. (2002). Fingerprint-based remote user authentication scheme using smart cards. IEEE Electronics Letters, 38(12):554.

[23] Li, C.-T. and Hwang, M.-S. (2010). An efficient biometrics-based remote user authentication scheme using smart cards. Journal of Network and Computer Applications, 33(1):1–5.

[24] Magalhães, P. S. (2003). Biometria e autenticação resumo.

[25] Maltoni, D., Maio, D., Jain, A. K., and Prabhakar, S. (2009). Handbook of Fingerprint Recognition. Springer, London, UK, 2 edition.

[26] Mayes, K. E. and Markantonakis, K. (2008). Smart Cards, Tokens, Security and Applications. Springer, London, UK.

[27] Mostowski, W. (2002). Java card tools for together control center. Development, pages 1–9.

[28] Mpcos, G. (2003). Mpcos simple , flexible , proven and reliable technology for multi-applications.

[29] M.S. Hwang, L. L. (2000). A new Remote User Authentication Scheme using Smart Cards. IEEE Transactions on Consumer Electronics, pages 1–3.

[30] O'Gorman, L. (2003). Comparing passwords, tokens, and biometrics for user authentication. Proceedings of the IEEE, 91(12):2019–2020.

[31] Ortiz, C. E. (cite Dec 2010). An introduction to java card technology.

[32] Perlman, R. and Microsystems, S. (1999). An Overview of PKI Trust Models. Ieee Network, (December):38–43.

[33] Prabhakar, S. (2003). Biometric recognition: security and privacy concerns. IEEE Security, 4677(2):275–42.

[34] Rankl, W. (2007). Smart Cards Applications: Design models for using and programming smart cards. Wiley, West Sussex PO19 8SQ, England.

[35] Rankl, W. and Effing, W. (2003). Smart Card: Handbook. Wiley, West Sussex PO19 8SQ, England, 3 edition.

[Republic] Republic, C. BIOMETRIC AUTHENTICATION: SECURITY AND USABILITY. pages 1–13.

[37] Rivest, R. L., Shamir, a., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126.

[38] Selimis, G., Fournaris, A., Kostopoulos, G., and Koufopavlou, O. (2009). Software and hardware issues in smart card technology. IEEE Communications Surveys & Tutorials, 11(3):143–152.

[39] T. Hwang, Y. Chen, C. L. (1990). NON-INTERACTIVE PASSWORD AUTHENTICATIONS WITHOUT PASSWORD TABLES. Information and Software Technology, 32(3):228.

[40] Terán, L. and Drygajlo, A. (2009). On development of inspection system for biometric passports using java. pages 260–267, Berlin, Heidelberg. Springer-Verlag.

[41] Tsujii, S. and Itoh, T. (1989). An ID-based cryptosystem based on the discrete logarithm problem. IEEE Journal on Selected Areas in Communications, 7(4):467–473.

[42] Uludag, U. (2004). Biometric cryptosystems: issues and challenges. Proceedings of the IEEE, 42(8):136–960.

[43] Usenix, F., Conference, N., Deville, D., Galland, A., and Grimaud, G. (2003). Smart card operating systems : Past , present and future. pages 1–18.

[44] Warnier, M. (2003). Java card remote method invocation.

[45] Zúquete, A. (2008). Segurança em Redes Informáticas. FCA, 2 edition.

A

BioServer Applet Manual

This manual is intended to detail the development of the BioServer applet. The motivation for the development of this applet, as well as the internal structure and Application Programming Interface (API) that it provides are described.

The BioServer applet was developed in the context of the implemented prototype for the proposed solution. This applet is an implementation of the Match-On-Card (MOC) module inside the Smart Card layer of the proposed solution 3.2.1. The main purpose of the BioServer applet is to manage the biometrics of the Java Card holder, allowing the user to perform the enrolment of his biometric template in the Java Card, and providing an internal API for the Proxy applet to be able to perform the user biometric authentication. The Java Card used during the implementation of the prototype already had a Java Card applet able to execute the same functionalities that the BioServer applet provides. However, the private API provided by this applet is not the generic Java Card Biometric Application Programming Interface (JC-BioAPI), reason why this new applet was developed.

BioServer applet is a Java Card applet, developed using the Java Card Development Kit (JCDK) for the Java Card version 2.2.1. This applet simultaneously provides and takes advantage of the JC-BioAPI. On the other hand it uses this API to interact with the BioMatch library developed by *Presice Biometrics*, present in the Java Card, and at the same time provides it as an internal API being used by internal applets, such as the Proxy applet.

Its internal structure is divided into two sections. The first section, BioServer API, is responsible for the communication with the outside of the Java Card and with the other internal applets. The other section, BioServer implementation, takes care of the interaction with the BioMatch library, being responsible for keeping the state and managing the biometric authentication for each biometric template enrolled on the Java Card. Figure A.1 depicts the internal structure of the BioServer applet.

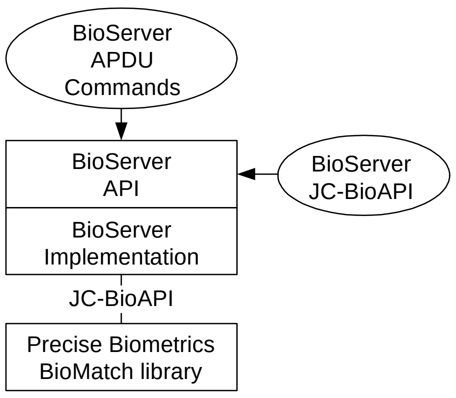


Figure A.1: Internal structure of the BioServer.

The communication with the BioServer applet can be performed from outside the Java Card, or internally by other applets sharing the Java Card. The internal communication with other applets

Commands Description	CLA	INS	P1	P2
Enroll biometric template: Init	’B0’	’30’	Template ID	’00’
Enroll biometric template: Update	’B0’	’30’	Template ID	’01’
Enroll biometric template: Final	’B0’	’30’	Template ID	’02’
Verify biometric data: Init	’B0’	’32’	Template ID	’00’
Verify biometric data: Update	’B0’	’32’	Template ID	’01’
Verify biometric data: Final	’B0’	’32’	Template ID	’02’
Get information about the biometric template	’B0’	’34’	Template ID	’00’
Delete biometric template	’B0’	’36’	Template ID	’00’
Select Applet	’00’	’A4’	’04’	’00’

Table A.1: External API provided by the BioServer applet.

is performed using the JC-BioAPI, specified in the [42]. The API provided for the communication with the external world is composed by the methods shown on table A.1.

Any internal applet, as is the case of the Proxy applet, can communicate with this applet using the generic and public known JC-BioAPI. Despite the use of the fingerprint the generic architecture of this applet covers a wide variety of biometrics, specifically all the supported by the JC-BioAPI.

B

IAS Applet Manual

This manual is intended to detail the development of the Identification, Authentication, and Signature (IAS) applet. The motivation for the development of this applet, as well as the internal structure and Application Programming Interface (API) that it provides are described.

The IAS applet was developed in the context of the implemented prototype for the proposed solution. This applet is an implementation of the authentication module inside the Smart Card layer of the proposed solution 3.2.1.The main purpose of this applet is to store the personal information of the user such as his name, id, photo, or address. It is also responsible for performing all the necessary operations for the user remote and local authentication, securely storing his private key and biometric template. The implementation of this applet followed the Identification Authentification Signature - European Citizen Card (IAS-ECC) standard. However, only the functions needed to develop the prototype of the proposed solution were implemented. Furthermore, this applet enables the communication with other applets in the Java Card by providing an internal API, known by the Proxy applet.

The IAS applet is divided in three core sections, the API, the Command Implementation, and the File System.The API section is responsible for allowing the communication between the IAS applet and the outside of the Java Card, as well as with the Proxy applet inside the Java Card. The implementation section is responsible for the internal processing of the arrived Application Protocol Data Unit (APDU) commands. It performs the execution of all the functionalities provided by the IAS applet, covering the implementation of the user authentication and the management of his personal information, as well as the execution ofall the cryptographic operations required. Furthermore, the implementation section is also responsible for the management of the applet states. The IAS applet has three states, *SELECTABLE* while it is in the personalization phase, *PERSONALIZED* after being personalized with the user information, and *BLOCKED* if the Java Card is blocked by an external command or because the user exceeded the limit of wrong authentication tries.

The security logic of the IAS applet, regarding the sensitive information stored by the applet, and the necessary level of authentication in each access is managed by the File System section. The internal structure of the IAS applet is depicted in Figure B.1, where it is even possible to see the section responsible for the management of the constant variables shared among each section. The File System implemented in the IAS applet is also divided in the binary file system and in the object data file system. The binary file system follows the International Organization for Standardization / International Electrotechnical Commission (ISO/IEC)7816-4 standard, and is responsible for managing the access to all the binary files keeping the user information, such as name, id and the public certificate. These files are kept in Elementary File (EF) distributed by the Dedicated File (DF). The object data file system stores the sensitive information such as the user Personal Identification Number (PIN) and private key.

Communication with the IAS applet can be performed from outside the Java Card, or internally

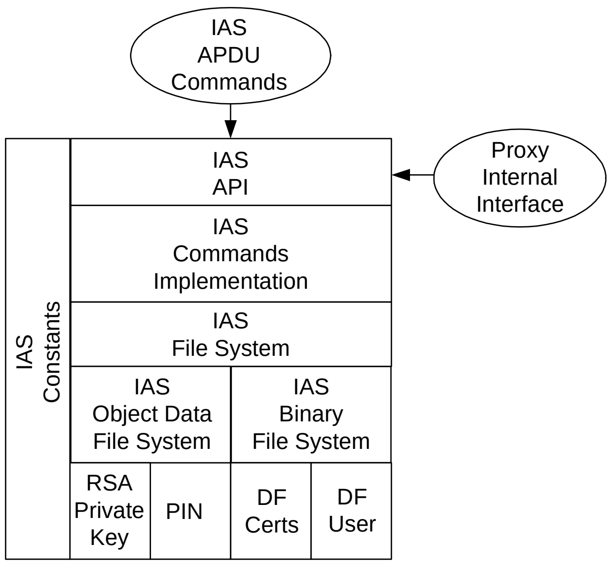


Figure B.1: Internal structure of the IAS.

Commands description	CLA	INS	P1	P2
VERIFY	’00’	’20’	’00’	pinRef
SELECT	’00’	’A4’	Data type	RetType
PUT DATA	’00’	’DB’	’00’	’FF’
CHANGE REFERENCE DATA	’00’	’24’	’00’	pinRef
RESET RETRY COUNTER	’00’	’2C’	Mode	pinRef
GET DATA TLV	’00’	’CB’	’00’	’FF’
CREATE FILE	’00’	’E0’	’00’	’00’
UPDATE BINARY	’00’	’D6’	P1	P2
READ BINARY	’00’	’B0’	P1	P2
ERASE BINARY	’00’	’0E’	P1	P2
PSO - Compute Digital Signature	’00’	’2A’	’9E’	’9A’
PSO - Hash	’00’	’2A’	’90’	’80’/’A0’

Table B.1: External API provided by the IAS applet.

by the Proxy applet. The internal communication with the Proxy applet is performed using the Shareable Interface Object (SIO), which respective API can be found at C. The external interface of the IAS applet is supported by the commands shown in the table B.1, all of them specified in the ISO/IEC7816. The other provided commands are specific from the proprietary IAS-ECC implementation of *Gemalto*, and as such will not be presented here.

C

Proxy Applet Manual

This manual is intended to detail the development of the Proxy applet. The motivation for the development of this applet, as well as the internal structure and Application Programming Interface (API) that it provides are described.

The Proxy applet was developed in the context of the implemented prototype for the proposed solution. This applet is an implementation of the proxy module inside the Smart Card layer of the proposed solution 3.2.1. The main purpose of the Proxy applet is to manage all the operations related with the user authentication. Therefore, during the remote authentication of the user, the Proxy applet redirects all the requests either to the Identification, Authentication, and Signature (IAS) or to the Bioserver applet. The challenge sent by the remote server has to be signed by the IAS applet, using the private key of the user. To use the private key of the user, a local authentication of the user using his Personal Identification Number (PIN) is required by the IAS applet. The Proxy applet ensures the release of the user private key in the IAS applet, inserting the user PIN, but only after a successful biometric authentication of the user is performed in the BioServer applet.

This applet is designed to be deployed in an existent system, taking full control of the user authentication in the IAS, and being the front end of the Java Card. To take control of the user PIN authentication on the IAS applet, the Proxy changes it when it is deployed to the Java Card. To perform this change, and the subsequent user PIN authentication in the IAS applet, the Proxy applet defines an API that has to be implemented by the authentication modules, such as the IAS applet. The Proxy applet represents the innovation module brought by the proposed solution on this thesis.

The Proxy applet is a Java Card applet, developed using the Java Card Development Kit (JCDK) for the Java Card version 2.2.1. This applet uses two API, one to interact with the BioServer applet, and the other one used for the interaction with the IAS applet. Furthermore, this applet provides one API to the external communication with the Java Card, covering all the functionalities and operations supported by the system. The structure of the Proxy applet is depicted in the Figure C.1.

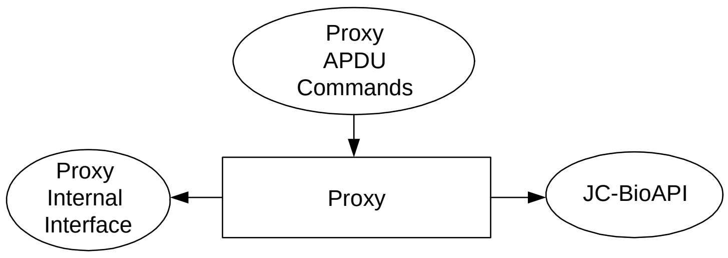


Figure C.1: Internal structure of the Proxy.

The Java Card Biometric Application Programming Interface (JC-BioAPI) is used to communicate with the BioServer applet, while the communication with the IAS applet is assured by a Shareable Interface Object (SIO) provided by the IAS applet. This SIO implements an API defined by the Proxy internal interface. The definition of the Proxy internal interface is described in the following:

```
public interface IASECCInterface extends Shareable {

        /**
         * Allows the Proxy to call the process method of IAS applet
         */
        public void process(APDU apdu);

        /**
         * Allows the Proxy to call the select method of the IAS applet
         */
        public void select(APDU apdu);

        /**
         * Allows the Proxy to perform an authentication of the user
         * in the IAS applet, using his PIN
         */
        public void verify(APDU apdu);

        /**
         * Allows the Proxy to change the PIN of the user in the
         * IAS applet
         */
        public void changeReferenceData(APDU apdu);
}
```

This interface is composed by the necessary methods for the Proxy applet to interact with the IAS applet, assuring all the requirements of the system. The *process* method is used to redirect normal requests to the IAS applet. Therefore, it is used when a command received by the Proxy applet is neither from it nor the BioServer applet. Since the *process* method is the entry function of all commands, the IAS applet may not even realize that the command was not received directly by an external source of the Java Card. The *select* command is used to select the IAS applet. This is necessary since the *select* command can arrive at the Proxy applet with different destinations, being necessary for the Proxy to request its data. Once requesting the data of a command, the Proxy cannot send it to the IAS applet over the *process* command as it was directly sent to it. The *verify* command is used by the Proxy applet for the authentication of the user in the IAS applet. This authentication is performed using the PIN of the user, stored safely by the Proxy applet. The change of the PIN of the user in the IAS applet is performed by the Proxy applet using of the *changeReferenceData* command.

D

Load Applets in a Java Card

The purpose of this appendix is to describe the load process of an applet into a Java Card, summarizing the technologies used and the steps performed. The Java Card platform does not specify any secure mechanisms for managing the load and deletion of the applets inside a Java Card. Therefore, the *Global Platform* specifications are used in most of the existent Java Card systems, to manage the load of the applets into the cards. In the development of the prototype of the proposed solution three applets were developed, the Identification, Authentication, and Signature (IAS), the Proxy, and the BioServer. These applets were loaded into the Java Card using an application developed according to the *Global Platform* specifications.

The Java Card applet code is developed in a .java file. To load the code into the Java Card it needs to first be compiled using the tools provided by the Java Card Development Kit (JCDK) into Converted Applet (CAP) files. If any external interface has to be imported by the applet, such as is the case of the Java Card Biometric Application Programming Interface (JC-BioAPI) in the Proxy and BioServer applets, an .exe file containing this external interface has to also be provided to the JCDK during the compilation. In addition to the CAP file, the compilation generates a .exe file, containing the definition of the interfaces provided by the applet compiled.

The CAP files contain the byte-code ready to be loaded over Application Protocol Data Unit (APDU) commands into the Java Card, and installed afterwards on its the Java Card Runtime Environment (JCRE).The sequence of the commands, respecting the security specifications of the *Global Platform*, were generated and sent to the Java Card by the *Global Platform Shell* version1.4.4.The main components used in the development and load of a Java Card Applet are depicted in the Figure D.1.

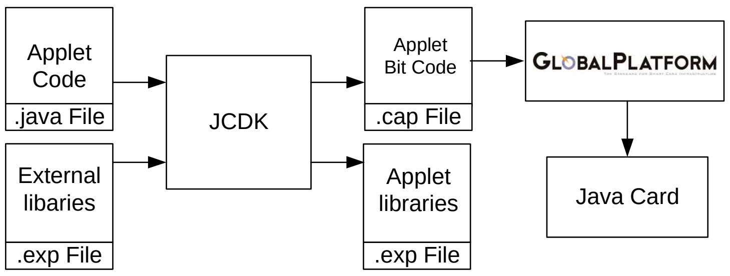


Figure D.1: Scheme for loading an applet into a Java Card.

The information specifying the Secure Channel Password (SCP) used, and the location of the CAP files are inserted in the Global platform shell in the form of a script file. An example of a script file is summarized next:

```
///////////////////////////////////////
// Script GlobalPaltform //
///////////////////////////////////////

//
//      Init
//
mode_211
enable_trace
establish_context
card_connect

//
//      Select Card Manager
//
select –AID A000000018434D00

//
// Open secure channel
//
open_sc –security 1 –scp 1 –scpimpl 5 –keyind 0 –keyver 0
        –keyDerivation visa2 –key <KEY>

//
//      List commands
//
//get_status –element e0 // List applets and packages and security domains
//get_status –element 20 // List packages
//get_status –element 40 // List applets or security domains
//get_status –element 80 // List Card Manager / Security Issuer Domain

//
//      IAS
//
install –file applets/ias.cap –nvDataLimit 500 –priv 2
        –nvCodeLimit 4000 –pkgAID 010101010102

//
//      Disconnect
//
card_disconnect
release_context
```

E

PKCS#11 Implementation List

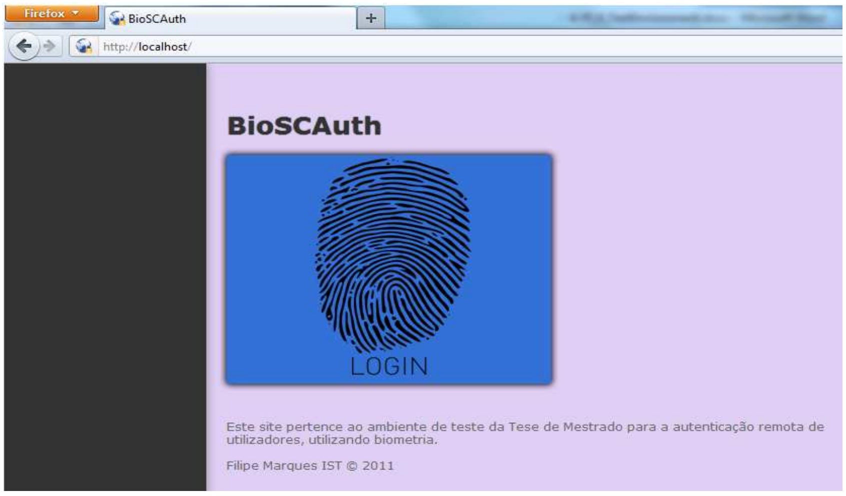
Category	Function	Description
General purpose functions	C_Initialize	Initializes the PKCS#11 library.
	C_Finalize	Signals that the application is through with the PKCS#11 library.
	C_GetInfo	Returns manufacturer and version information about the cryptoki library.
	C_GetFunctionList	Returns all functions of PKCS#11 interface;
Slot and token management functions	C_GetSlotList	Returns a list of available slots.
	C_GetSlotInfo	Obtains information about a particular slot in the system.
	C_GetTokenInfo	Gets information about a specific token.
	C_WaitForSlotEvent	Wait for a slot event. (only non-blocking)
	C_GetMechanismList	Gets a list of mechanism types that are supported by the specified token.
	C_GetMechanismInfo	Obtains information about a particular mechanism.
Session management functions	C_OpenSession	Enables an application to start a cryptographic session with a specific token in a specific slot.
	C_CloseSession	Close a cryptographic session with a specific token.
	C_CloseAllSessions	Close all cryptographic sessions of a slot.
	C_GetSessionInfo	Obtains information about a particular session.
Object management functions	C_GetAttributeValue	Obtains the value of one or more object attributes.
	C_FindObjectsInit	Initialize a search for an object.
	C_FindObjects	Return the search result.
	C_FindObjectsFinal	Finalize the search and clean memory.
Signing and MACing functions	C_SignInit	Initializes a digital signature operation. (private key encryption)
	C_Sign	Encrypts with private key, data in a single part.
	C_SignUpdate	Continues a multiple-part signature operation.
	C_SignFinal	Finishes a multiple-part signature operation, returning the signature.
Random number generation functions	C_SeedRandom	Generate a seed.
	C_GenerateRandom	Generate a random sequance of bytes.

F

Test Environment

The purpose of this appendix is to illustrate the results obtained during the test of the proposed solution. It is shown in the print screens of the developed web site, illustrating the interaction of the user with the authentication system.

Figure F.1 depicts the first page of the web site, that can be accessed by a HyperText Transfer Protocol (HTTP) connection. The login button represents the link for the user page, only accessed by an HyperText Transfer Protocol Secure (HTTPS) connection.

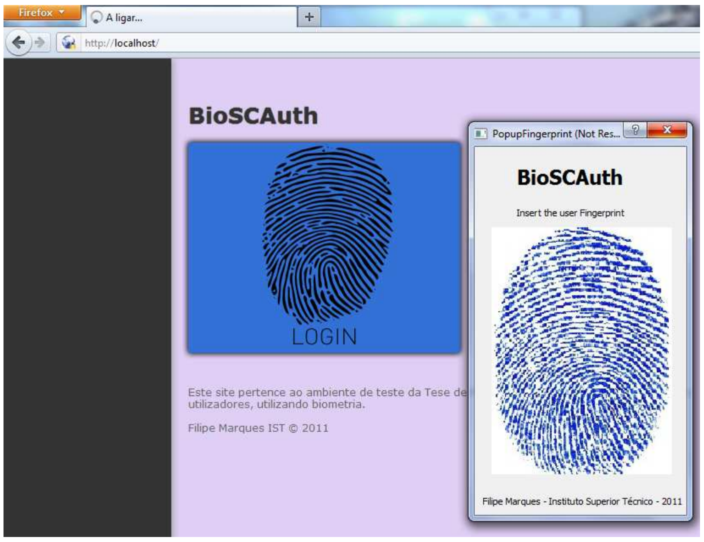


Este site pertence ao ambiente de teste da Tese de Mestrado para a autenticação remota de utilizadores, utilizando biometria.

Filipe Marques IST © 2011

Figure F.1: Login web page.

During the establishment of the HTTPS connection the Java Card requests the user biometric authentication to the PKCS#11 Middleware. The PKCS#11 prompts a window requesting the user to put his finger over the biometric sensor. The prompt window can be seen in Figure F.2.



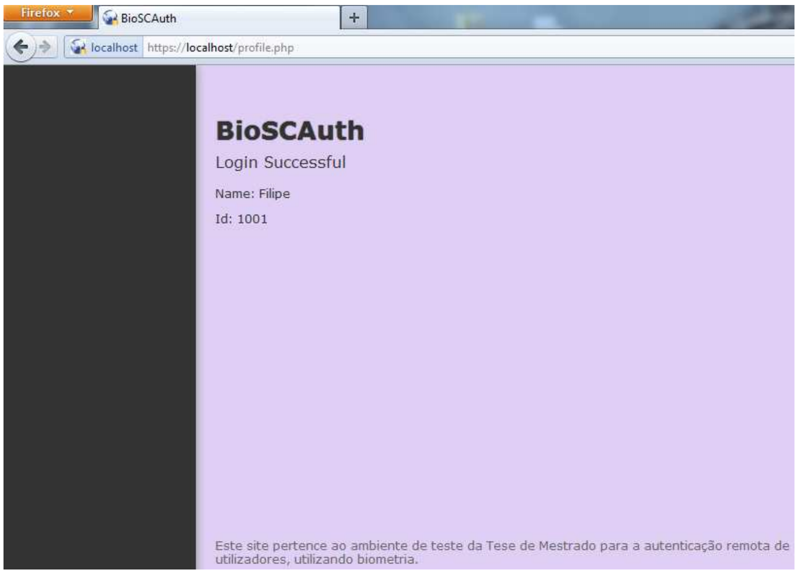
Este site pertence ao ambiente de teste da Tese de
utilizadores, utilizando biometria.

Filipe Marques IST © 2011

Filipe Marques - Instituto Superior Técnico - 2011

Figure F.2: Requesting the user fingerprint.

The Middleware captures the fingerprint biometric template through the biometric sensor, and sends it to the proxy applet into the Java Card, in order to process the user biometric authentication. After a successful biometric authentication, the Identification, Authentication, and Signature (IAS) applet is able to process the signature of the challenge sent by the web server using the user private key. The result of this signature is passed to the browser that will finish the Transport Layer Security (TLS) handshake with the web server. If the HTTPS connection is established successfully, the user page is filled using Hypertext Preprocessor (PHP) calls, where the username and id are collected from the user public certificated by the web server. Figure F.3 shows the user page, after a successful authentication. If the TLS authentication fails, the web browser displays an error page informing the user that the HTTPS connection cannot be correctly established due to an error in the TLS handshake.



BioSCAuth

Login Successful

Name: Filipe

Id: 1001

Este site pertence ao ambiente de teste da Tese de Mestrado para a autenticação remota de utilizadores, utilizando biometria.

Figure F.3: User web page.