# Reliable electronic certification on mobile devices

**Nuno Alvarez Fernandes**

Thesis to obtain the Master of Science Degree in

## Telecommunications and Informatics Engineering

Supervisors: Prof. Ricardo Jorge Fernandes Chaves

## Examination Committee

Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisor: Prof. Ricardo Jorge Fernandes Chaves
Members of the Committee: Prof. Miguel Filipe Leitão Pardal

**November 2015**

# Acknowledgments

The work developed in this thesis would not have been possible without the help of several people.

I would like to give a big thank you to my supervisor, Professor Ricardo Chaves, for all his support. His insight, knowledge, and patience provided me the proper guidance to achieve this work. My thanks also to Francisco Loureiro from PDM&FC for his precious inputs and sharing of knowledge.

I would also like to thank my parents, especially my mother for her understanding, friendship and caring over all these years (not just the academic ones). Also her availability in loaning her car has to be acknowledged, as it proved to be very useful on many nights.

I would like to thank my girlfriend, Inês Fernandes, for her encouragement and caring throughout these past years, and for always being there for me during the good and bad times.

I would also like to acknowledge Kevin Lee from GO-Trust Technology Inc. for his help during the initial phase of the implementation of this work.

Last but not least, I would like to thank all who participated in the evaluation of this thesis by performing the usability tests.

To each and every one of you – Thank you.

# Abstract

Nowadays many documents are still signed in a handwritten way, being highly susceptible to forgery. Digital signatures address this vulnerability by providing a cryptographically secure way to do it. They provide a secure and reliable way to sign digital documents, thereby improving the security of the three key services stipulated by the handwritten signatures: i) Authentication: the signer is who he or she claims to be; ii) Integrity: the data has not been modified or tampered with since the signature was applied; iii) Non-repudiation: an irrefutable proof of signature. Furthermore, this type of signatures can also be performed remotely.

With the exponential growth in the use of mobile devices in everyday life, there is an increasing availability of mobile technologies, giving rise to new applications that take advantage of such devices to improve the way users perform their daily tasks. The work herein proposed aims to facilitate the signing process of digital documents on mobile devices by creating a viable and trusted certification system that uses mobile devices, eliminating the need for external readers, and increasing the users' flexibility. Specifically, it consists of a simple and intuitive mobile application that enables users to digitally sign electronic documents on their devices, using a private signature key stored in a smart card (in this case boxed on a micro Secure Digital (SD) card) inserted in the device, thus allowing to provide qualified digital signatures. All private material can be transferred from one device to another simply by moving the secure micro SD card. Thus, by using a hardware-based security technique, the developed solution provides a protected environment for the user credentials which is never exposed, and cannot be compromised.

# Keywords

Security; Qualified Digital Signatures; Mobility; Smart Cards.

# Resumo

Hoje em dia muitos documentos ainda são assinados de forma manuscrita, estando bastante sujeitos à falsificação. As assinaturas digitais lidam com esta vulnerabilidade, proporcionando uma maneira criptográfica segura de o fazer. Elas oferecem uma forma segura e confiável de assinar documentos digitais, melhorando assim a segurança dos três principais serviços estipulados pelas assinaturas manuscritas: i) Autenticação: o assinante é quem afirma ser; ii) Integridade: os dados não foram modificados ou alterados desde que a assinatura foi aplicada; iii) Não-repudiação: uma prova irrefutável da sua assinatura. Para além disto, este tipo de assinaturas também pode ser realizado remotamente.

Com o crescimento exponencial no uso de dispositivos móveis na vida quotidiana, há uma disponibilidade cada vez maior das tecnologias móveis, dando origem a novas aplicações que tiram partido de tais dispositivos para melhorar a forma como os utilizadores realizam as suas tarefas diárias. O trabalho aqui proposto visa facilitar o processo de assinatura de documentos digitais em dispositivos móveis, criando um sistema de certificação viável e confiável que usa dispositivos móveis, eliminando a necessidade de leitores externos, e aumentando a flexibilidade dos utilizadores. Concretamente, o sistema consiste numa aplicação móvel simples e intuitiva que permite aos utilizadores assinarem digitalmente documentos eletrónicos nos seus dispositivos, usando uma chave de assinatura privada armazenada num cartão inteligente (neste caso encaixotado num cartão micro SD) inserido no dispositivo, permitindo assim fornecer assinaturas digitais qualificadas. Todo o material privado pode ser transferido de um dispositivo para outro movendo simplesmente o cartão micro SD seguro. Desta forma, usando uma técnica de segurança baseada em *hardware*, a solução desenvolvida fornece um ambiente

protegido para as credenciais do utilizador que nunca é exposto, e que não pode ser comprometido.

# Palavras Chave

Segurança; Assinaturas Digitais Qualificadas; Mobilidade; Cartões Inteligentes;

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **AP** | Application Provider |
| **APDU** | Application Protocol Data Unit |
| **API** | Application Programming Interface |
| **ASiC** | Associated Signature Containers |
| **ASiC-E** | Extended |
| **ASN.1** | Abstract Syntax Notation One |
| **BCA** | Biometric Certification Authority |
| **BER** | Basic Encoding Rules |
| **CA** | Certificate Authority |
| **CC** | Common Criteria |
| **CAdES** | CMS Advanced Electronic Signatures |
| **CAdES-A** | CAdES Archiving validation data |
| **CAdES-C** | CAdES Complete validation data |
| **CAdES-T** | CAdES Timestamp |
| **CAdES-X** | CAdES eXtended validation data |
| **CAdES-X-L** | CAdES eXtended validation data incorporated for the Long term |
| **CAP** | Converted Applet |
| **CMS** | Cryptographic Message Syntax |
| **CPU** | Central Processing Unit |

| | |
|---|---|
| **DER** | Distinguished Encoding Rules |
| **DLL** | Dynamic Link Library |
| **DSA** | Digital Signature Algorithm |
| **EAL** | Evaluation Assurance Level |
| **ECDSA** | Elliptic Curve DSA |
| **EEPROM** | Electrically Erasable Programmable Read-Only Memory |
| **ETSI** | European Telecommunications Standards Institute |
| **EU** | European Union |
| **GB** | Gigabyte |
| **GHz** | Gigahertz |
| **GPRS** | General Packet Radio Service |
| **GSM** | Global System for Mobile communication |
| **GUI** | Graphical User Interface |
| **HTML** | HyperText Markup Language |
| **HTTP** | Hypertext Transfer Protocol |
| **IAIK** | Institute for Applied Information Processing and Communication |
| **ICC** | Integrated Circuit Card |
| **IEC** | International Electrotechnical Commission |
| **IETF** | Internet Engineering Task Force |
| **iMSSP** | MNO-independent MSSP |
| **IP** | Internet Protocol |
| **ISO** | International Organization for Standardization |
| **IT** | Information Technology |
| **Java ME** | Java Platform Micro Edition |
| **JCRMI** | Java Card Remote Method Invocation |

| | |
|---|---|
| **JCVM** | Java Card Virtual Machine |
| **JSR** | Java Specification Request |
| **kB** | Kilobyte |
| **LTV** | Long-Term Validation |
| **MASP** | Application/Service Providers |
| **MB** | Megabytes |
| **MNO** | Mobile Network Operator |
| **MNO-iMSS** | Mobile Network Operator-independent MSS |
| **MOS** | Mobile Operating System |
| **MSAU** | Mobile Signature Application Unit |
| **MSS** | Mobile Signature Service |
| **MSSP** | Mobile Signature Service Provider |
| **m-signature** | mobile signature |
| **NFC** | Near Field Communication |
| **OO** | Object Oriented |
| **OS** | Operating System |
| **OTA** | Over The Air |
| **PAdES** | PDF Advanced Electronic Signatures |
| **PDF** | Portable Document Format |
| **PIN** | Personal Identification Number |
| **PUK** | Personal Unblocking Key |
| **PKCS** | Public-Key Cryptography Standards |
| **PKI** | Public-Key Infrastructure |
| **PKIX** | Public Key Infrastructure X.509 |
| **RA** | Registration Authority |

| | |
|---|---|
| **RAM** | Random-Access Memory |
| **ROM** | Read-Only Memory |
| **RSA** | Rivest-Shamir-Adleman |
| **SAT** | SIM Application Toolkit |
| **SATSA** | Security And Trust Services API |
| **SD** | Secure Digital |
| **SE** | Standard Edition |
| **SIM** | Subscriber Identity Module |
| **SIP** | Session Initiation Protocol |
| **SMS** | Short Message Service |
| **SOAP** | Simple Object Access Protocol |
| **SPMS** | Self-Proxy Mobile Signature |
| **SSCD** | Secure Signature Creation Device |
| **S/MIME** | Secure / Multipurpose Internet Mail Extensions |
| **TEE** | Trusted Execution Environment |
| **TLS** | Transport Layer Security |
| **TS** | Technical Standard |
| **UICC** | Universal Integrated Circuit Card |
| **UMTS** | Universal Mobile Telecommunications System |
| **USAT** | Universal SIM Application Toolkit |
| **USAT-I** | USIM Application Toolkit Interpreter |
| **USIM** | Universal Subscriber Identity Module |
| **WAP** | Wireless Application Protocol |
| **WIB** | Wireless Internet Browser |
| **WIM** | WAP Identity Module |

| | |
|---|---|
| **WML** | Wireless Markup Language |
| **W3C** | World Wide Web Consortium |
| **XAdES** | XML Advanced Electronic Signatures |
| **XAdES-T** | XAdES Timestamp |
| **XAdES-C** | XAdES Complete |
| **XAdES-X** | XAdES eXtended |
| **XAdES-X-L** | XAdES eXtended Longterm |
| **XAdES-A** | XAdES Archiving |
| **XHTML** | Extensible HyperText Markup Language |
| **XML** | Extensible Markup Language |
| **XMLDSig** | Extensible Markup Language Signature |
| **XML-RPC** | XML Remote Procedure Call |
| **XSECT** | IAIK XAdES add-on for XML Security Toolkit |
| **3FF** | 3rd Form Factor |
| **3GPP** | 3rd Generation Partnership Project |
| **4FF** | 4th Form Factor |

# 1

# Introduction

## Contents

## 1.1 Motivation

Throughout our lives we sign our name countless times on many different documents with legal value, such as commercial contracts, loan applications, checks, etc. Since ancient times, signatures have been employed pretty much the same way – by writing one's name or using stamps/seals. Handwritten signatures have been generally accepted as a means that offers sufficient certainty about the signer's identity (authentication) for many documents and transactions. Additionally, they also provide non-repudiation, proving to third-parties that a particular party participated in a transaction, and thus protecting other parties involved in the transaction from false denials of participation by the signer. Although handwritten signatures do not in themselves provide data integrity, the security practices surrounding them, like tamper-evident paper or indelible ink, offer some level of data integrity. The strength of this type of signatures is hard to determine. They can have different levels of confidence, depending on the technical expertise of the person verifying the signature.

Over the years, many cases of forgery have occurred. However, handwritten signatures are still very much used, as they generally provide a strength of security services sufficient for the purposes to which they are applied. When stronger security mechanisms are required, such as when entering into treaty agreements, notarized, witnessed signatures are used. Digital signatures provide a way to perform this process digitally. While serving the same purpose as a handwritten signature, a digital signature uses digital keys (asymmetric cryptography) instead of using pen and paper, strengthening the three security properties previously mentioned – authentication, data integrity and non-repudiation. When properly implemented, digital signatures are more difficult to forge than the handwritten ones. With the continuous evolution of technology, digital signatures have become an essential requirement in electronic business. When produced on a mobile phone or a SIM card contained within that device, these signatures are called mobile signatures. If properly created, they can have the legal equivalent of handwritten signatures, according to the European Union (EU) directive for electronic signatures [10]. Nonetheless, handwritten signatures remain a fast, cheap and easily understood signing method, despite their vulnerabilities.

With more and more users using mobile devices, mobile computing has become one of the main sources of processing, through which most users accomplish their daily electronic tasks. As a result of this "mobile wave", consumers are constantly demanding mobile solutions, and companies are changing the way they do their business, including mobile technology in their solutions to reach the largest number of consumers. To clarify, mobile devices can be roughly defined as small computing devices with an Operating System (OS) capable of running several types of application software, and that can be carried around by users without a significant effort. This includes devices such as laptops, smartphones, tablet computers (these three being the most popular ones), and others. Thus, to be able to produce mobile digital signatures, such type of devices is required.

There are several different solutions to provide digital signatures on mobile devices, based on asymmetric cryptography and capable of providing qualified signatures. The word 'qualified' states that the signatures are generated only by a security token issued by a trusted certification entity. The private signature key never leaves this token, thus ensuring that the generated signatures can only be produced by who holds the token.

The existing solutions are mainly used in mobile electronic commerce. There are those that only make use of the SIM card of the mobile phones. Others use both the SIM card and the middleware of the devices to aid in the execution of the signature functions, and there are also those that use the SIM card and high level services, which do not depend on the mobile device technologies, to provide digital signatures. However, all these solutions use the SIM card as the security element that stores the sensitive data used in the signing processes. Despite being a secure smart card capable of providing qualified digital signatures, the SIM's main problem lies in the fact that it is dependent on the telecommunications operator, which means that the user would need to have a SIM card with a usable cryptographic token, which is hard to find.

There are other ways, besides using a smart card, to secure the sensitive data on mobile devices. The semiconductor firm ARM has a security technology called TrustZone [11], which is a feature of the processor architecture that provides a Trusted Execution Environment (TEE), by "hardware-separating" a "secure world" from a "normal world". The first handles all security-related functions and hosts all secure applications, while the normal world handles everything else. This means that the underlying hardware of the ARM processor enforces a separation of data that is tagged as secure from data that is not. Consequently, an attacker that manages to obtain full root privileges in the normal OS cannot access data that is tagged as secure. Therefore, with TrustZone the sensitive data, like the private signature key, can be created inside the tightly-constrained secure OS of the mobile device's processor, and the signature process can be performed within a secure environment, outside the reach of the normal OS. However, TrustZone in itself does not provide any mechanism to implement secure storage, which means that it is able to create data, like the signature key, but not store it. Still, TrustZone has a particular feature that enables the assignment of a specific peripheral to be accessed only by the secure world, but is has to be the chip vendors or the TEE developers to decide which peripheral is used as the secure storage. Some security architectures have been proposed that take advantage of TrustZone to develop security-sensitive applications or trusted applications on mobile devices [12] [13]. However, these architectures require the development of specialized systems to be able to use ARM TrustZone.

As previously stated, digital signatures are greatly needed nowadays. To offer a higher mobility to the signature process, a viable and reliable signature system that uses mobile devices will be proposed. The system considers the use of a mobile device to perform the digital signing of documents, interacting with a smart card that is contained in that same device. A smart card is chosen to provide a tamper

proof element, as it is the most robust and available solution in the vast majority of mobile devices. This way the system will ensure that the private signature key is securely stored inside the smart card. So, even if the mobile device is attacked, the security of the signature remains intact.

The implementation of such system is carried out in a business context with access to existing equipment on the market, such as a micro SD smart card, portable computer, and tablet device, by collaborating with PDM&FC[1], a Portuguese Information Technology (IT) Company specialized in cloud computing, system security, mobile applications, among other fields. The final product is intended for integration into a project that is of the responsability of a public administration organization, and which concerns the Electronic Certification of the Portuguese State.

## 1.2 Work Goals

When analyzing the limitations of the state of the art mobile solutions that provide qualified digital signatures, it can be concluded that there is the need for a mobile signature solution which aims to digitally sign electronic documents through the provision of qualified signatures, using more available security elements, such as operator-independent smart cards. Thus, the proposed solution keeps the strength of the previous solutions and offers greater flexibility.

In order to meet this goal, this work strives to develop a qualified signature system that enables users to digitally sign any electronic document with the use of mobile devices, facilitating the signature process. When designing the proposed system and analyzing the state of the art, there are several properties that must be taken into account, as described in the following:

**Performance**

The cryptographic operations can be time consuming and cause a negative impact on the performance of a system. Despite this, the solution should provide timely responses.

**Portability**

The concept of mobility has a strong presence in the solution, as it uses mobile devices to fulfill its tasks. Therefore, the solution must be designed to be portable, allowing it to be integrated into different mobile devices.

**Usability**

For a positive user experience, the solution must be easy to deploy and to use, and as least intrusive as possible.

**Availability**

The solution aims to provide a secure and flexible way for users to sign digital documents at any

[1] https://www.pdmfc.com/

time using mobile devices. Thus, the solution should not break down and should be physically transportable.

**Security**

The solution must ensure that only an authorized user can carry out the signing processes. Thus, the solution must be secure, otherwise the sensitive data can be accessed by malicious entities, which is a major security breach.

Creating a solution that achieves these properties and provides a secure way for users to sign digital documents using mobile devices is the main goal of this document.

## 1.3 Threat Model

It is also important to identify and understand the potential security threats to the proposed solution. A well known threats model is called STRIDE [14]. By applying the STRIDE model the following threats were identified (the type of threat is in bold):

- The operating system of the mobile device, namely the certification application, can be targeted for attacks by a malicious user or application (e.g., malware) in order to manipulate the data to be sent to the smart card (e.g., hash of the document to be signed); **– Repudiation**
- The certification application may be replaced in the mobile device manually if a malicious user gains physical access to the device, resulting in a modified certification app; **– Elevation of privilege**
- The smart card may be stolen and consequently ”broken” by attackers, leading to the disclosure of sensitive information; **– Information disclosure**
- An attacker can insert a malicious smart card into the legitimate user's mobile device without his knowledge, and consequently acquire the authentication Personal Identification Number (PIN) surreptitiously; **– Spoofing identity**
- A malicious user or application may enter wrong sequences of PIN values until the maximum number of attempts is reached, causing the smart card to be blocked. The same may also happen with the unblocking PIN, leaving the card unusable; **– Denial of Service**
- A malicious user may provide a fake or malicious version of the certification application, deceiving the users and leading them to install it; **– Elevation of privilege**

- A malicious entity may obtain physical access to the mobile device and enter the correct authentication PIN, gaining full-access to carry out any operations available (e.g., sign documents of behalf of the legitimate user, change the authentication PIN); **– Elevation of privilege**

## 1.4 Requirements

From the work goals, several formal requirements can be defined, which have to be satisfied by the proposed signature solution. The requirements are the following:

**Non-functional requirements**

- The performance of the solution should not affect the user's experience;
- The solution should have a low impact on the computational resources of the host device;
- The solution should be able to use different mobile devices.

**Functional requirements**

- The user must be able to digitally sign any electronic document by entering a security PIN;
- The solution must efficiently process one document at a time;
- The solution must be easy to interact with, promoting the automation of the signing process;
- The solution must be usable regardless of user location;
- Before signing any document, users must be informed that the digital signature is as legally binding as a manual signature;
- The solution must provide qualified digital signatures;
- The format of the digital signatures should be according to a standard format, so that they can be verified easily through different cryptographic suites that already exist. This format should ensure the integrity, authenticity and non-repudiation of the signed contents, and also provide the possibility of Long-Term Validation (LTV).

**Security requirements**

- The mobile signatures, to be considered equivalent to the handwritten signatures, should be generated in a Secure Signature Creation Device (SSCD) (the security token, like the smart micro SD card) and the user should have a qualified certificate. Thus, the SSCD shall perform on-chip key generation, guaranteeing the protection of the private keys and reinforcing their secrecy since they never go outside the smart card. No one other than the card owner can use the private signature key (ensures the secrecy of the keys as well as non-repudiation);

- The solution requires the authentication of the user so that it can reliably perform its functions: when digitally signing a document, or executing other sensitive operations, the authentication of the signer must be ensured through validation of a secret code known only by the signer (PIN code or password). The PIN is a secret that shall be protected in the same way as the keying material (stored in the smart card);
- The solution must take steps to protect the smart card from brute force attacks. Thus, the solution must set a maximum number of authentication attempts, after which the card is blocked or the data contained therein is erased.

## 1.5 Document Structure

This document presents and describes the proposed certification solution which uses modern technologies and mechanisms in order to ensure a reliable electronic certification.

Thus, the document is organized as follows: Chapter 2 presents the state of the art, starting with an analysis of the concept of digital signatures, and describing the different protocols, standards and mobile signature solutions that currently exist. It also examines the various mobile operating systems and smart cards. Chapter 3 describes the proposed solution, taking into account the state of the art. It presents the components, architecture, and functions of the solution. Chapter 4 details the proposed solution by describing the technologies selected to implement each component of the proposed architecture, and their behaviour. It also describes some details regarding the implementation of the solution. Chapter 5 presents the evaluation of the proposed solution. It compares it with other solutions described in the state of the art, and analyzes it from the security and performance point of view. It also depicts the results of the usability testing performed to the solution, and how the desired goals and requirements are met. Chapter 6 concludes this document. It summarizes the developed work and presents future work that can be done to further enhance the proposed solution.

# 2

# State of the Art

## Contents

This chapter covers the performed state of the art analysis to devise the proposed solution. Section 2.1 presents the digital signatures, detailing the cryptography employed by them, and describes the different protocols that can be used to perform these signatures. This section also presents the standards and the existing m-signature implementations, and the used technologies and infrastructures. Section 2.2 provides an overview of the available mobile operating systems in the market and performs a comparative analysis. Section 2.3 introduces the smart card technology, describing its surrounding components, such as technologies, formats, and communication protocols. Section 2.4 concludes this chapter by presenting the most significant aspects of the state of the art.

## 2.1 Digital Signatures

Over the last years the number of e-government, e-commerce and business information systems has been increased. Ensuring the security of such systems is progressively demanding, because malicious entities are constantly devising new ways to impersonate someone and forge documents.

Digital signature technologies can be used to address these vulnerabilities. They create a binding between an entity (e.g., a person or a company) and a data record, as they help to establish two key security properties – authentication and non-repudiation. Adittionaly, they achieve another fundamental property - data integrity -, ensuring that the data has not been modified since the signature was applied. Before starting this chapter, it is important to clarify the term “electronic signature” which is mentioned several times hereinafter: an electronic signature is not a ‘picture’ of a handwritten signature, but rather “(. . . ) a paperless way to sign a document using a unique credential associated with a given person that is logically attached to or associated with the document, carrying an authority equivalent to a handwritten signature.” as stated in [15].

### 2.1.1 Algorithms and Protocols

There are several solutions that implement the signing process by using public key cryptography [16], also known as asymmetric key cryptography. Although these solutions have the same purpose, they differ in some aspects, such as the storage of the private keys, the type of information on which they are based on and the type of technology they use. This allows to divide and categorize them into: biometric-based digital signature systems, which utilize some piece of user information (e.g., fingerprint and iris) to help generate the signatures; smart card-based digital signature systems, in which smart cards store all the necessary cryptographic information of the user (e.g., keys and certificates); and hybrid systems, which combine features from both systems. Only the smart card-based system is analyzed hereinafter, since it is the only one that is used by this thesis, as it will be explained in Chapter 3. The analysis of the other two is placed in Appendix A.

#### 2.1.1.A Asymmetric Key Cryptography

Asymmetric key cryptography, also known as public-key cryptography, refers to a cryptographic technique which uses a pair of keys – one private and the other public. Although the keys are different, they share a mathematical relation. This type of cryptography is mainly used for public-key encryption and digital signatures. The public key can either be used to encrypt data or verify digital signatures whereas the private key can be used to decrypt that same data or create the digital signatures. This public-key scheme was first published by Diffie and Hellman [17] and works as follows: the user makes his public key available to everyone while his private key remains secret and is only used by authorized parties. The keys are generated in a way that it is computationally infeasible to determine the private key from its corresponding public values. Thus, the public key can be distributed without compromising its security. Anyone with a copy of this key can encrypt information or verify digital signatures, whereas only the holder of the private key can decrypt the information or create the signatures. This is the strength of this scheme.

Its use for digital signatures is now considered, as it is the focus of this thesis. Figure 2.1 illustrates the basic process of one of the most widely used digital signature algorithms today - RSA [18] (after the inventors Rivest, Shamir, and Adelman). There are two entities, the originator (whom we call Alice) and the recipient (called Bob). Alice wants to send a signed document to Bob so, she processes her message, creating the hash value or the message digest. The hash is then encrypted with her private key (securely stored), producing the digital signature, which is sent along with the message. Bob in turn receives the message and signature, and verifies the latter by decrypting the signature value with Alice's public key, resulting in the original hash (created by Alice). He then compares this result with the newly-created hash of the received message. If both values are identical, the digital signature was successfully verified and Bob can be sure that the message was unmodified since it was signed by Alice. This assurance is based on the assumption that Alice's private key has not been disclosed to anyone and that no one else can perform this signature operation.

When using this type of cryptographic system, besides the need to keep the private key a secret, it is also necessary to be cautious with the origin of the public key. Trusting this key is a delicate procedure and represents a major concern in order to assure the correct association between this key and its owner. If this association is corrupted, it could lead to falsely signed documents. Typically, this concern is addressed through the use of a Public-Key Infrastructure (PKI), in which one or more third parties, named Certificate Authorities (CAs), certify the ownership of the key pairs, meaning they bind the public keys with the respective user identities.

There are other digital signature algorithms frequently used, such as the Digital Signature Algorithm (DSA) and Elliptic Curve DSA (ECDSA), which differ in the way they create and verify the signature, and also in their performance.

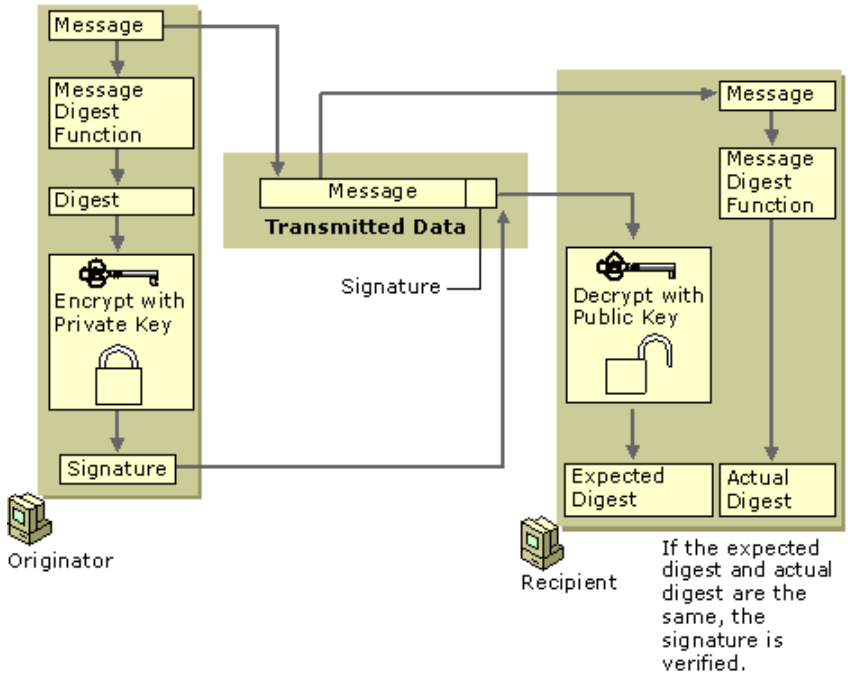



**Figure 2.1:** Basic RSA digital signature process [1]

#### 2.1.1.B Smart Card-based Protocol

A smart card (also called chip card or Integrated Circuit Card (ICC)) is a portable computing device which contains programmable memory and offers some tamper-resistance capabilities [19].

Over the years, companies and end-users have given increasing importance to public-key cryptography, because they see it as a very reliable technology to ensure the security of their applications [20]. Consequently and also due to the increasing technological evolution, smart cards supporting public-key cryptography have undergone huge developments and are nowadays used in many types of security systems throughout the world, such as financial, identification, healthcare and computer security [21].

Today's cryptographic smart cards can perform on-chip key generation and store the private key to avoid the risk of having more than one copy of it. They can also provide a secure computing environment for private key operations. This type of cards can be seen as a security token capable of providing secure identification and digital signatures, thus serving as a support system [20].

In a typical digital signature implementation, the hash is computed from the document and is sent to the smart card, which encrypts the hash using the stored private key, and returns the encrypted hash, which represents the signature. Usually, a user is required to enter his PIN code in order to enable the signing process.

The main weakness of this solution is the high risk of loss, theft (leaving the scheme's security reduced to that of the PIN system) or breaking of the smart cards. However, this risk can be significantly minimized depending on the format of the used card. There are formats as small as SIM or Micro SD

cards that allow the user to place the smart card inside a computational device, like a smartphone, which then interacts with it and performs public-key cryptography. These formats are described in greater detail in Section 2.3.

### 2.1.2 Standards

Electronic documents form a vital part in many communication processes since they provide an easy and flexible way to carry important information that is relevant to all parties involved in these processes. In order to ensure the integrity, authenticity and non-repudiation of the documents' contents, digital signatures are used and therefore are of great importance. Ever since the Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures [10] was set up, many countries and companies have set different requirements for digital signatures. Consequently, these requirements affect the type of signatures that are used.

There are several signature formats defined which can be divided into three classes: Cryptographic Message Syntax (CMS) based signature formats; Extensible Markup Language (XML) based signature formats; and Portable Document Format (PDF) based signature formats.

The following sections detail these three classes in greater detail.

#### 2.1.2.A CMS

The CMS is a widely used standard defined by the Internet Engineering Task Force (IETF) for cryptographically protected messages. It describes an encapsulation syntax for data protection using Abstract Syntax Notation One (ASN.1) and can be used to digitally digest, sign, encrypt or authenticate any type of digital data.

The CMS is derived from the syntax of Public-Key Cryptography Standards (PKCS) #7 and it allows multiple encapsulations. For instance, one encapsulation envelope can be nested inside another or one participant can sign digital data that has already been placed into an envelope. Arbitrary attributes, like signing time, are also allowed to be signed along with the content of the message. Furthermore, it enables other interesting attributes such as countersignatures to be associated with a signature [22].

A CMS signature provides a digital signature in a Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER) encoded (binary encoded) ASN.1 structure. This structure contains the signed content as well as an arbitrary number of signatures on that content, along with information about each signer (digest and signature algorithms, CMS version, etc.).

Several architectures for certificate-based key management can be supported by the CMS, like the one defined by the Public Key Infrastructure X.509 (PKIX) working group [23].

Many other cryptographic standards use CMS as the key cryptographic component, such as: Secure / Multipurpose Internet Mail Extensions (S/MIME), a protocol for doing secure emails (encrypted mail);

PKCS #12, a standard for a container commonly used to bundle a private key and its X.509 certificate or to bundle a certificate chain; and Trusted (digital) timestamping protocol, a process for securely keeping track of the creation and modification time of a document, without the possibility of someone changing it once it has been recorded.

#### 2.1.2.B CAdES

The European Union Directive 1999/93/EC [10] defines a legal base for electronic signatures. It distinguishes between ”normal” and ”advanced” electronic signatures: while the first is given reduced legal validity, the latter is required to meet the same legal requirements for electronic information as the handwritten signatures for information on paper [24].

As previously seen, CMS is a general framework for digitally signing documents such as PDF or e-mail (S/MIME). CMS Advanced Electronic Signatures (CAdES) is a standard which comprises a set of extensions to CMS signed data. Since CMS lacks certain important features of extended electronic signature, the most evident of which is LTV, CAdES specifies a number of additional profiles for use with advanced electronic signature in the meaning of EU Directive 1999/93/EC [25]. Since digitally-signed documents may be used or archived for long periods of time, it must be possible to prove signature validity at any time in the future in spite of the technological advances. Accordingly, any later attempts by the signer or the verifying party to repudiate the validity of the signature will be futile. This is an important benefit, because even if the cryptographic algorithms responsible for the signature are broken, the electronically signed documents can keep their validity for long periods.

There are six profiles defined by CAdES specification, each building up on the previous one [26]

1. CAdES – basic form defining elements for authentication and protection of integrity of records but not providing non-repudiation of its existence;

2. CAdES Timestamp (CAdES-T) – addition of the timestamp ensures non-repudiation;

3. CAdES Complete validation data (CAdES-C) – builds up on the CAdES-T by adding references to the set of data supporting the validation of the electronic signature (i.e. the references to the certification path and its associated revocation status information). This form is useful for those situations where such information is archived by an external source, like a trusted service provider;

4. CAdES eXtended validation data (CAdES-X) – builds up on CAdES-C by adding timestamps to protect against the risk that any keys used in the certificate chain or in the revocation status information may be compromised;

5. CAdES eXtended validation data incorporated for the Long term (CAdES-X-L) – builds up on CAdES-X by adding the validation data (i.e. certificates and revocation values) for those situations

where the validation data are not stored elsewhere for the long-term;

6. CAdES Archiving validation data (CAdES-A) – builds up on CAdES-X-L by adding time-stamps for archiving signatures.

This signature format is particularly useful for numerous types of transactions, including business transactions (e.g., purchase requisitions, contracts) where LTV of such signatures is important.

#### 2.1.2.C XMLDSig

Extensible Markup Language Signature, also called XMLDSig, is defined by the World Wide Web Consortium (W3C) Recommendation, in the XML Signature Syntax and Processing, and defines an XML syntax for digital signatures. It comes from the need for native XML security services, in order to ensure the security of transactions that make use of XML [26] [27]. It is intended to assure the integrity, message authentication and/or signer authentication for data of any type. This data can be located within the XML that includes the signature or somewhere else. It is also used by several Web technologies like Simple Object Access Protocol (SOAP) and XML Remote Procedure Call (XML-RPC) [28].

One advantage of XMLDSig is the possibility of signing only certain parts of an XML document. Therefore, an XML document can possess several signatures, created at different times, by different entities and associated to different data. An XML Signature can be represented in three different ways based on its location regarding the signed data [26].

- Detached – the signature is used to sign a resource outside its containing XML document, i.e. the signature is associated with data that is external to the file in which it is located. These data are usually accessible on the Web. Consequently, the signature is "detached" from the content it signs;

- Enveloped – the signature is located inside the element that contains the data and signs parts of these data, i.e. the signature is used to sign some part of its containing document;

- Enveloping – the signature contains the signed data within itself, meaning it includes the signed data. More precisely, the signed data is located on an Object element inside the signature itself.

Compared with other formats of digital signatures such as CMS, XML Signature is more flexible since is does not operate on binary data, but on the XML Information Set, which allows to work on subsets of the data and consequently to have different ways to bind the signature and signed information. Another benefit is the concept of canonicalization, signing only the "essence" and removing differences like whitespaces and line endings which are meaningless. However, there are some disapprovals regarding this concept that state that XML canonicalization is a process that triggers excessive latency, contributing to the overload of some transactional, performance sensitive applications like Web Services [29] [30].

#### 2.1.2.D XAdES

XML Advanced Electronic Signatures (XAdES) specification extends the XMLDSig standard in the same way that CAdES extends CMS. It defines several XML formats for advanced electronic signatures that feature a large validity and are conforming to the EU Directive 1999/93/EC [10], and also include other useful information. This contains proof regarding its validity even if the signer or verifying party later repudiates the validity of the signature. Thus, this type of signature fulfills the requisites for non-repudiation and long-term validity, and can in case of a dispute between the signer and verifier be used for arbitration [26] [27]. Like CAdES, XAdES specification defines six specific profiles. Each profile builds upon the previous one, adding new information that improves the signature's security level in terms of non-repudiation and long-term validity: XAdES – basic form, XAdES Timestamp (XAdES-T), XAdES Complete (XAdES-C), XAdES eXtended (XAdES-X), XAdES eXtended Longterm (XAdES-X-L) and XAdES Archiving (XAdES-A).

The difference between CAdES and XAdES specifications is that while the first renders the signatures as binary data, the latter provides an XML solution.

#### 2.1.2.E PAdES

PDF is known as a file format used to present documents and has supported digital signatures for several years based on PKCS #7 (a precursor to CMS). This support is defined in International Organization for Standardization (ISO) 32000-1 (PDF 1.7) [31].

The PDF Advanced Electronic Signatures (PAdES) standard introduces a number of restrictions and extensions to PDF and ISO 32000-1, making it suitable for advanced electronic signatures in the meaning of the EU Directive 1999/93/EC [10]. Like CAdES and XAdES, it shares the advantage of its electronic signed documents remaining valid for long periods of time. However, PAdES differs from these two standards as "it applies only to PDF documents and defines requirements that PDF viewing and editing software must follow when using digital signatures in PDF documents." as stated in [2].

For PDF documents, the digital signature is integrated in the signed PDF document, like an ink-on-paper signature is placed at a specific position on a given page and becomes part of it. This enables the copy, storage and distribution of the complete PDF document as if it were a simple electronic file. The digital signature can also be visually represented as a form field, which is an important property that helps to distinguish it from CAdES and XAdES, which are targeted for applications that may not involve human-readable documents [26].

One big advantage of PAdES is that it does not require the development or customization of dedicated software, as it can be deployed using widely available PDF software, like the free Adobe® Reader® [32].

#### 2.1.2.F Summary

Figure 2.2 provides a comparison of the features of the three advanced signature standards previously presented.

| PAdES | CAdES | XAdES |
|---|---|---|
| • Contains signatures within the PDF<br>• Supports XML data<br>• Included within the ISO PDF Standard<br>• Includes signing and verifying in PDF software—no customized programming required<br>• Supports serial form-fill and signatures for approval workflows<br>• Supports a visual signature appearance in the document<br>• Provides long-term validity | • Enables signing of any data, including PDF<br>• Renders signature as binary data<br>• Often requires customization of applications or generic signing outside the application<br>• Supports multiple signatures applied in parallel, serial by repeated signing<br>• Appearance is up to the application to provide<br>• Provides long-term validity | • Provides an all XML solution<br>• Signs any data including PDF and binary<br>• Supports XML package or separate files<br>• Often requires customization of applications or generic signing outside the application<br>• Supports multiple signatures applied in parallel, serial by repeated signing<br>• Supports a visual signature appearance, depending on the application<br>• Provides long-term validity |

**Figure 2.2:** Comparison of PAdES, CAdES, and XAdES (Adapted from [2])

It can be seen that the PAdES format only applies to PDF documents, which is somewhat restrictive, whereas CAdES and XAdES allow to sign any kind of data, including PDF and binary. However, PAdES has the advantage of not requiring customized applications for signing and verifying documents. The XAdES format does not operate on binary data, like CAdES, but on the XML Information Set. This allows it to work on subsets of the data and consequently having different ways of binding the signature and signed information. All three formats provide LTV.

### 2.1.3 SIM Card-supported m-signature solutions

More and more, users are in possession of multiple mobile devices, such as smartphones and tablets. Consequently, there is an increasing interaction with information systems through these devices, turning them into personal trusted devices. The mobile signatures provide a way to verify the signer's identity to third-parties without the need to communicate face-to-face. Thus, they allow the users to digitally sign documents, transactions (e.g., bank transactions) and other types of data via mobile phones, providing strong authentication, integrity and non-repudiation properties.

There are a variety of mobile technologies and infrastructures that support these signatures by using the SIM card (smart card) as the security token, thus providing qualified signatures. Some use only the SIM card while others use the SIM card and the middleware of the mobile device. There are also those that, besides using the SIM card, include high level services that are independent of the mobile device's technology. The different solutions are described next.

#### 2.1.3.A WIM technology

WIM (Wireless Application Protocol (WAP) Identity Module) is a security module that is implemented in the SIM card, which has tamper-resistant capabilities, with the main purpose of being used by WAP applications. It provides secure storage for cryptographic credentials (certificates, keys and authentication objects, like PINs) as well as processing capabilities. Thus, it allows the execution of cryptographic operations, providing the WAP applications with security services and allowing the use of digital signatures. The WIM functionality is implemented as an independent application on the SIM card. This way, the key pairs are created inside the card, the signature processes are done by the WIM application and the keys never leave the card, causing the SIM card to be considered as a SSCD. For these reasons, the digital signatures generated by the WIM application can be classified as qualified signatures [3].

The WIM technology can be integrated with mobile Web applications. These can take advantage of the cryptographic features of the WIM application. Languages like Wireless Markup Language (WML) and Extensible HyperText Markup Language (XHTML) are used for mobile Web apps and they possess several script libraries for performing asymmetric operations on the client side. These libraries can be incorporated in the mobile browsers that support WMLScript or XHTMLScript. Then, when a signature function (belonging to a library) is invoked, the WIM application contained in the SIM card generates the digital signature and returns it according to the CMS format. Since the private key is always kept inside the card during the signature process, this technologies can also provide qualified signatures.

Over the years, several manufacturers have included the WIM functionality in SIM cards and it was used in some commercial solutions [33] [34] [35], namely in securing mobile commerce and financial transactions (e.g., banking services) over the mobile Internet, and in providing security and authentication technologies over General Packet Radio Service (GPRS) mobile networks. Presently, the level of usage of WAP is very low, since all modern mobile devices support full HyperText Markup Language (HTML) browsing and do not use any WAP markup, like WML.

#### 2.1.3.B USAT-I technology

First of all, SIM Application Toolkit (SAT) refers to a Global System for Mobile communication (GSM) standard which allows the SIM card to initiate certain actions (being proactive), instead of just reading commands from the mobile device. More precisely, it defines a collection of commands which acts as an interface between a 2G SIM card and a mobile device. Typically, a SAT application is a Java Card applet, which can be placed on the SIM card at the manufacture process or downloaded and installed through an Over The Air (OTA) server at any point during the lifetime of the card [3]. SAT later evolved into Universal SIM Application Toolkit (USAT). This technology, defined by the 3rd Generation Partnership Project (3GPP) [36], is designed for 3G networks and has similar concepts to SAT, and also some improvements such as the ability to open Hypertext Transfer Protocol (HTTP) connections with Internet

Protocol (IP) devices from a command.

Finally, USAT-I (USIM Application Toolkit Interpreter) is a new type of SAT application of the 3G SIM cards which consists on a byte-codes interpreter. This interpreter is capable of processing applications which arrive in the form of byte-codes inside Short Message Service (SMS) messages. This procedure is managed by a network infrastructure with two main components – an application server and a gateway – depicted in Figure 2.3. The application server hosts the applications, each being a series of pages written in a subset of WML which can have tags that define what the application should do once it arrives at the SIM, like creating a user interface or calling a SAT command [3]. The gateway in turn, receives these pages, transforms them into byte-codes and sends them to the USAT-I application of the mobile device's SIM card, typically via SMS messages. Upon reception, the interpreter will interpret each page just as a desktop browser interprets an HTML page that retrieves from the Web [37], thus working like a browser inside the SIM card.

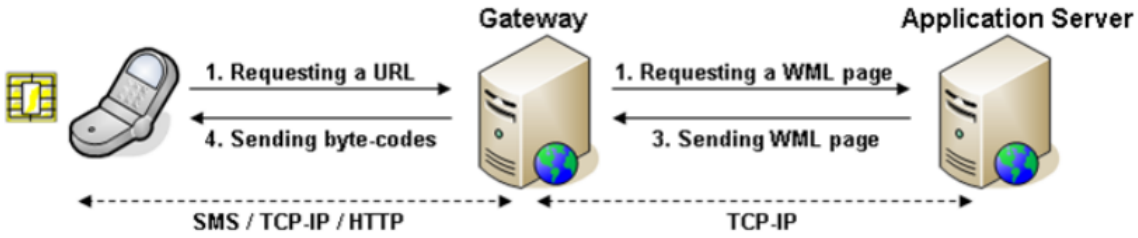



**Figure 2.3:** USAT-I Infrastructure [3]

An important feature of a USAT-I browser defined by the 3GPP is that it can be extended through plug-ins. Each plug-in consists in a Java Card applet and is installed on the SIM card at the manufacturing stage, enabling the USAT-I browser to call the plug-in commands. There are special plug-ins, such as PKI plug-ins, that some manufacturers can provide to the USAT-I browser of a SIM/WIM card and which can work with the WIM application of the card, thus allowing the execution of cryptographic operations (e.g., digital signing) from the USAT-I browser.

The major advantage of this solution is that it avoids installing applications through OTA servers which is especially useful for heavy applets. Instead, the applications are packed into byte-codes and sent to the SIM card, which can be easily interpreted and removed. Another strong point is the ability to use extensible plug-ins, namely PKI plug-ins. These enable the integration with the WIM application, causing the keys to never leave the card (hence never compromised) and to classify the resulting digital signatures as qualified signatures. Also an advantage is the fact that it does not require any software in the mobile device and is capable of communicating with the GSM network through SMS messages. The downside is that the network operators have to provide and maintain the network infrastructure that supports this technology, which requires some investment on their part. Some implementations have been deployed based on the SIMalliance Toolbox (S@T), such as the Logos' S@T browser [38] and the

SmartTrust Wireless Internet Browser (WIB) [39].

#### 2.1.3.C SATSA

The Security And Trust Services API (SATSA) is a specification for Java Platform Micro Edition (Java ME), also known as Java Specification Request (JSR)-177, that enables Java ME applications (Java ME MIDlets) to communicate with a security element, such as a SIM card [3]. This specification (depicted in Figure 2.4) allows the security element to perform different security processes on Java ME MIDlets like authentication of users or electronic signatures. The SATSA API defines four optional packages that serve different needs of communication with the security element. The packages are the following, as described in [40]:

- Application Protocol Data Unit (APDU) - Enables applications to communicate with smart card applications using a low-level protocol;
- Java Card Remote Method Invocation (JCRMI) - Provides an alternate method for communicating with smart card applications using a remote object protocol;
- PKI - Enables applications to use a smart card to digitally sign data and manage user certificates;
- and CRYPTO - A general-purpose cryptographic API that supports message digests, digital signatures, and ciphers.

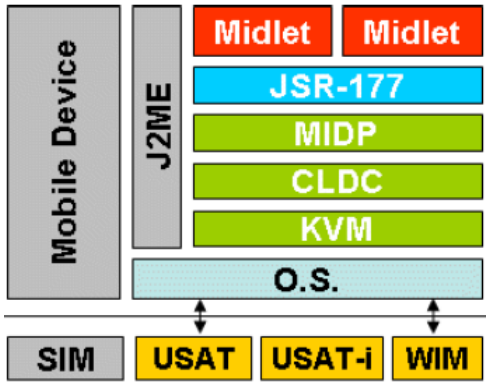



**Figure 2.4:** SATSA architecture [3]

Assuming for instance a SIM card as the security element, there are three ways to communicate with it, depending on the SIM's application type: via the APDU package that allows to send APDU commands to USAT applets using the ISO 7816 protocol [41]; through the JCRMI package which enables the request of Java Card RMI objects; and using the PKI package that can use the SIM card's WIM application in cryptographic operations like generating electronic signatures without disclosing the private key.

SATSA provides some advantages such as the MIDlets' portability, allowing them to be transported between different mobile devices, and the ability to perform signature processes that follow the standards by using a WIM application, resulting in qualified signatures. The main weakness is the low number of deployments by device manufacturers. However, over the years several manufacturers like Nokia (from Series 40 onwards) and Samsung have included SATSA in some of their mobile phones.

#### 2.1.3.D MSS

The Mobile Signature Service (MSS) is a specification defined by the European Telecommunications Standards Institute (ETSI) which includes several technical reports and specifications published in [42]. It is designed to simplify the development of mobile signature-based solutions for the mobile application/service providers [43] and is comprised of several entities: end-user, smart card issuer (typically the mobile operator), Registration Authority (RA), CA, Mobile Signature Service Provider (MSSP), roaming MSSP and of course, Application Provider (AP) [3]. These entities are shown in Figure 2.5.

Starting with the end-user, he/she has a mobile device that contains a smart card issued by the card issuer. The card in turn contains a signing application that is protected with a signing PIN and capable of generating electronic signatures to authenticate the user. The signing application can also produce new keys and store the certificates associated with these keys. The entity responsible for distributing the certificates is the CA and does so through the RA, which is in charge of checking the identity of the end-users. The MSSP is the entity through which the MSS is provided and is associated to a Mobile Network Operator (MNO). It encapsulates for the AP the complexity of the mobile devices and the different methods supported for generating the signatures, providing to the AP a Web service interface. To use the service, the end-user must be registered with the MSSP associated to his/her MNO. The MSSP can then invoke the user's signing app to get an electronic signature for a specific data. Since the app is protected, the user must enter the signing PIN in his/her mobile device to confirm that he/she is validating the signature, ensuring that the signature is only performed with his/her authorization. The MSSP can also deliver some value-added services, for instance timestamping.

In this type of systems, the APs do not need to supply the end-users with any kind of software. If the AP have a transaction that requires an electronic signature, they send a request to the MSSP with certain data related with the transaction, and this one invokes the user's signing application, obtaining the signature of the provided data. This is the procedure performed when the end-user and the AP both work with the same MSSP, and is described with greater detail in [3]. There are other possible scenarios, like the end-user and AP having different MSSPs or the user not being in his/her "home" network (MNO). This is where the roaming MSSP is useful, because it forwards the AP's signature requests to the MSSP associated to the user's MNO, allowing the communication with the signing application.

This solution has been implemented by some companies that provide mobile signature services like

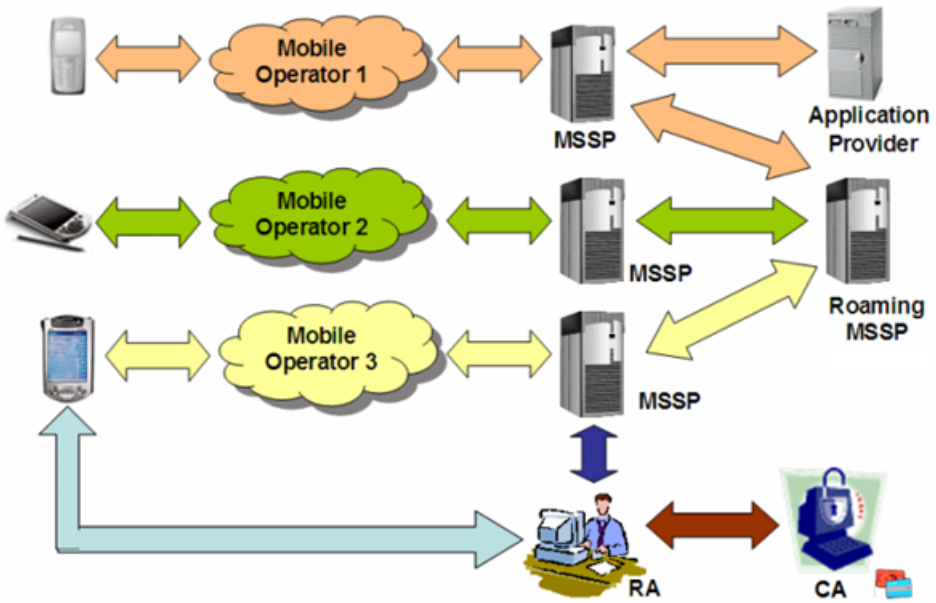



**Figure 2.5:** Mobile Signature Service Framework [3]

Methics Ltd [44], Valimo Wireless Ltd [45] and G&D SmartTrust [46]. Its main advantage is that APs do not need to develop specific m-signature solutions for the different mobile devices. That is performed by the MSSP. Thus, the MNO does not allow any third party to access the user's SIM card. Another advantage is the promotion of interoperability between the APs and MSSPs, since the service is defined as a Web service. The ability of the APs to configure the signature process is also advantageous, allowing them to specify certain advanced properties of the signatures like their format and the key used to generate them.

In contrast, the main problem of this solution is that the AP is obligated to set up an agreement with a MSSP that is capable of delivering the aforementioned services. Moreover, there is no definition for the interface between the mobile user and MSSP. The low availability of the solution in the market is also a drawback. Almost all the existing mobile operators do not offer the infrastructure or the SIM cards with the necessary features. In Europe for instance, only some commercial solutions have been deployed based on these specially designed SIM cards, mostly using Methics and Valimo Wireless's technologies. Another limitation is if an AP wishes to provide a universal solution (to be used by all MNOs), every MNO would need to support it, being forced to develop a MSSP. Furthermore, the roaming mobile signature services among the MNOs would have to be developed.

#### 2.1.3.E MSAU

With the evolution of mobile technology, new and more efficient architectures of mobile services were developed. The Mobile Signature Application Unit (MSAU) [4] is a proposed architecture which is based

on Java ME and SATSA, and aimed at ”reducing” the role of the MSSP. This entity's sole purpose is now to support the mobile users and Application/Service Providers (MASP) in signature verification and in providing the necessary software components for the creation of applications or services.

This architecture integrates the MSS in the mobile device as an application unit (the MSAU), making the Application/Service Provider the entity responsible for communicating with the MSAU through different mobile technologies such as Universal Mobile Telecommunications System (UMTS), Bluetooth and Near Field Communication (NFC), as depicted in Figure 2.6. Whenever necessary, the MSAU accesses the SIM/Universal Subscriber Identity Module (USIM) card to generate m-signatures, providing qualified signatures, which are then sent to the MASP to be verified so as to complete the transaction in progress (banking, payment, etc.).

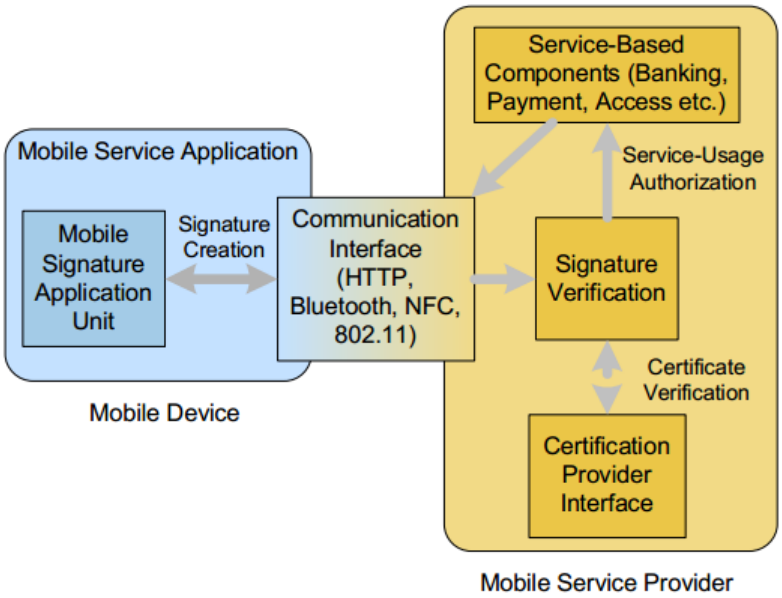



**Figure 2.6:** Mobile Service Architecture [4]

Specifically, the MSAU comprises two parts: the Cryptographic Engine (CryptoEngine) located on the smart card and the CryptoEngine Interface located on the mobile handset, as illustrated in Figure 2.7. These parts are able to communicate with one another via APDU messages, performing cryptographic processes like signatures, public key generation, etc.

This architecture is restricted to mobile devices that support Java ME and SATSA technologies which is its major disadvantage, given the low market share of Java ME [7]. Also, since it does not offer a standardized communication interface to the MASP, and the invocation of the service is open, then the MASP needs to support several different types of access depending on the user. Different applications with different technologies for different types of devices would need to be developed. Consequently, it is very hard to achieve interoperability and a greater investment is needed [6].

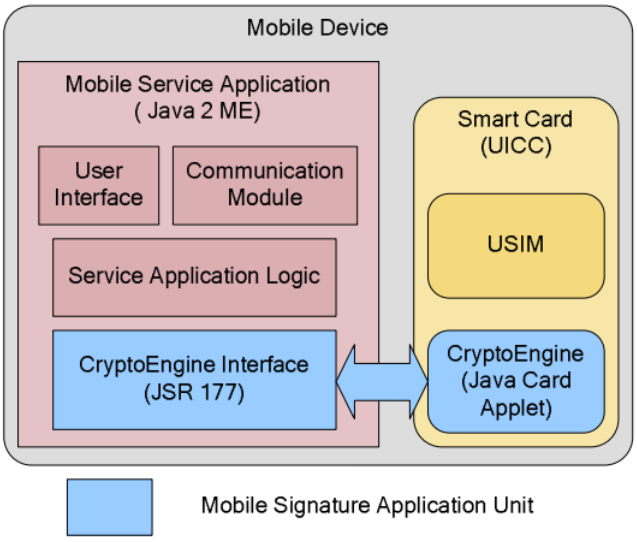



**Figure 2.7:** Mobile Service Application Architecture [4]

#### 2.1.3.F SPMS

The Self-Proxy Mobile Signature (SPMS) [5] is a proposed model which uses proxy certificates to ensure the security of the user's private key and speed up the signature generation process.

A common feature between all previously defined SIM-supported signature solutions (WIM, SATSA, etc.) is that the generation of digital signatures is usually a slow process or at least not as fast when compared to device-based signature solutions. Although smart cards (SIM cards) have dedicated cryptographic processors, sometimes something more powerful is needed like the mobile devices' Central Processing Unit (CPU). Some m-commerce apps, like mobile auctions and stock marketing, may have timing constraints and thus need several mobile signatures in a short period of time. This model is able to comply with such constraints using a special type of certificates. A proxy certificate is a short lived certificate which is issued (and thus signed) by an end-user instead of a CA for the purpose of providing restricted proxying and delegation. This means that the holder of the proxy certificate is authorized to act on the signer's behalf [47].

In the SPMS model, the private key of the user is also stored inside the SIM card. However, the signature process is performed differently since it occurs in the mobile device's environment, taking advantage of the device's powerful CPU. For this to happen, the SIM card issues a proxy certificate to the mobile device, acting like a CA for the device. This issuing phase is comprised of two processes, both performed in the SIM card: the key generation and certificate signing. Although the key generation takes a considerable amount of time to execute, it can be made offline, thus having no negative impact on the time of online signature generation. Once the SIM card produces the keys, it creates a proxy certificate for the device which is passed (along with the proxy's private key) to that device, which in turn stores

them in a proper location. Thus, the private key of the user never has to leave the SIM card, and the mobile device can sign the data on behalf of the SIM card using the proxy's private key. Consequently, when verifying the signature, the verifier has to carry out an additional step besides the traditional ones, which is checking the proxy certificate's signature and confirming that the user has indeed delegated his/her signature rights to his/her mobile device. The security of the proxy's key is assured by assigning it a short validity period and also protecting it with a secret code. Even if an attacker cracks the key, it is not possible for him to discover the user's private key.

The SPMS model is in turn integrated with the previously presented solution - MSAU - through the addition of a proxy manager module to the Mobile Service Application, and some modifications of the Java Card applet to satisfy the model's requirements. The resulting architecture is depicted in Figure 2.8.

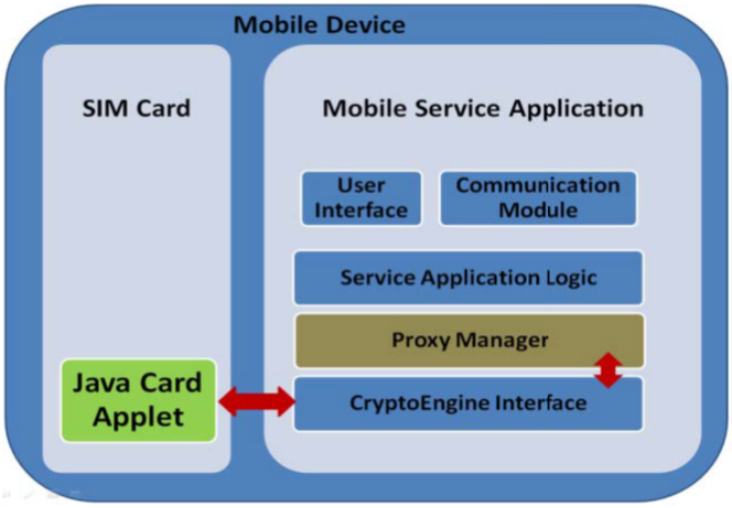



**Figure 2.8:** The SPMS model integrated with mobile service application architecture [5]

This SPMS has some benefits besides those already discussed, such as the low usage of the user's private key, since it is only used to sign proxy certificates; the possibility of the proxy certificates having large key lengths which are not supported or processed efficiently by the SIM card; and the improvement of the signature algorithm's performance, as the signature is performed in the mobile device and thus, subject to optimization. Therefore, this model is able to increase the efficiency and flexibility of mobile signatures by making the user's SIM card delegate its signature powers to a more powerful entity – the mobile device –,and never disclosing the user's private key. This delegation does not come without security risks: the device in which the user trusts his signature powers is subject to attacks. In that same device, the proxy's private key is stored and, at the signing time, that key is revealed to the device.

#### 2.1.3.G MNO-iMSS

In [6] a novel MSS is proposed that is independent of the MNO and does not depend on any handset-based specific technology for generating the signatures in the handset. In order to facilitate its acceptance, it is based on the previously presented MSS specification, thus featuring most of its services and security mechanisms. However, it differs in some respects such as the fact of not requiring the existence of roaming MSSPs and not demanding the development of a MSSP by each MNO.

In this m-signature solution the MSSP, which is called MNO-independent MSSP (iMSSP), does not verify if the end-user's mobile handset is available to perform an m-signature, like in the previous MSS specification. Instead, it is the user who, when ready to generate signatures, verifies through a Web service in the iMSSP if there are m-signature requests waiting to be signed. By defining a Web service interface, the interoperability between users and MSSPs increases, allowing the user's signature application to be used with different MSSPs. Also, the users have a higher control over the signature processes, since they can choose when to receive the m-signature requests and thus perform them.

In the ETSI-specified MSS, the mobile device has to wait for the signature requests to arrive from the MSSP, as if it were a server. This causes the mobile device to establish additional mechanisms in order to be protected against potential attacks from unknown entities aimed at obtaining a signature from the user. On the other hand, in the MNO-iMSS there is no need for the mobile device to behave as a server, since it is the device that connects to the server to retrieve the signature requests. However, this could cause a problem: part of the instantaneity derived from the ETSI specification could be lost and also some signature requests, when downloaded, could no longer be valid. To avoid this, the iMSSP uses a notification mechanism for the user (e.g., via an e-mail or SMS). Figure 2.9 shows the various entities of this proposed system.

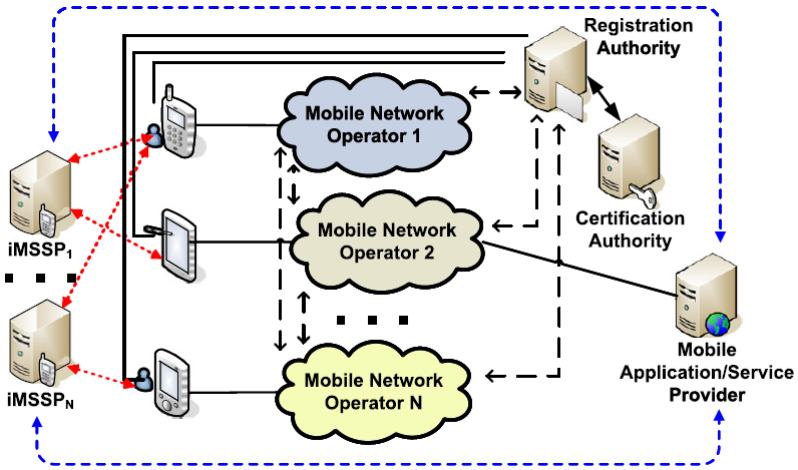



**Figure 2.9:** Roles in the system [6]

The communication between the different parties is protected using security mechanisms (e.g.,

SOAP / HTTP over Transport Layer Security (TLS)), ensuring the confidentiality, integrity and authentication of the information exchanged. Additionally, the m-signature created by the mobile user ensures the non-repudiation of the information signed.

Regarding the security of the m-signature app, it must be in accordance with the security requirements established in [48], assuring the existence of trusted channels between the SSCD and the application.

The iMSSP solution presents many improvements in regard to the ETSI-specified MSS: it allows access to its services by using Web services and mobile devices which can develop their signing application with any technology; the end-user is able to use different handsets for generating the signature, making it easier to deliver mobile signatures; and also, despite this solution being MNO-independent, if an MNO chooses to integrate it, it can get some additional benefits. Thus, for MNOs that are not supporting the ETSI specification they can benefit from the network traffic generated when the users use the iMSSP's network services. For MNOs that are supporting the ETSI specification, complementing it with this solution can result in increased number of mobile users using their iMSSP, since users from different MNOs can register in their iMSSP and use their solution, supposing more network traffic and thus, more benefits. Also in roaming scenarios, users are able to use the MSS in which they are registered even if the visited MNO does not support the ETSI specification or the iMSSP solution. The only requirement is that the MNO offers connectivity. This is because the network services that the visited MNO provides are only used to access the iMSSP in which the users are registered.

The previously presented MSS-based solutions (MSS, MSAU and MNO-iMSS) have two kinds of drawbacks. On the one hand, some of them require that all MNOs develop the service, whereas others do not use efficient communications. To cope with these limitations, the same author proposes a lightweight MSS named SIPmsign [49]. This proposal aims to provide an MSS for the new generation of mobile communications, defining a secure and more efficient way for exchanging the data through the use of the Session Initiation Protocol (SIP).

#### 2.1.3.H Summary

Table 2.1 provides a comparison between the abovementioned mobile signature solutions in relation to several features, such as the implied technologies and signature platforms.

It can be seen that all solutions provide asymmetric capabilities as well as device OS-independent systems, although some solutions require that the mobile devices' OS has Java ME support. They can also offer smart card possibilities. Next, the solutions are classified according to the used signature platform (SIM card, hybrid and services). Finally, it can be seen that only WIM-related solutions are considered SSCD directly, as they use a secure device that is under the sole control of the end-user. Consequently, the signatures provided by them (qualified signatures) can be considered legally equiv-

alent to the handwritten signatures. The MSS-based solutions - MSS and MNO-iMSS - depend on the technology supported by the mobile device. If the solution is based on SATSA and WIM for instance, then its signatures can also be considered equivalent to handwritten signatutres.

Thus, there are several mobile signature solutions that are based on asymmetric cryptography and capable of delivering qualified digital signatures. These solutions either use outdated or unsupported technology, or are based on external services, depending on components that are external to the mobile device. Also, they all share a common drawback: the smart card used as the security element (SIM card) depends on the mobile operator. Therefore, it is concluded that there is a need for a mobile signature solution that provides qualified signatures by itself using a more available and operator-independent smart card. The following section analyzes the existing mobile operating systems in the market.

| Solutions \ Features | Asymmetric cryptograpy | Device OS-independent | Smart card | SIM card-based | Handheld and SIM card-based | Services-based | Qualified signature | SSCD |
|---|---|---|---|---|---|---|---|---|
| **WIM** | X | X | X | X | | | X | X |
| **USAT-I** | X | X | X | X | | | X | X |
| **SATSA** | X | $X_1$ | X | | X | | X | X |
| **MSS** | X | X | $X_2$ | | | X | $X_2$ | $X_2$ |
| **MSAU** | X | $X_1$ | X | | X | | X | X |
| **SPMS** | X | $X_1$ | X | | X | | X | X |
| **MNO-iMSS** | X | X | $X_2$ | | | X | $X_2$ | $X_2$ |

**Table 2.1:** Feature-based comparison

Notes:

$X_1$ - Only if the device's OS supports Java ME

$X_2$ - It depends on the device-side technology

## 2.2 Mobile Operating Systems

Over the years, laptops, smartphones, tablets, and other mobile devices have become widely used by people all around the world mostly due to their computing and communication features. The number of existing mobile devices and connections has suffered a tremendous increase and is expected to continue its growth throughout the years [50]. The constant competition between the major vendors for market dominance has had a determining role in this increase, since vendors continuously provide new Mobile Operating Systems (MOSs) with new communication technologies and better processing capabilities. Nowadays, there are many MOSs occupying the market with different proportions as illustrated in Figure 2.10.

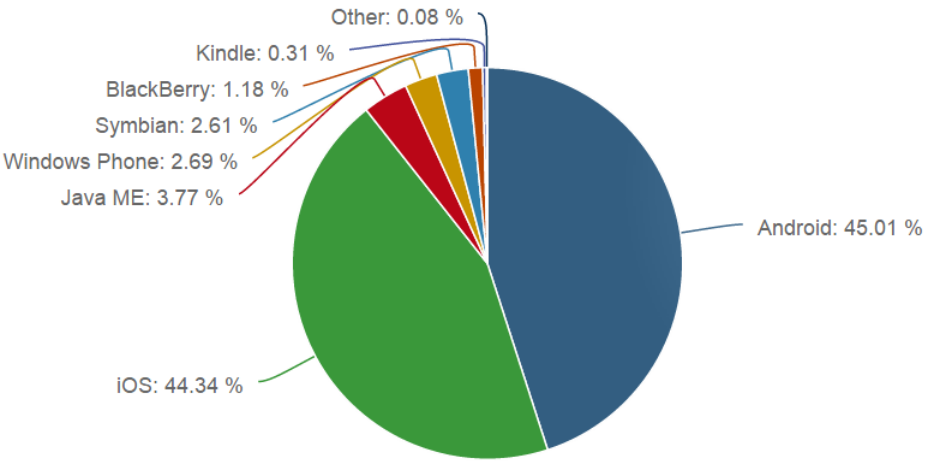



**Figure 2.10:** Global Mobile OS Market Share (August 2014) [7]

When studying the different MOSs, one must consider its strengths and weaknesses, not only from the technical standpoint but also from the commercial perspective. Figure 2.10 shows the market share of the various mobile platforms in terms of online usage, with Android devices browsing the Web more than iOS devices or any other MOS. However, both platforms stand out from all others. Following is the Java ME, Windows Phone and Symbian (which is supported by Accenture only until 2016 [51]). Java ME is specially created for feature phones [52] [53] (low-end mobile phones, like Nokia's Series 40), and other embedded systems. It can be seen that there are still many users that browse the Web on feature phones. And while Java ME can still be used for modern mobile phone (smartphone) development, it is not well-suited for rich and powerful applications or for targeting the latest smartphones [54]. Considering the market share of the smartphone OS in 2014, according to Strategy Analytics [55], Windows Phone emerges as the third most used platform for mobile devices, surpassing BlackBerry's proprietary system and other MOSs with very small market shares, as depicted in Table 2.2.

Given this, Android, iOS and Windows Phone are the only platforms considered hereinafter.

| Rank | OS Platform | Units |
|---|---|---|
| 1 | Android | 1,455 M |
| 2 | iOS | 373 M |
| 3 | Windows Phone | 59 M |
| 4 | Blackberry | 38 M |
| 5 | Symbian | 27 M |
| Others | | 8 M |

**Table 2.2:** Installed base of smartphones by OS as of 30 September 2014 [9]

**Android:** Android [56] is the world's most popular mobile OS and is currently developed by Google. Its main implementations are focused on smartphones and tablet computers. This MOS is defined as an open-source platform, enabling developers to use third-party tools (libraries) to create or optimize their

applications. Android apps can publish data that other apps can consume, broadcast notifications that other apps can subscribe to, and many other things, providing a great flexibility for developers. This open nature of Android is a significant advantage.

The market share of this MOS is also an advantage compared to all others. In the smartphone industry alone, over 84% of the users are using an Android device [57].

In terms of security, Android runs its applications in full isolation (apps are "sandboxed") and therefore does not allow them to interact with sensitive data or system files/resources, or to modify or collect information stored by other apps [58]. It also prevents them from accessing or modifying the OS kernel, greatly reducing the possibility of malicious apps obtaining administrator-level control over the device. Yet, within this isolation policy, apps can exceptionally examine each other's programming logic (but not the private data) and have shared access to the device's removable storage, like the SD card. Android also permits users to grant an application access to specific features [59].

Also an advantage is the simple publishing process of applications [60]. A developer can publish his apps within several hours and is only required to pay a one-time fee of twenty five dollars (US$25). However, this procedure has a low supervision, making Android more susceptible to malware.

A serious problem of Android is hardware and software fragmentation. Given its open nature, many manufacturers build several Android devices and so, there are many users using different OS versions or devices with significant differences between them, making it harder for developers to create new apps.

**iOS:** iOS [61], formerly iPhone OS, is a mobile OS developed by Apple Inc. intended only for Apple hardware such as iPhones, iPads, iPods Touch and Apple TVs. All these devices rely on onchip storage and therefore do not allow the use of SD cards.

Unlike the previous MOS, iOS is defined as a closed source platform. However, with its most recent release – iOS 8 – it has become a more open and Android-like platform, providing more freedom to developers. By using a new set of features - Extensibility [62] -, they can create widgets and also apps that can communicate with other apps, sharing data, features and functions between each other. While this relase makes the platform more open for developers, Android is still in the lead in terms of customization and tailor-made experience.

Nevertheless, these features are not yet broadly available. The most significant problem with iOS devices is the lack of support for external SD devices.

**Windows Phone:** Windows Phone [63], the successor of Windows Mobile, is a MOS developed by Microsoft Corporation exclusively for smartphones. Like iOS, it is defined as a closed source platform which means that there is a smaller ecosystem (less attractive to third-party developers) and thus less chance for innovation. Nevertheless, it is worth noting that this platform supports the use of SD cards.

Windows Phone has the lowest number of apps in the app store between all three: according to Statista [64], as of July 2014 there are approximately 1.3 million apps published for Android, around 1.2 million apps for iOS and about 300 thousand apps for Windows Phone [65].

Another disadvantage when developing apps for this smartphone OS is that the current level of technical support is not as strong as for Android and iOS.

## 2.3 Smart Cards

Section 2.1.3 presented and described several technologies and infrastructures that allow users to digitally sign data and/or transactions on their handsets to use a specific service/application and/or perform transactions. This is accomplished through the use of qualified digital signatures that are created on the SIM card of the handsets, which is also a smart card. One of the problems of these solutions is the low availability of mobile operators in the market that are capable of offering SIM cards with a usable cryptographic token. Of course there is always the possibility for the user to have two handsets, one with the operator's SIM card and the other with the cryptographic SIM card, although this requires the user to carry two devices. A Dual-SIM handset could also be an alternative, but there are not that many devices of this type available in the market.

To cope with this limitation, different kind of smart cards can be used to provide the needed cryptographic features without the user having to own a special SIM card.

This Section begins with an overview of smart cards, their architecture and main characteristics. Following this, the smart cards' operating systems and run time environments are covered in Section 2.3.1.A. Section 2.3.1.B details the existing formats of smart cards, which are used for different purposes. The communication with smart cards, namely the exchanged messages, is addressed in Section 2.3.2.

### 2.3.1 Overview

Currently, there are two types of cards in the market: the ones with chips, like smart cards, and the ones without chips, such as magnetic cards which contain a magnetic stripe where the data is written. Chip cards, or ICC are plastic cards with an embedded computer chip that is capable of storing or processing data. They can be categorized into memory cards or processor cards, according to the type of chip embedded in the card and its capabilities. Memory cards do not have any processing power, unlike processor cards which can store and process information, and can thus be used to provide authentication or identification of the user. They can also perform some on-card functions such as encryption and digital signing of data. Henceforth, only processor-based cards are herein considered.

The internal structure of smart cards is comprised of a CPU which is linked to the cards' Input/Output system which in turn is the point of communication with the cards. The CPU is also connected to three types of memory: the Read-Only Memory (ROM), the Random-Access Memory (RAM) and the Electrically Erasable Programmable Read-Only Memory (EEPROM). The first is where the operating system or runtime environments are placed, while the second serves as data storage to be used during computation. The latter is where applications and their persistent associated data are stored [66]. Figure 2.11 illustrates the described structure.

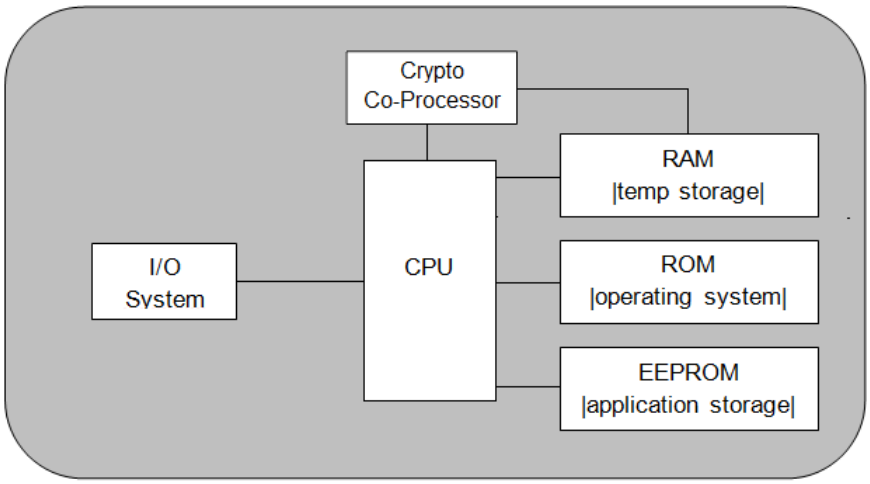



**Figure 2.11:** Smart card architecture

Typically, smart cards communicate with other systems, such as card readers, using the chip's contact interface with electrical signals (for contact cards) [67], or using the chip's remote contactless radio frequency interface (for contactless cards) [68].

In order to be considered secure, smart card systems must be tamper-resistant, which means that the data contained within the cards cannot be obtained successfully through physical attacks. To achieve this, most smart cards are comprised of anti-physical tampering mechanisms and cryptographic functions, like ciphering and hashing, which are implemented by hardware specific-processors in such a way that they are able to prevent side-channel attacks [69].

Smart cards are currently used as devices in providing different security solutions, like authentication mechanisms based on something the user has. In some solutions, smart cards are used to provide a biometric based authentication, performing the matching of the user's biometrics inside the card. In such cases, the card securely stores the user's sensitive data such as the authentication key or the biometric template, and ensures that this information cannot be externally accessed.

#### 2.3.1.A Smart Card Technologies

In order to implement a smart card application, it is essential to study the different types of operating systems or run time environments that exist in the market. They can be classified into native environments and interpreter-based environments. Native environments and the applications that run under them execute without an interpreter to translate the programs into the machine language of the target processor. Therefore, the application programs have to be generated in the machine language, which allows them to consume less energy and achieve faster response times. On the other hand, interpreter-based environments have an interpreter, allowing applications to be written in an interpreted programming language (e.g., Java). This type of environment also provides a way for applications to be implemented independently of the smartcard hardware, turning them into portable applications. Typically it also provides an application binary interface which implements the most essential functions used in smart card applications, speeding up their development and thus reducing development costs [70].

Java Card [71] is a well-known example of an interpreter-based environment developed by Sun Microsystems (now Oracle). Its technology enables small applications (called applets) to be run securely on a wide range of smart cards. Any smart card supporting Java Card technology, regardless of its vendor and underlying hardware, can run applets developed with Java Card technology. Additionally, the technology supports the secure installation of new applications after the smart card has been issued. When creating Java Card applications, developers can use a subset of the Java programming language. Java Card specifies the file format – Converted Applet (CAP) – that is runnable by the Java Card Virtual Machine (JCVM) and to which the applets must be translated before being deployed. However, it does not specify how to install them or communicate with them in a multi-applet smart card. For this, Global Platform can be used. Global Platform card specification [72] is a widely used standard for the management of the infrastructure of a smart card, including its installation, removal and other management tasks.

#### 2.3.1.B Smart Card Formats

Currently, there are different types of smart card formats used in different solutions [73]. The most commonly used format is specified in ISO/International Electrotechnical Commission (IEC) 7810 as ID-1, and is used for banking (e.g., credit cards) and ID cards. Figure 2.12 illustrates an example of such a card. There is a smaller version of this format – Visa Mini format – which is also used for payment systems and aims to meet customer demand for cards with the smallest possible dimensions [66]. The benefit of having a standardized market is that companies do not have to depend on specific suppliers while developing their systems [70].

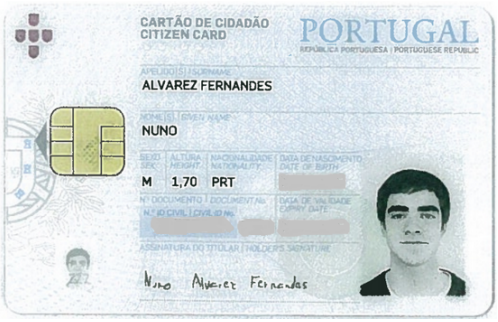



**Figure 2.12:** ID-1 smart card

The ID-000 (also known as plug-in) format is also very common. It refers to SIM cards and is thus used in mobile telecommunications applications, such as identification and authentication of users on mobile phones. This format is depicted in Figure 2.13. There are other smaller, less used formats defined by the ETSI Technical Standard (TS) 102 221 [74] that succeed the ID-000, such as the mini-Universal Integrated Circuit Card (UICC) or 3rd Form Factor (3FF) and the 4th Form Factor (4FF), used for instance by Apple for iPads and iPhones respectively.

**Figure 2.13:** SIM card

As a result of the growth in the mobile industry and the ongoing miniaturization trend, there is an increasing interest in designing applications for mobile devices, especially smartphones and tablets, making use of their hardware characteristics, such as the physical support for smart cards. However, some limitations exist. These devices typically do not support external digital interfaces, so they cannot communicate with ID-1 smart cards. SIM cards could be used since they have a built-in smart card and are supported by practically all mobile devices, yet most SIM cards are provided by mobile operators usually as closed platforms, making it impossible to install custom applications on it. Therefore, to provide applications that use smart cards, it is necessary to use a format that is capable of surpassing the previous limitations. Micro SD cards are flash memory cards in which smart cards can be embedded, and can thus be deployed with the strength of smart card chips. Most Android-based smartphones and tablets support them through a micro SD slot. This type of cards can be deployed as a closed element with a pre-installed application, disabling the user from installing additional applications, or as an open element, enabling the user to install several custom applications [70]. Additionally, users can access these cards from their personal computers using an embedded or external micro SD card reader, which can be very useful. An example of such cards is the GO-Trust® Secure microSD Java, shown in Figure 2.14.


**Figure 2.14:** GO-Trust® Secure microSD Java with embedded smart card

### 2.3.2 Communication

As previously stated, the communication with contact and contactless smart cards is specified in ISO/IEC 7816-4 [41] and 14443 [68] respectively. In the case of contact smart cards, the communication is performed through a half-duplex, bit-serial link, which means that, at any given time, there can only be one communicating party transmitting data and the data is received by the second party while the first party is transmitting. In order to avoid collisions, the master-slave principle is implemented, in which the terminal that communicates with the smart card is the master (as it always starts the communications) and the card is the slave. For contactless cards the communication is performed similarly, although the communication conditions are more difficult, as the data is transmitted via radio-frequency signals and thus subject to interference and loss [66].

The messages used to communicate with smart cards, regardless of their type, follow a specific format defined in ISO/IEC 7816-4, the APDU. When the master of the communication sends a command to the smart card, the message must be in the command APDU format.

A command APDU message contains from 0 to 255 bytes of data, representing a limited amount of data. Thus, to transfer larger amounts of data, the protocols used to communicate with smart cards must be optimized. There are also some smart cards that support a different APDU format - Extended APDU - allowing larger amounts of data to be sent in each message to the cards. In this document, only the common APDU format is considered since it is adequate for the data sent in the proposed solution of this thesis, which is presented in the following section.

## 2.4 Conclusion

Handwritten signatures are used worldwide for different purposes and have been the subject of forgery for several years. Digital signatures have been increasingly used, especially by companies, mainly because they allow the signing process to be digital. Furthermore, they carry significant improvements regarding the three security services that are provided by handwritten signatures – authentication, integrity and non-repudiation. Different protocols that implement this type of signatures exist: there are those based on biometrics, others that use smart cards and also the hybrids, which combine the strenghts of the previous two. There are also multiple signature formats – CMS, CAdES, XMLDSig, XAdES and PAdES – that differ in a number of aspects, such as the type of data they are intended

to sign, their flexibility, the type of signature they provide, and others. The choice of the format to use obviously depends on the requirements set by the countries or companies for the digital signatures.

One practical way to use these signatures is through the use of mobile devices. Today, these devices have a strong presence in everyday life, and with the continued growth of technology they are increasingly capable of performing tasks in a simple and secure way. There are several solutions that are able to perform the digital signing process in mobile devices. These solutions provide qualified digital signatures over messages, commands or other pieces of data for the purpose of using a particular service/application or completing some mobile transaction. The MSS solution proposed in [3] and the MNO-iMSS proposed in [6] stand out from all others for the benefits they offer, especially the MNO-iMSS which allows the user to use different handsets for generating the signatures, and is also independent from the MNOs. The problem with these solutions is that, besides not meeting the purpose of this thesis, particularly in providing qualified digital signatures for signing digital documents, they also use a SIM card to store the private data. This card, even though it is a secure smart card, has some disadvantages, the most significant of which is the fact that it depends on the mobile operator.

To address the limitation of the SIM card, a different kind of smart card is presented. This card has a specific technology and format which can solve the problem of the card's dependency, as well as contribute to a greater number of these cards in the market.

# 3

# Proposed Solution

## Contents

As seen in the related work, particularly in Section 2.1.3, there are several solutions that deliver qualified digital signatures via mobile devices, and which allow users to use a particular service/application or perform some mobile transaction. These solutions use the SIM card of those mobile devices (which is a smart card) as the security element performing the necessary cryptographic tasks. Besides the fact that none of these solutions provide a mobile system which focuses on providing qualified signatures exclusively for signing digital documents, a common problem with all of them is related with the use of the SIM card as the security element. Most of these cards are provided by mobile network operators and there are very few that are capable of offering cards with a usable cryptographic token, leaving the user with two unsatisfactory options: acquiring a dual SIM device (which has a low commercial presence in the mobile industry), with a common mobile SIM card and a SIM card provided by a security vendor which includes the cryptographic features; or owning two different devices, each having a SIM card implementing a different function.

A possible solution is the use of a "separate" smart card that seamlessly integrates the mobile devices as the security element, such as a mobile security card. For this a micro SD card which contains an integrated smart card element can be used. This solution has several advantages: the first is that it does not depend on the mobile operator, allowing the user to take advantage of cryptographic features on any mobile device covered by any mobile operator. Second, there is a wider availability and diversity of such cards in the market. And lastly, it has a stronger portability concept, since it is easier to remove/insert a micro SD card from/in mobile devices, such as laptops, smartphones and tablets. Additionally, almost all mobile devices from these three main categories are compatible with this card format through an SD or micro SD slot embedded in their hardware.

The solution herein consists of a mobile certification system based on a smart micro SD card. The private keys used in the signature processes are securely stored inside the card and can only be used by means of user authentication, minimizing the risk of being used by attackers, as they would need to gain access to the device and authenticate as the legitimate user. It also prevents malicious applications installed in the mobile devices from accessing the smart card.

This chapter is divided in 5 sections. Section 3.1 provides an overview of the proposed solution and of its main components, including their roles for carrying out the desired features. Section 3.2 presents the trust model, describing the assumptions taken by the proposed solution with respect to the main components. Following, Section 3.3 presents the proposed solution with further detail by describing its architecture. Section 3.4 analyzes the signature process performed by the solution. Section 3.5 concludes this chapter, summarizing all aspects of the proposed solution.

## 3.1 Overview

The goal of the proposed solution is to enable users to digitally sign any electronic document in their own mobile devices using secure cryptographic keys. To achieve this, a viable and reliable signing system is proposed. This system, instead of storing the sensitive data on the SIM card, stores them on a "separate" smart card which is supported by a mobile device through the SD slot and accessible only through PIN authentication. This PIN is known only by the legitimate user (owner of the card). To prevent online attacks a maximum number of PIN attempts is imposed by the smart card, after which the card is blocked. From this point on, the card can only be unblocked through a specific unblocking PIN, or Personal Unblocking Key (PUK), which is also stored in the card. This unblocking PIN can become blocked in the same way and if that happens, the card can no longer be used.

In order to assure its functionalities, the proposed solution (depicted in Figure 3.1) is comprised of two main components, the certification application installed on the user's mobile device and the signing module running on the smart card. The first handles, with the help of a software component, all communication with the smart card via APDU messages, sending all user requests to the card and receiving the respective responses. These requests may include the generation of cryptographic key pairs, signing of documents stored locally, and others. The second component is responsible for fulfilling the user requests, performing the necessary cryptographic operations. It also provides a secure environment to store sensitive data, such as the private keys, enforcing an authentication process upon any user who tries to access it, thus protecting the data from unauthorized accesses.

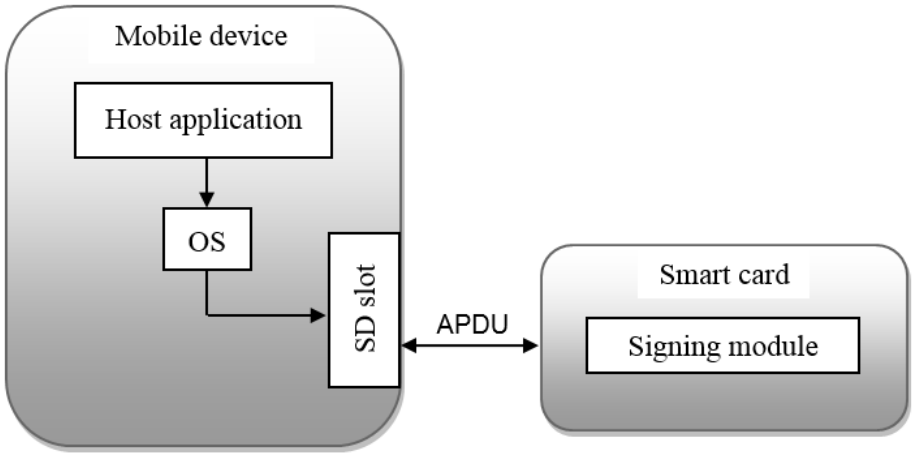



**Figure 3.1:** Certification system overview

## 3.2 Trust Model

The main components identified in Section 3.1 are executed over different environments. The host application runs on a mobile operating system and the smart card module over a smart card OS. In order to better understand how these components work, it is important to consider some properties of the systems in which they run, and to take certain assumptions about them. The herein considered assumptions are:

**Mobile app's assumptions:**

1. The mobile device's OS is trustworthy – The OS where the app is installed is reliable and not in control of a malicious user;
2. The user inserts in the mobile device the proper smart card;
3. The mobile device only sends data to the smart card if authorized by the user;
4. The correct hash is provided to the smart card - The hash of the document initially chosen by the user is indeed delivered to the card, and not replaced or tampered with by a malicious entity.

**Signing module's assumptions:**

1. The smart card has tamper-resistant capabilities;
2. The data stored within the signing module cannot be accessed by any other applet contained in the smart card;
3. The smart card is managed only by the trusted entity responsible for the creation of the private data in the card, such as the signature keys or public-key certificates.
4. The private keys are securely stored – The keys used to sign documents never leave the card;
5. The card owner is the only one who knows the PIN value to access the smart card – If the PIN value is used to access the card, then the authenticated user is the expected user.

## 3.3 Architecture

In order to have a better understanding of the proposed solution, its architecture is presented describing the necessary structure to build such a system. This architecture is designed in a way that it enables the integration of the certification system with different systems or applications. Thus, the goal is to create an interoperable solution that has a low-cost to the end user. This will become clearer as this section is analyzed.

To achieve its objectives, the proposed solution must make use of a smart card with a secure signing module and a middleware that ensures the proper communication between the smart card and the end-user application, as is depicted in Figure 3.2.

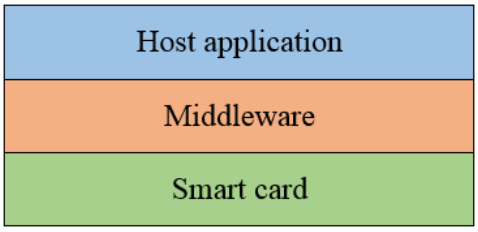



**Figure 3.2:** Proposed solution architecture

The Smart card layer has as its main requirements the secure storage of the private data inherent to the signature processes and the execution of all related cryptographic operations. The Middleware layer aims to build and maintain the communication between the card and the user's application by "speaking" the language of each one and performing the respective translations through the use of Application Programming Interfaces (APIs). The Host application layer is the application that interacts directly with the users and receives all their requests.

The following sections detail these three layers.

### 3.3.1 Smart Card Layer

The Smart Card layer is the lowest, or first, layer of the proposed architecture. It is charged with storing the user's information, which are the public-key certificates, PIN and PUK (or unblocking PIN) objects, and signature keys. It is also responsible for processing the cryptographic operations that are required to fulfill the requests coming from the Middleware layer.

All the material contained inside the smart card can be used by the authenticated user. However, some data can also be exported or retrieved from the card, depending on their nature. If it is private or sensitive information that is essential to ensure the proper functioning of the solution, then it can only be used (and not retrieved) by the authenticated user. This includes the PIN and PUK objects, as well as the signature keys. Such data can never leave the card, and thus can only be accessed by certain operations running inside it. This way the sensitive data are protected against external attacks. On the other hand, the data that are considered public can indeed be exported or retrieved from the card without any authentication, since there is no risk to the integrity of the solution. This includes the public-key certificates.

As for the cryptographic operations that the smart card makes available, these can perform all sorts of requests coming from the Middleware layer. These requests include: verification of the authentication PIN, generation of cryptographic key pairs, signing of documents stored locally, import, export and

removal of public-key certificates, update and unblock of the PIN value. If the first of these requests (PIN verification) is successful, then all other requests are allowed. The export of public-key certificates is the only one that can be executed whether the user is authenticated or not, as these objects are considered publicly accessible. Figure 3.3 depicts the structure and properties of the smart card.

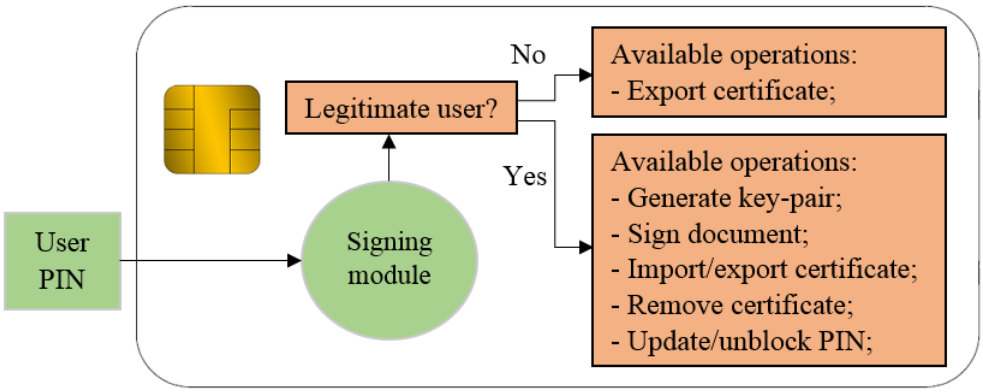



**Figure 3.3:** Smart card structure

### 3.3.2 Middleware Layer

In some systems the entire user certification is performed at the application level. However, in the case of the proposed solution, the most significant operations are performed by the smart card layer, since they handle sensitive data. The application layer does not actually have a direct communication with the smart card, since the goal is to achieve the interoperability capabilities previously mentioned. This is the reason for the existence of the second layer – the Middleware. Thus, the purpose of this layer is to manage the processes initiated by the application, and deliver a simple and standard interaction with the smart card layer by means of a standard API. The middleware is then considered as a software component located on the user side which hides the complexity of the certification processes' management as well as of the communication with the card.

When implementing the user application, the developer does not need to worry about the complexity of dealing with the smart card. The middleware encapsulates the details of the card's protocols and command structures, providing the developer the ability to use a standard API.

Figure 3.4 depicts the middleware structure which is divided in two sections, the Host application API, and the Smart card library.

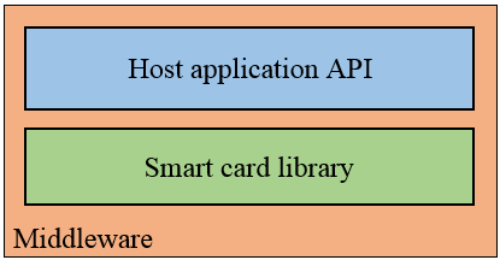



**Figure 3.4:** Middleware structure

The application API is the interface that the middleware provides to the user host application, and can be generic provided that it meets the smart card recommendations and standards. The PKCS#11 [75], also known as Cryptographic Token Interface Standard or Cryptoki, is such an interface. It is defined as an API designed for devices containing cryptographic information and capable of performing cryptographic operations (such as a smart card). It presents to applications a common, logical view of the cryptographic token.

The smart card library is tasked with performing all direct communication with the card. Each command to be sent to the card must be in accordance with the standard format and with the card's own requirements. The library is able to encapsulate the complexity of creating these commands and of handling the respective responses. Typically, this kind of libraries is provided by the smart card vendor along with the card.

When running the user application, each request that is made goes through the Middleware layer before reaching the card. The Middleware performs the needed data transformations. In this process all data related to the request, using the PKCS#11 specific data structures, is converted into the format that is understood by the smart card library. Then, some logic is performed, in particular the verification or validation of the multiple converted parameters that are going to be used to build the APDU commands to be sent to the smart card, so as to prevent, to the maximum extent possible, the sending of invalid parameters. The responses to these requests go through the reverse process. In a way, one can say that the middleware supervises each request, ensuring its integrity from the source (application) to its destination (card). This gives rise to the possibility of the smart card being used by any application that uses the PKCS#11 standard as the communication interface to interact with the cryptographic token.

### 3.3.3 Host Application Layer

The application layer is the last layer of the proposed solution. It represents the application with which users interact in order to accomplish their cryptographic requests. It uses the middleware previously described for transforming the user-supplied input data to the format that is accepted by the smart card layer. The structure of this application (depicted in Figure 3.5) comprises two components, the end-user

application, and the registration officer application.

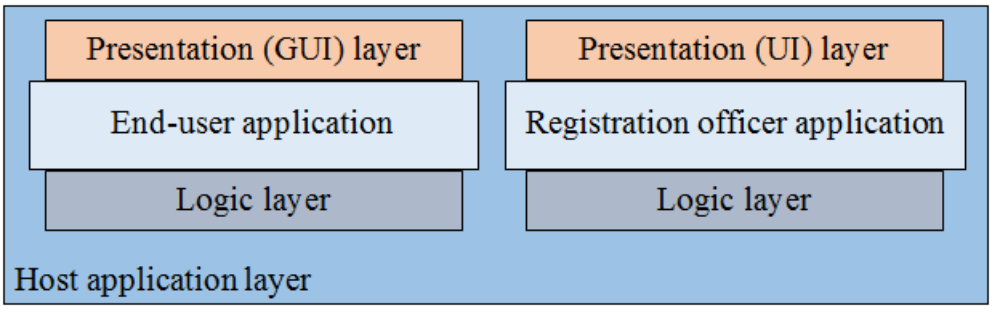



**Figure 3.5:** Host application structure

The end-user app enables users to perform various requests at the client level by means of a Graphical User Interface (GUI). This means that any operations other than the creation or removal of objects, i.e., read and (some) update operations, are carried out here. The operations that this app can perform are: digital signing of files, verification of signed files, changing and unlocking the PIN value, and reading the public-key certificates stored on the card. The logic associated with each operation is performed on the Logic layer.

The registration officer app is intended for more experienced users. These should be the managers of the smart card. It allows these users to perform the operations which are not allowed in the former app, i.e., create and delete operations, through an user interface. The objects to be created or deleted are items such as key-pairs and public-key certificates. Thus, the available operations are: import or removal of certificates, and key-pair generation.

Both applications use the PKCS#11 specification as the communication interface with the smart card (or in more detail, with the middleware that in turn interacts with the smart card).

## 3.4 Signature Functional Process

One of the most important cryptographic operations of the certification application is the signing of files. To better understand how this process is performed, the steps required to sign a document and the involved communication, a test-application was created on the laptop. This app allows the user to select any document stored locally and performs the digital signing of that document according to the X.509 standard. This includes the generation of asymmetric key pairs through a strong public-key cryptosystem, and also the generation of public-key certificates. The graphical interface of the developed test-app is depicted in Figure 3.6.

Hence, the signature functional process of the proposed solution involves the exchange of some messages and data, as well as some processing of the main components not only to perform the pro-

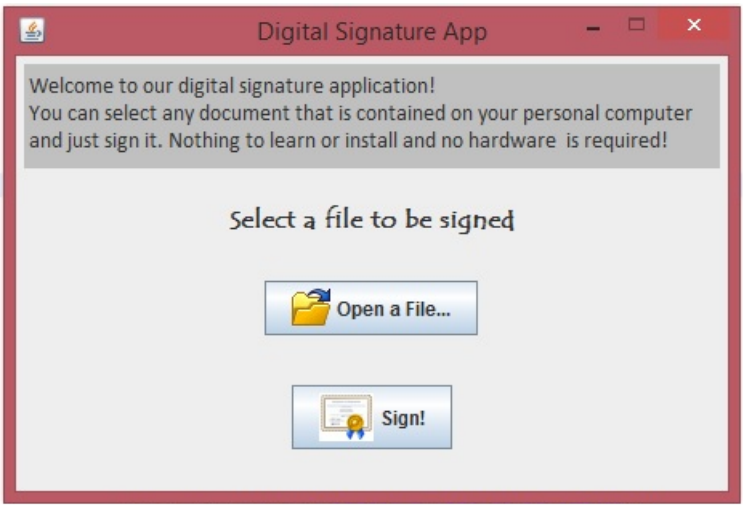



**Figure 3.6:** Application screen

posed functionalities, but also to confirm the identity of the signer and prevent any attackers or malicious applications from having the ability to sign documents. This process is depicted in Figure 3.7.

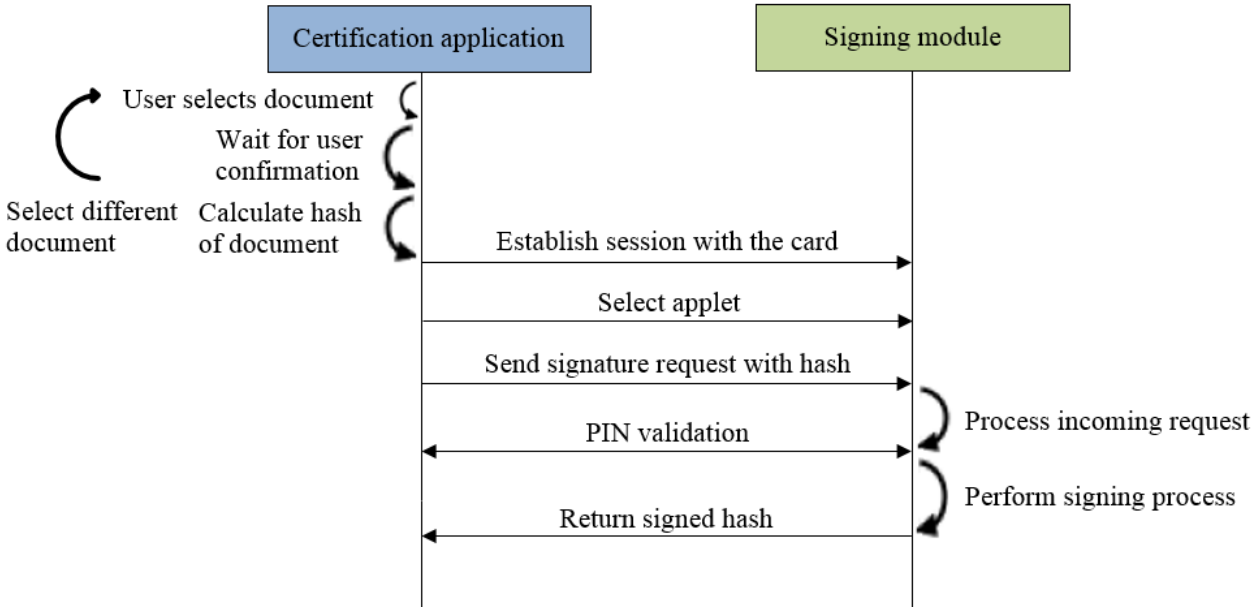



**Figure 3.7:** Components interaction in the signing process

As it can be seen, the process starts with the user selecting the document to be digitally signed. The certification app then calls the user's attention to the selected file so that he or she makes the proper decision: continue the process if the file is indeed the right one, or select a different file. If the user chooses to continue, the app calculates the hash of the document. Subsequently, the app establishes a session with the smart card and then selects the applet to be used (signing applet). As stated in Section 2.3.2, the smart card is only capable of receiving data with a length no greater than 255 bytes.

Considering the fact that a document can exceed this length, the app cannot send the document to the smart card. It could however split the document in pieces and send them in a chaining process, but that would have a significant performance cost. Instead, it sends the hash calculated from the document, which is far smaller, making the process much more efficient.

After receiving the signature request with the hash of the document, the signing applet processes the request, in which it requires the signer to submit a PIN value that only he or she knows. After receiving it, the applet checks if the PIN is correct. If it is not, the applet could be facing an unauthorized entity and denies the access. Consequently the user request is not fulfilled and the document is not digitally signed. In case the PIN is correct, the applet can assume with high confidence that the user who is trying to sign is the owner of the card. It then performs the necessary cryptographic operations, namely the encryption process of the previously sent hash (using a stored private signature key), and returns the signed hash to the application.

The certification app also has other functionalities that require a process similar to the one that was just described, i.e., that require authorization from the legitimate user. These features include importing or removing the public-key certificates, generating asymmetric key-pairs, and updating the smart card's authentication PIN. The first two are simple requests, performed in the same way as the signature request. The latter requires the user to submit the old PIN and the new one. Both values are sent to the applet and if the user is correctly authenticated, the applet updates the PIN, otherwise the operation is cancelled and the old PIN remains active.

## 3.5 Conclusion

In this chapter the proposed solution is presented. This solution consists on an electronic certification system that, with the help of a smart micro SD card as a secure environment for storing the sensitive data, is capable of digitally signing any electronic document stored on the user's mobile device.

The solution is composed of three layers. The first is the Smart card layer which aims to provide a secure environment for storing the sensitive data of the user, such as the signature keys, protecting them from unauthorized accesses. This layer also has the responsibility to perform the necessary cryptographic operations, such as signing a document. The second layer is the Middleware. This one is charged with simplifying the use of the certification system by providing a standard API, using the PKCS#11, that hides the complexity of communicating with the smart card. The third and last layer is the Host application, which represents the certification application that runs on the user's mobile device. It is the source of all user requests, and makes use of both the middleware and the smart card to accomplish them.

To properly design the system, the considered trust model is also described. This includes the

considered assumptions for each component of the system.

Regarding functionality, there are several operations that the signing module provides to the certification application, and which depend on the user is authenticated or not. One of the most relevant is the digital signing of files, which obviously requires the user to authenticate by entering his or her PIN value. After this, given that a file to be signed may have a rather large size, which easily surpasses 255 bytes (maximum size allowed in sending data to the card), the hash of the file is performed and sent instead. The module then signs the hash using the user's private signature key, and returns it to the application for future use.

The main advantage of the proposed solution in relation to other mobile signature solutions is that it allows to provide qualified signatures over electronic documents using a security token like the smart micro SD card that does not depend on the mobile operator. It offers the ability to sign documents "anytime, anywhere" in a large variety of devices through a common embedded hardware entry, namely an SD or micro SD slot.

# 4

# Solution Implementation

## Contents

There are several components that comprise the proposed solution. They interact with each other in order to provide the desired features. But for this to happen, technologies must be chosen that comply with the required properties. This chapter describes the technologies selected for each component, and the implementation details.

Section 4.1 describes the digital signature format used to create the digitally signed files. The implementation details of the signature applet are described in Section 4.2. Section 4.3 is intended to address the technologies used to implement the Middleware components, and the interaction between them. Section 4.4 describes the library used to apply the format chosen in Section 4.1. The implementation details regarding the certification application are described in Section 4.5. Section 4.6 concludes this chapter with some conclusions regarding the implementation of the solution.

## 4.1 Digital Signature Standard

The importance of digital signatures is clear for everyone who is concerned about trust and security in data storage and processing. From people and companies doing electronic business to non-profit organizations, everybody wants to have some reliable way to sign documents and verify signatures from third parties.

For the creation of reliable digital signatures to be possible, a proper signature format or standard is essential. It is required that the chosen standard ensures the integrity, authenticity and non-repudiation of the documents' contents, and also the possibility of long-term validation. The XML Advanced Electronic Signatures (XAdES) standard was selected because this specification has a number of advantages in relation to other advanced electronic signature formats such as CMS Advanced Electronic Signatures (CAdES) and PDF Advanced Electronic Signatures (PAdES): It allows to work on data of any kind, meaning it enables signing of any data, including PDF and binary. CAdES can also sign any data, but not PAdES; this only applies to Portable Document Format (PDF) documents, which is somewhat restrictive. However, XAdES is more flexible than CAdES since it does not operate on binary data, but on the XML Information Set, allowing it to work on subsets of the data and consequently having different ways of binding the signature and signed information.

In addition to all this, the possible formats were also discussed with the team of PDM&FC in order to figure out the most suitable one. It was agreed between us that XAdES should be part of the solution's technical requirements as the digital signature format, since they argued that, besides the format having the features previously stated, it was increasingly being used by new applications.

## 4.2 Signature Applet

Before starting to describe the smart card behavior, it makes sense to justify the choice of the card itself. Several cards were analyzed with respect to various features, such as: available security algorithms, security level (Common Criteria Evaluation Assurance Level (EAL)), EEPROM size, supported platforms, and Java Card and Global Platform versions. Within the six cards initially considered, three were selected: GO-Trust® Secure micro SD Java card, SP-TEK Secure Flash Micro SD card, and Swissbit® PS-100u Standard Edition (SE) Micro SD card, with the final choice falling on the first. All three were capable of performing practically the same security algorithms (with Rivest-Shamir-Adleman (RSA) being the most relevant one), all had high Common Criteria (CC) EAL (5+), and all had the Java Card and Global Platform accepted versions 2.2.2 and 2.1.1, respectively. However, GO-Trust's card was the one that had the largest EEPROM size (144 Kilobytes (kBs)), and was supported by multiple platforms such as Microsoft Windows, Linux and Android OS. Furthermore, this has also been the selected device in related works [70].

The first step towards the implementation of the proposed architecture is the development of the smart card behavior. For this, the technology chosen to program the card was the Java Card. This technology enables a certain level of abstraction over the smart card implementation, making it possible for the applications (called applets) to be independent of the card's underlying hardware or specific manufacturer. Another advantage of Java Card is the support to the installation of multiple applications in the same card, which allows the proposed solution to be deployed alongside other applications, thus contributing so that the user does not have to carry a card for each use. Furthermore, Java Card implements a firewall in order to keep the applets from accessing data that does not belong to them, thus ensuring data security. This property is compliant with the signing module's security assumption no. 2.

Regarding the creation of the applet, even though Java Card uses a precise subset of the Java programming language, there are specific features which must be taken into account. The first one is that, since the RAM is considerably scarce on smart cards, and also volatile and erased each time the smart card is unplugged, the objects are stored in persistent memory (EEPROM). Objects such as the PIN and PUK, the certificates, signing keys, and others. The downside of this type of memory is that, typically, the write access times are significantly longer than when using the RAM. Therefore, to achieve a good performance, any kind of processing writes only on the RAM, except during the storage or creation of the user's sensitive data, these are held in the EEPROM. This way, the solution is capable of providing a smart card with efficient response times. Another relevant aspect is making sure that the operations that handle persistent data inside the card are atomic, so that the system does not get corrupted in case the card is ever unplugged during a particular operation. For this, Java Card provides a transactional method which controls the persistence of objects, guaranteeing that either all or none

of the operations will have effect on the system. Thus, transactions are used when creating or storing persistent objects, ensuring the proper state of the system.

Also to be taken into account is the fact that the EEPROM space is shared not only by the persistent data but also by the executable code of the application, which means that the size of the application code influences how much persistent data can be stored. Consequently, the code must be written very efficiently so that there is no redundant and/or useless code. For instance, an operation which checks whether the PIN policies are satisfied. This type of operation must be performed every time the user wants to verify, update or unblock the PIN value. And since most of the security operations of the signature applet require the PIN to be verified, the verification of the PIN policies happens quite often. As such, its code is implemented only once in a centralized location, and is invoked when needed.

In order to load the developed applet into the smart card, a Java card loading tool provided by GO-Trust [76] was used, given that the Java Card technology does not specify any load mechanism. This tool interacts with the Applet manager of the Java card and, after performing a mutual authentication, sends the CAP file (the applet's file format) to the card, where the applets that are defined in the CAP file are created.

All communication with the applet is then performed using the methods provided by the smart card library – GTSDUpi – which encapsulates the handling, sending and receiving of the APDU commands.

## 4.3 Middleware

As previously stated in Section 3.3, the Middleware layer is responsible for providing a standard interface to the certification application, and for managing all requests that come from it, thus encapsulating the complexity of the communication with the smart card. In order to implement such a middleware, the structure proposed for the Middleware layer (depicted in Figure 3.4) is followed, which is divided in two main components: the Host application API and the Smart card library.

Starting with the application API, the selected interface standard was the PKCS#11. This is a well known and widely used API standard intended for cryptographic devices associated with a single user, such as a smart card, and supported on several operating systems. This specification comprises a large number of functions, but only the necessary ones need to be implemented. Besides the required cryptographic mechanisms, many functions related with the general-purpose usage were implemented, since they are needed to interact with the smart card layer. These general-purpose mechanisms include: session management, slot and token management, object management, and others. Table 4.1 contains a list of all implemented PKCS#11 functions.

Regarding the smart card library, this is the component that communicates directly with the smart card. It is provided by the card vendor (GO-Trust [76]) and is named GTSDUpi. This library provides

| Category | Function | Description |
|---|---|---|
| Slot and token management functions | C_GetSlotList | Obtains a list of slots in the system |
| | C_GetSlotInfo | Obtains information about a particular slot |
| | C_GetTokenInfo | Obtains information about a particular token |
| | C_SetPIN | Modifies the PIN of the current user |
| Session management functions | C_OpenSession | Opens a connection between an application and a particular token |
| | C_CloseSession | Closes a session |
| | C_Login | Logs into a token |
| Object management functions | C_CreateObject | Creates an object |
| | C_DestroyObject | Destroys an object |
| | C_GetAttributeValue | Obtains an attribute value of an object |
| | C_FindObjectsInit | Initializes an object search operation |
| | C_FindObjects | Continues an object search operation |
| Message digesting functions | C_DigestInit | Initializes a message-digesting operation |
| | C_Digest | Digests single-part data |
| | C_DigestUpdate | Continues a multiple-part digesting operation |
| Signing functions | C_SignInit | Initializes a signature operation |
| | C_Sign | Signs single-part data |
| Key management functions | C_GenerateKeyPair | Generates a public-key/private-key pair |

**Table 4.1:** PKCS#11, or Cryptoki, functions

its mechanisms under three distinct categories: connection, command and auxiliary functions. The first and third are responsible for handling requests related with slot, token and session management. Thus, they are specifically used in the beginning and end of each session with the smart card. If they are not performed with success, there can be no further exchange of messages with the card. The second category (command) is used for data transmission. If the session is successfully established with the smart card, these mechanisms are responsible for transporting the relevant data between the card and the middleware. As stated in subsection 2.3.2, the communication with the card is based on APDUs. Hence, each mechanism is responsible for building and transferring the respective APDUs to the smart card.

For the application to be able to communicate with the card, i.e., to make the connection between them feasible, a PKCS#11 module was required. A PKCS#11 module is a software library with a defined API that allows the access to a cryptographic hardware. It makes the mapping of the application to the card. More specifically, it integrates the smart card library functions with the PKCS#11 ones, so that each request made on the application API can be properly translated to the language of the card library and forwarded to the card, and vice versa. In many cases, the cryptographic hardware manufacturers provide PKCS#11 modules for their products. However, in the case of the GO-Trust® Secure micro SD Java card, no PKCS#11 module was provided, so it was necessary to build one.

The new module was then developed with the help of the IAIK PKCS#11 Wrapper for Java [8], a

library for the Java platform that enables the access to PKCS#11 modules from within Java. The need for such a wrapper was due to the fact that, as seen ahead, Java was the programming language selected to develop the certification application. Thus, the wrapper uses the Java Native Interface [77], which enables to access native applications and libraries written in other programming languages (such as C), in this case the PKCS#11 module of the smart card. Other libraries were also considered, such as the OpenSC-Java [78] and JCryptoki [79]. The first did not provide a level of documentation as complete as the IAIK Java Wrapper. The second was similar to IAIK, both offered good documentation and support base. However, the choice fell on the IAIK wrapper since it has more usage examples and the RSA Data Security published it as the implementation option [75]. Also, it seemed to gather greater popularity amongst developers. Another advantage is that it can be used for commercial purposes or redistributed for any other purposes.

Figure 4.1 depicts the previously explained components that comprise the developed middleware.

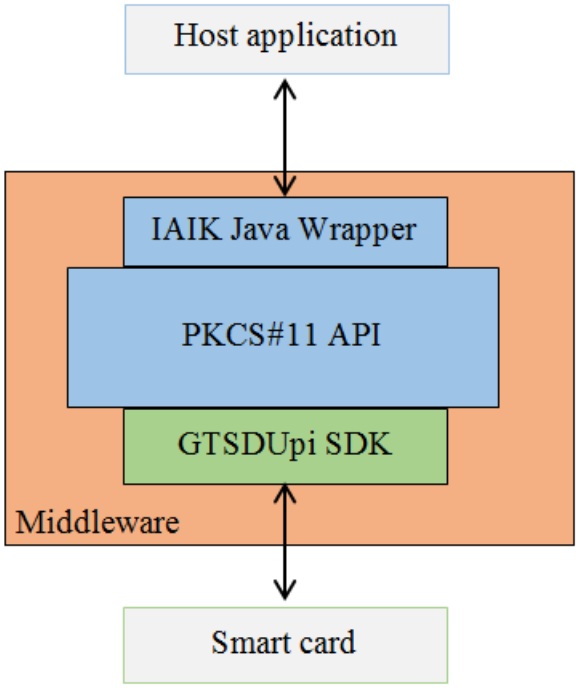



**Figure 4.1:** Middleware structure of the implemented solution

The wrapper developed by IAIK of Graz University of Technology is comprised of three layers, the Object Oriented (OO) Wrapper API for PKCS#11 for the Java platform, the (non-Object Oriented) Wrapper API for PKCS#11 for the Java platform and the Native Module of the Wrapper implemented in the C programming language. Figure 4.2 depicts the layer model of the library.

The highest layer (OO Wrapper API for PKCS#11) delivers a straightforward mapping of the PKCS#11 standard to a set of classes and interfaces. The middle layer consists of a collection of Java classes and interfaces that reflect the PKCS#11 API. This differs from the upper layer in the sense that this is not an

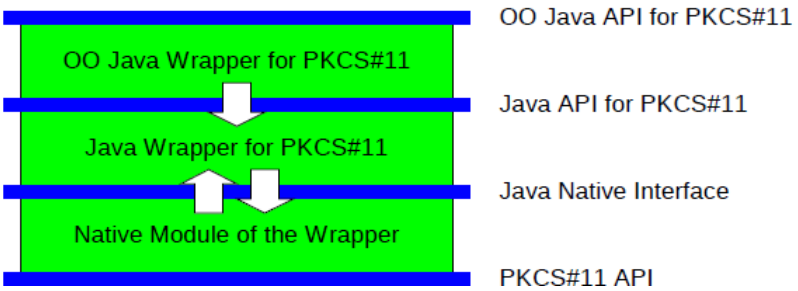



**Figure 4.2:** Layer model of IAIK Java Wrapper (Adapted from [8])

object oriented approach, it is just a straightforward mapping of the data structures to Java. The lowest layer is tasked with translating the Java data structures coming from the above layer to native PKCS#11 data structures, and vice versa. It then provides access to the functions defined by PKCS#11.

In order to create the PKCS#11 module for the developed application, the lowest layer (Native Module of the Wrapper) underwent some modifications and was recreated by integrating the card's library (GTSDUpi) with the layer. This way, after the layer finishes translating the Java data structures to the native PKCS#11 data structures, it translates these to the GTSDUpi data structures and performs all the necessary logic to adequately prepare them to be transported to the smart card. Then, it forwards them to the smart card using the respective GTSDUpi functions.

With this module the application can make a call to a PKCS#11 function in Java, which is converted to the native PKCS#11 format. Then this is converted again to the format accepted by the GTSDUpi library, which finally gets the initial request to the smart card.

The development of the PKCS#11 module's code was carried out using the C programming language syntax, so as to facilitate the integration of the last layer - Native Module of the Wrapper (already coded in C) - with the smart card's library (GTSDUpi), and also to provide the PKCS#11 interface in C. In compiled form, this module comes in the form of a Dynamic Link Library (DLL) or shared library.

Every application that wishes to use this solution's cryptographic token can simply load this module and benefit from all cryptographic operations made available by the smart card.

## 4.4 XAdES Library

In order to provide a proper implementation of the chosen signature format (XAdES), it is necessary to select an appropriate library. Several options were considered: the IAIK XAdES add-on for XML Security Toolkit (XSECT) [80], which is a library that enables the creation of advanced electronic signatures that are compliant with the EU directive on electronic signatures; and the XAdES4j [81] which is the result of a master thesis. It consists of a Java library which also allows the creation and verification of XAdES signatures. Both libraries treated the smart card as an object representing a PKCS#11 key store,

and the card is not meant to be handled that way. As a result, they revealed several incompatibilities when trying to perform the security operations over the smart card, like the digital signing of a document or generation of a key-pair, and so both of them were discarded.

As a result of the previous incompatibilities, another library was considered - JDigiDoc [82]. This is also a Java library designed for applications that handle digital signatures and their verification, using smart cards or other supported cryptographic tokens. It was tested in the same way as the previous libraries, and showed no incompatibilities when interacting with the smart card. Another great advantage of this library is that it is configured for integration with the IAIK Java Wrapper. For these reasons, JDigiDoc was chosen to be the XAdES library.

By using the library's functionalities, each digitally signed file is created in a specific format, the "DigiDoc format" (with .ddoc or .bdoc file extensions), and is compliant to XAdES technical standard published by ETSI. Between the two available formats, the BDOC was chosen because, according to the library's documentation [83], this is the newer format, recommended for new signatures. With BDOC, the original data files (which were signed) as well as the signatures are encapsulated within a ZIP container, which is based on ETSI TS 102 918 [84] called Associated Signature Containers (ASiC). In the case of BDOC files, the container type used is Extended (ASiC-E). This is a ZIP file with the following objects:

- A file called "mimetype", which contains only one value: application/vnd.etsi.asic-e+zip;
- The data files in their original format;
- A META-INF subdirectory consisting of:
  - "manifest.xml" – a file that contains a list of all folders and files in the container, except for the "mimetype" file and the files in META-INF subdirectory;
  - "signatures*.xml" – one file for each signature. It also incorporates certificate's info, validity confirmation and meta-data about the signer. The "*" denotes the sequence number of a signature (starting from zero);

## 4.5 Certification Application

After the development of the smart card and middleware layers, the next and final stage was the creation of the host application. This is the component that interacts directly with users, enabling them to benefit from all the features described in the preceding sections. However, there are features that should not be performed by the so called common users, but rather by experienced users or by the managers of the card itself, as they are features that deal with sensitive data. Therefore, the interaction with the card was divided into two applications: the client application and the registration officer application.

The programming language selected to develop both applications was Java, capable of running on multiple mobile platforms such as Windows and Android. To perform user requests, both apps make use of the IAIK Java Wrapper presented in Section 4.3. This library is comprised of two main components: the Java component and the native component, as depicted in the previous Figure 4.2. The apps do not need to access the native part itself; they only use the Java classes and interfaces of the IAIK Wrapper, particularly its higher layer (OO Java Wrapper for PKCS#11). Internally, the Java component uses the native component to connect to the PKCS#11 module of the smart card. This way, the Java apps are able to hide the complexity that exists in dealing with the details that are needed to create the data structures to be sent to the lower layers.

Figure 4.3 depicts the architecture of software and hardware components that are used when using any of the certification applications. More specifically, each box belonging to the two left-hand columns represents a software component that is used by either the Java application or the PKCS#11 module. The rigthmost column has only hardware components. Each arrow represents the communication (data flow) that is performed between the respective components.

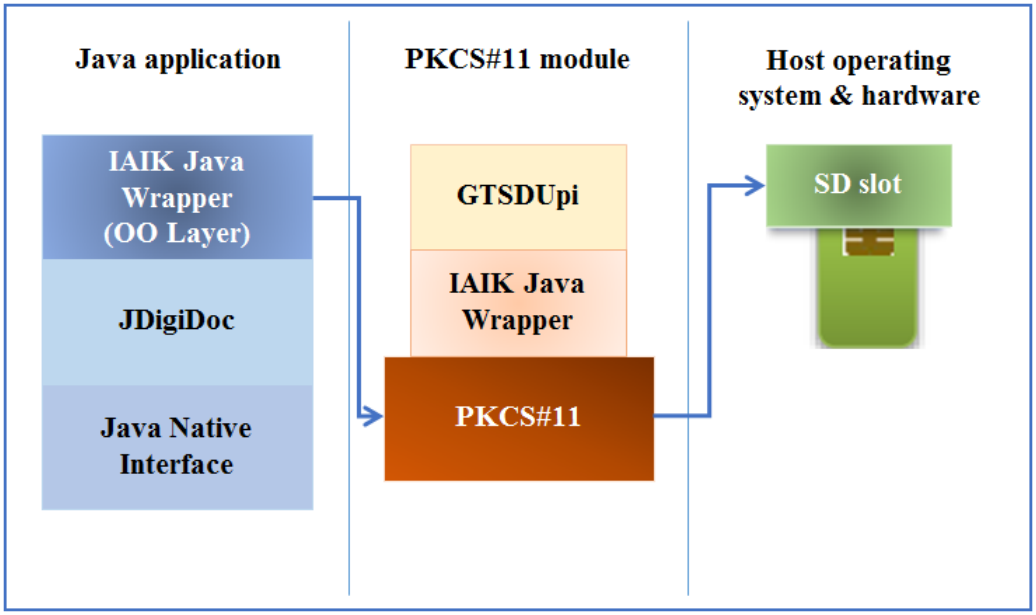



**Figure 4.3:** Architecture of certification applications

### 4.5.1 Client Application

This is the application that was developed for the “common” users. It is thus addressed for users that are not intended to order the creation or removal of sensitive information, such as the signature keys. Instead, they can perform all operations associated with reading that sensitive data, such as requesting to read a public-key certificate, and a limited number of operations related with updating the data, such

as requesting the modification of the PIN value. These users do not need to concern themselves about how the private data was created or is maintained; that is the responsability of the registration officer as it will be seen ahead. The users' focus should only be placed on using the available services, which are the following:

- Connect/Disconnect to/from the smart card;
- Digitally sign any file stored on the mobile device;
- Verify any signed file stored on the mobile device;
- Change the PIN value;
- Read the public-key certificates stored on the smart card;
- Unblock the PIN value;

This app has a GUI designed to guide the user while performing the cryptographic operations. This interface was built using the Swing [85] technology. This is a widget toolkit for Java that provides a sophisticated, and also platform-independent, set of GUI components, allowing users to interact with the application through graphical elements, such as icons and buttons. Thus, the interface becomes a user friendly way to interact with these users and exchange any needed information, including private information (e.g., authentication PIN value).

The Java application supporting the GUI is responsible for performing the validation of the entire data entered by the user, so as to reject any data that is invalid. This prevents any cryptographic process from ever being initiated, allowing the system to remain stable and saving it from unnecessary computational work that would be performed from the middleware to the smart card. Its main function is therefore to manage the execution of the user requests that ultimately reach the smart card, and to present the information of the corresponding responses to the user.

The created GUI consists of a main window where the user can connect to the card and carry out the operations he has at his disposal. In turn, the main window includes several sub-windows that are used for the user to select or enter the parameters necessary for conducting the operations. Some sub-windows are also used to provide feedback about the operations during their course. The main interface is depicted in Figure 4.4, while some of the sub-windows can be seen in Appendix B.

### 4.5.2 Registration Officer Application

This is the application that was created specifically for the administrative users. These are users which are qualified or authorized to create and manage sensitive or private data, such as signature key-pairs. They are also the ones that import objects, like public-key certificates, to the card, since these


**Figure 4.4:** Client Application main screen

certificates will be unequivocally associated with the card user, making it necessary to do this securely. Thus, they are allowed to perform all sort of operations associated with the creation and removal of objects.

Since the Registration Officer app is designed for more experienced users, it presents a command-line interface in which the user enters the desired command and associated parameters in order to create, delete or import a particular object. The user input data are also subject to validation, so as to prevent the app from making invalid or unnecessary requests to the system. Taking as an example the generation of a key-pair: to be able to execute this operation, the user needs to enter the respective pre-defined command with the proper parameters - the name of the PKCS#11 module, the type of key-pair (signature or authentication) and the key-bit size (1024 or 2048 bits). Besides the validation of the parameters, this process is obviously subject to authentication through a PIN value. This holds true for each operation that the user tries to perform, thus managing the security of the application. The interface

for this app is depicted in Figure 4.5 with the stated example.

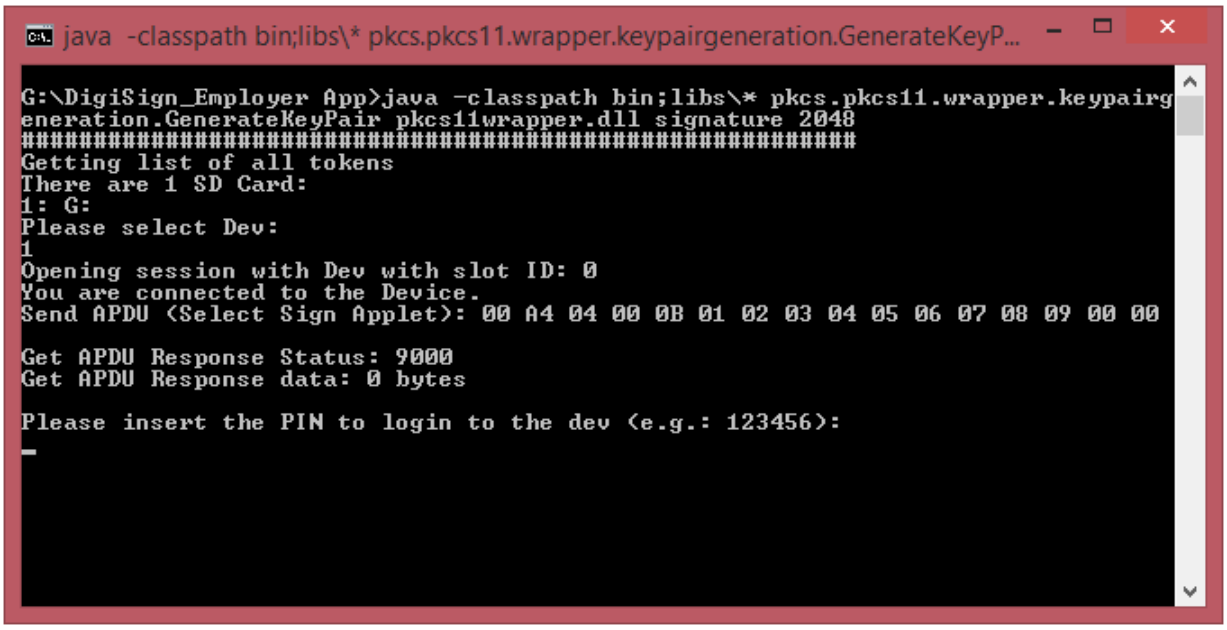



**Figure 4.5:** Registration Officer Application interface

The set of commands available for the user are:

- Generate an RSA key-pair with specific key size;
- Import a public-key certificate stored on the device in which the app is executed;
- Update a public-key certificate stored on the smart card;

## 4.6 Conclusion

This chapter describes the selected technologies and also the implementation details in order to fulfill the proposed solution. This allows achieving several desired properties.

The definition of XAdES as the digital signature format allows the solution to have a very flexible format, and which provides the authentication, non-repudiation and integrity of the signed documents' contents. This advanced format also provides the possibility of long-term validation.

In relation to the smart card behavior, the applet is implemented using the Java Card technology. This allows the application to be installed on the cards independently of their underlying hardware or manufacturer. It also enables multiple applications to be deployed on the same card, which means that the user is not required to carry a card specific to the solution.

In order for the certification application to be able to communicate with the smart card, a middleware is developed which manages the user requests, encapsulating the complexity of that communication. It

also provides, with the help of the IAIK Java Wrapper, an open-source standard interface to the application – PKCS#11 –, which causes the solution to be interoperable, i.e., any application that is able to interact via the PKCS#11 standard can take advantage of the developed smart card app.

Regarding the library that implements the XAdES format, the choice falls on JDigiDoc. This is a Java library that is capable of providing digitally signed files that conform to XAdES technical standard published by ETSI. It also provides these signed files in a specific format (BDOC) which is a ZIP container that is based on ASiC.

In relation to the certification application, it is divided into two apps with different purposes: one intended for the basic users, and another designed for users with more experience. The first is developed using the Java language and Swing technology, while the latter uses only Java. Both use the IAIK Java Wrapper to hide the complexity of dealing with the smart card, and the JDigiDoc library to provide proper digitally signed files and verify them.

# 5

# Solution Evaluation

## Contents

This chapter considers not only the state of the art, but also the design and implementation details to evaluate the features and properties of the developed solution. Section 5.1 compares the implemented solution with the existing solutions in relation to some of its features. Section 5.2 provides a security analysis of the developed solution. Section 5.3 provides the results of the performance tests carried out on the solution. Section 5.4 describes the usability tests performed to assess the solution from the user's perspective. Section 5.5 describes how the requirements (defined in Section 1.4) are met by the design and implementation details of the developed solution. Section 5.6 concludes this chapter by summarizing the features and properties provided by the solution, and how these fit into the objectives of this thesis.

## 5.1 Solution Comparison

It is important to compare the proposed solution with the existing solutions. The main features considered in this evaluation are the ones that are discussed in Section 2.1.3, namely the involved technologies and signature platforms.

Table 5.1 complements the table presented at the end of Section 2.1.3 by adding a row for the proposed solution, as well as new features that were not present in the existing solutions, thus enabling a better understanding of its comparison with the existing solutions.

| Solutions \ Features | Asymmetric cryptograpy | Device OS-independent | Smart card | SIM card-based | Handheld and SIM card-based | Services-based | MicroSD card-based | Security element | Qualified signature | SSCD |
|---|---|---|---|---|---|---|---|---|---|---|
| **WIM** | X | X | X | X | | | | $X_4$ | X | X |
| **USAT-I** | X | X | X | X | | | | $X_4$ | X | X |
| **SATSA** | X | $X_1$ | X | | X | | | $X_4$ | X | X |
| **MSS** | X | X | $X_2$ | | | X | | $X_4$ | $X_2$ | $X_2$ |
| **MSAU** | X | $X_1$ | X | | X | | | $X_4$ | X | X |
| **SPMS** | X | $X_1$ | X | | X | | | $X_4$ | X | X |
| **MNO-iMSS** | X | X | $X_2$ | | | X | | $X_4$ | $X_2$ | $X_2$ |
| **Proposed solution** | X | $X_3$ | X | | | | X | $X_5$ | X | X |

**Table 5.1:** Feature-based comparison including proposed solution

Notes:

$X_1$ - Only if the device's OS supports Java ME

$X_2$ - It depends on the device-side technology

$X_3$ - With the exception of iOS

$X_4$ - SIM card

$X_5$ - Smart micro SD card

The implemented solution is similar to the other solutions, but differs in key aspects. It has the advantage of being based on a smart card embedded on a micro SD card, meaning that the solution does not depend on a SIM card, like all the others do. This property causes the solution to use a security element which not only has a greater availability on the market but also is independent of the network operator. The developed solution also has an improved concept of portability, since it makes use of a smart card type that is easier to integrate into mobile devices. Plus, this card is widely supported in the market, as all operating systems provide support for SD devices, either internally or externally, with the exception of iOS. Furthermore, the solution does not involve the use of outdated or unsupported technologies (like in WIM), or external entities (like in the MSS-based solutions) to deliver qualified digital signatures.

## 5.2 Security Analysis

Another relevant part of the evaluation of the proposed solution is the security analysis. This aims at analyzing the main components of the solution, including the strengths and weaknesses or potential points of failure in its use. From this analysis one can define the limitations of the solution security-wise and also what precautions to take with it, providing an insight into the global security of the system.

Starting with the Java Card system, it is the source of all security features of the solution, because it not only stores the private information of the user, but also carries out the essential operations required to fulfill any requests that originate in the certification application. Since Java Card provides a secure environment for its applications to run, the user's sensitive information is securely protected against logical attacks targeting the card. It is also protected against physical attacks due to the smart card's tamper-resistant capabilities. Even if the user loses the smart card, or the card is stolen, a malicious person cannot have access to the information contained within the card, as this data is kept inside Java applets which, if well implemented, ensure that the private data never leaves the card. If this data were to be compromised, the entire security of the solution would also be compromised.

The signature applet is responsible for assuring the proper storage, management and use of the sensitive information. The operations that require the use of this information (e.g., a private signature key), such as the digital signing of data, always require the local authentication of the user. This authentication is managed by the applet, forcing the user to perform an authentication with the correct PIN value. This way, the system is able to ensure the user's presence at the time of the digital signing process. If an attacker gains possession of the smart card, he or she cannot actually use it because he or she does not know the PIN value of the legitimate user. Obviously the security of the PIN is not infallible. There is

always the possibility that it is guessed, and in this case the more digits the PIN has, the harder it is to guess it. In any case, this is a highly unlikely event, not only due to the amount of existing combinations, but also due to the established limit of PIN attempts (three times).

It is also important to consider the vulnerable point present in the connection between the smart card and the certification application, in particular regarding the digital signing operation of the solution. This operation involves the selection of a document by the user and consequent hash, both made in the certification application. Then this hash is provided to the card which, as previously stated, signs it in a secure and reliable way. However, the sending of the hash to the card is vulnerable in the sense that it does not assure unequivocally that the hash does not belong to another document. There could be the case where an attacker could find a way to tamper with or replace the user's hash with his own without the user noticing, causing him to sign a malicious hash. Thus, providing the correct hash to the card is assured by the certification application, which is an assumption required to ensure the proper functioning of the solution.

In relation to the certification application, it is tasked with providing to users all the operations that the proposed solution can perform, and presenting their results. Therefore, all user requests are initiated in this component. It runs on an operating system, such as Windows or Android, which is also a vulnerable point, and subject to attacks. If a malicious user were to gain control of the user's OS without him noticing, and if the certification app was used in such a system, the user requests could be tampered with, possibly leading to the discovery of sensitive data, such as the authentication PIN. Thus, the user needs to rely on the OS with which he is interacting, which means that the OS must be trustworthy. This is also a necessary assumption to make sure that the solution can operate securely.

Neverthless, the private keys stored on the smart card can never be retrieved by the attacker.

## 5.3 Solution Performance

One factor that can negatively affect the usability of a system is having a poor performance. Thus, it is also necessary to analyze the behavior of the solution and how it conducts its most commonly used operations, especially when communicating with the smart card.

To perform this evaluation, performance tests were carried out. These were made using a software application called Process Explorer [86], a process management utility that provides detailed information about a particular process, in this case the certification application. The tests were accomplished over an Intel(R) Core(TM) i7-4700HQ CPU running at 2.40 Gigahertz (GHz), with 8 Gigabytes (GBs) of memory, and with the Microsoft Windows 8.1 as operating system, runnning the standard processes inherent to the execution of the OS. The smart card was physically attached to the device. Several measures (at least six) were taken at different times. The data herein presented represents their average value.

The tests consist on the solution, specifically the client app, performing several operations, many of which involve making requests to the smart card for writing or creating data, and for reading data. These requests include: connecting to the smart card, digitally signing files, verifying signed files, reading public-key certificates, and changing the PIN authentication value, in that order. All operations were performed with approximately 6 seconds of interval from one another, with the smart card directly connected to the computer on which the solution runs. The first operation always starts with the solution already running, i.e., in a steady state.

During the execution of the operations, three relevant performance attributes are assessed: the CPU usage, the required memory (private bytes), and the amount of input/output data. The results are depicted in Figure 5.1.

The first graph represents the evolution of the CPU usage in percentage terms when performing the previously stated operations. It can be seen that this usage reaches its peak at the moment of the digital signature operation, with approximately 10% of CPU usage. This is a not a high value, yet it is the operation that causes the most CPU consumption. This is because the operation is comprised of many tasks that are carried out by the application: it performs the hash of the file that was selected by the user and sends it to the smart card in order to be signed. Then, upon the reception of the signed hash, the application requests to the card the export of the public-key certificate that is associated with the used signature key, so as to incorporate its information on the signed file, thus finalizing the signing process. There are also other operations that require some processing by the CPU, however not as much as the previous operation. And then there are those whose CPU usage is not even visible, since the needed CPU is very slight.

The second graph shows the behavior of the memory requested by the application to the system, specifically the private bytes. These refer to the amount of memory that the process (the application) has asked for. It can be observed that once again it is the digital signature operation that requires the largest amount of the resource, approximately 10 Megabytes (MB), because of the reasons previously stated. Following is the verification operation, which requires that the application validates the digital signature and the structure of the signed document as well. Then again there are operations that trigger only little variation on the memory of the system compared with the previous ones. Therefore, none of the operations requires more than 10 MB of memory. The total memory requested by the application throughout its cycle adds up to approximately 58 MB.

The third graph depicts the input-output operations that happen through the file system where the application runs, namely the creation, writing, reading and/or deleting of data. The operation corresponding to the verification of the signed file stands out due to the reading of the signed file by the application, and the huge amount of validation tests that it performs, searching for any validation warnings or errors that the signed file may have.

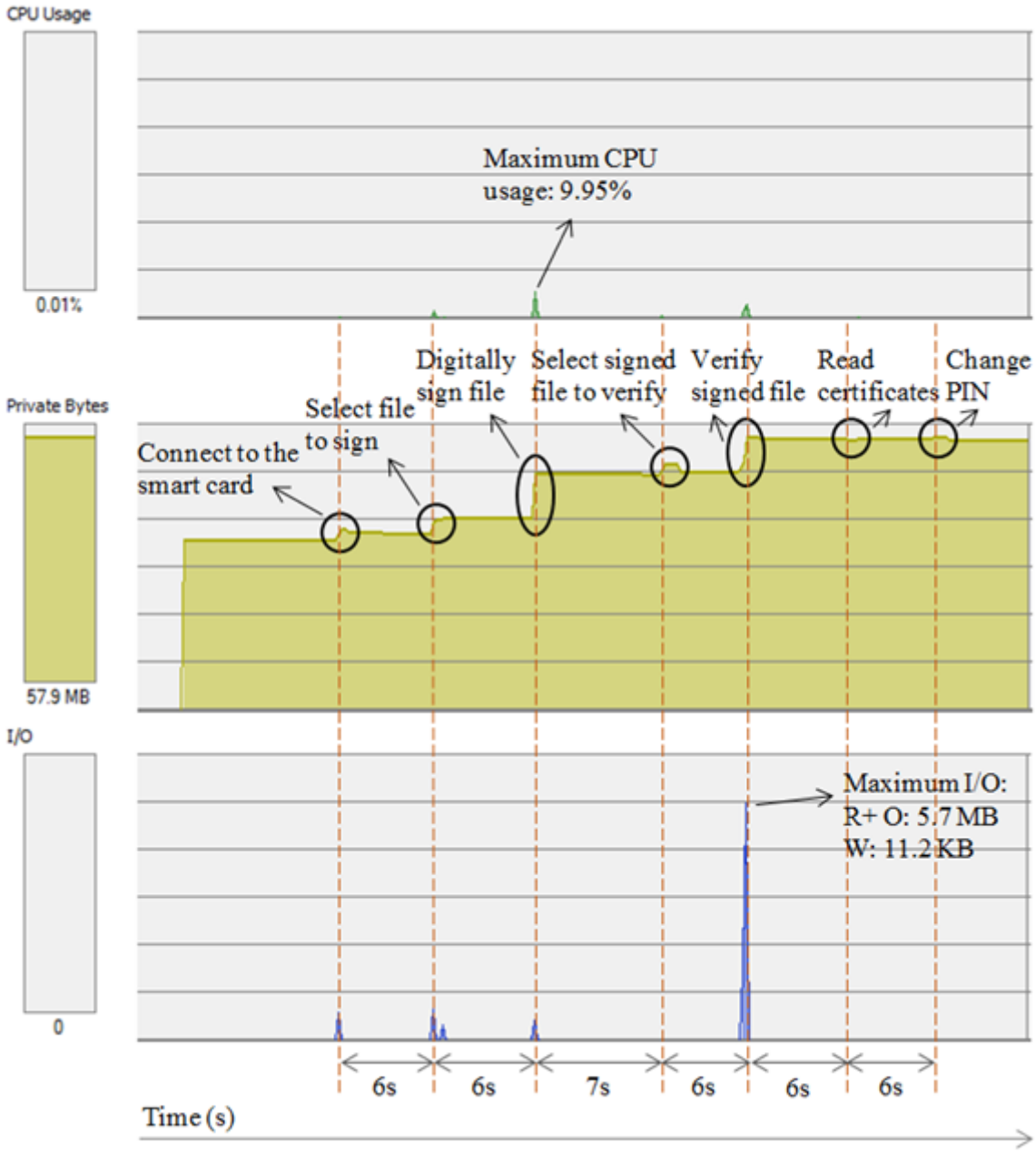



**Figure 5.1:** Solution performance tests

## 5.4 Usability Tests

In order to have a practical evaluation of the solution, usability tests were performed. These tests ensure the usability and viability of the system, and also provide a direct input on how real users use the developed system.

The tests were performed not only on the client application, but also on the registration officer ap-

plication, so as to achieve a thorough and complete evaluation of the system. For both applications, test cases were developed, each comprising a series of tasks for the user to do, with the smart card physically attached to a laptop. Twenty (20) people took part in the tests, in which the majority were employees (technical and non-technical) belonging to PDM&FC, providing greater reliability to the results. Following are the test cases used in this evaluation.

| Scenario | Test steps | Expected result |
|---|---|---|
| 1. Perform a digital signature using the signature key of the smart card, so as to guarantee the authenticity, non-repudiation and integrity of the signed file. | - Connect to the smart card (if not already);<br>- Browse filesystem and select a random file (e.g., .pdf, .txt, .doc);<br>- Digitally sign the file;<br>- Enter the proper authentication PIN; | The selected file should be signed with success. |
| 2. Confirm that the signature process was well conducted by validating or verifying the digitally signed file. | - Connect to the smart card (if not already);<br>- Browse filesystem (go to the same folder where you chose the file to sign) and select the newly generated signed file (*.bdoc);<br>- Verify the file; | The verification should present some warnings. However, its result should still be valid (with warnings). |
| 3. Read the public-key certificates that are stored on the smart card. | - Connect to the smart card (if not already);<br>- Read the certificates; | The information of the certificates (if any) should be presented correctly. |
| 4. Unblock the authentication PIN. | - Connect to the smart card (if not already);<br>- Unblock PIN;<br>- Enter the proper unblock PIN (or PUK): 123321; | The operation should fail, as the authentication PIN is not blocked. |
| 5. Change the authentication PIN value; | - Connect to the smart card (if not already);<br>- Change PIN value;<br>- Enter the proper value of the current PIN;<br>- Enter the new PIN value; | The authentication PIN should be modified with success; |
| 6. Block the authentication PIN by performing a digital signature with the wrong PIN value. | - Connect to the smart card (if not already);<br>- Browse filesystem and select a file;<br>- Digitally sign the file;<br>- Enter the old authentication PIN;<br>- Repeat the previous step three more times to block the PIN; | The operation should fail, since the proper authentication PIN is not being provided. Afterwards, the authentication PIN should be blocked. |
| 7. Unblock the authentication PIN. | - Connect to the smart card (if not already);<br>- Unblock PIN;<br>- Enter the proper unblock PIN (or PUK): 123321;<br>- Disconnect from the smart card; | The authentication PIN should be unblocked with success. |

**Figure 5.2:** Test cases for usability testing of Client application

| Scenario | Test steps | Expected result |
|---|---|---|
| 1. Generate a pair of keys for an authentication certificate inside the smart card. | - Run the GenerateKeyPair.java function with the help of the command defined in the README.txt file; | The specified key pair should be generated with success. |
| 2. Import a public-key certificate into the secure smart card. | - Run the ImportCertificate.java function with the help of the command defined in the README.txt file; | The specified certificate should be imported with success. |
| 3. Manage a certificate stored on the smart card by updating a certificate. | - Run the UpdateCertificate.java function with the help of the command defined in the README.txt file; | The specified certificate should be replaced with success. |

**Figure 5.3:** Test cases for usability testing of Registration Officer application

Once the test cases were completed, the users answered a survey regarding the various aspects of the solution. This survey is presented in Appendix C, and a summary of its results can be found in Appendix D.

From the test results it can be seen that, through the various interactions with the solution, the majority of the users felt that it is easy to use, thus providing a good user experience. The results are depicted in Figure 5.4.

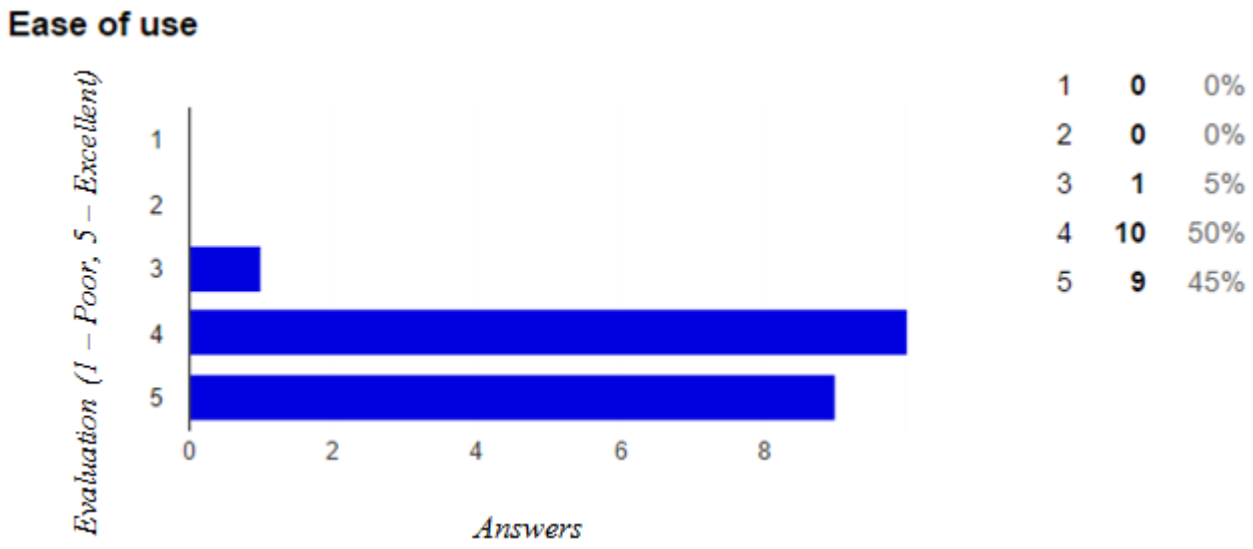



**Figure 5.4:** Ease of use

Regarding the security of the solution, the feeling that most users had was that the system is rather strong. This was particularly evaluated by performing authorized and unauthorized operations, lock, unlock and change attempts on the authentication PIN, among others. This can be seen in the different scenarios depicted in the previously presented tables. The results can be seen in Figure 5.5.

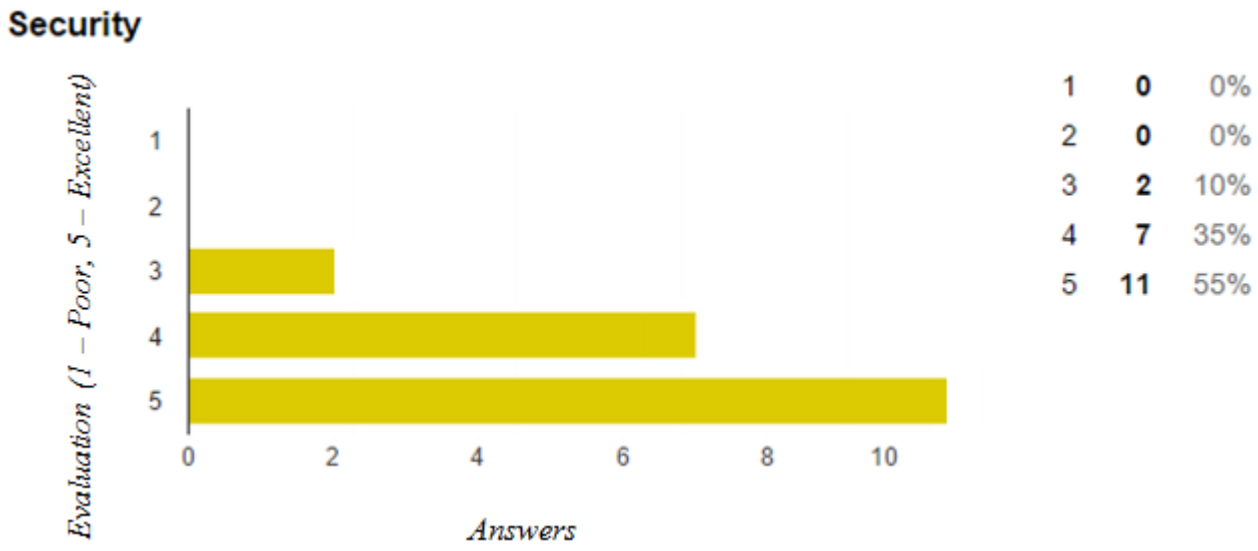



**Figure 5.5:** Security sense

As for the level of acceptability or tolerance of the users to carry the smart micro SD card, the majority of the users considered that it is an object quite easy to transport, and therefore have no problems in doing so. The results are depicted in Figure 5.6.

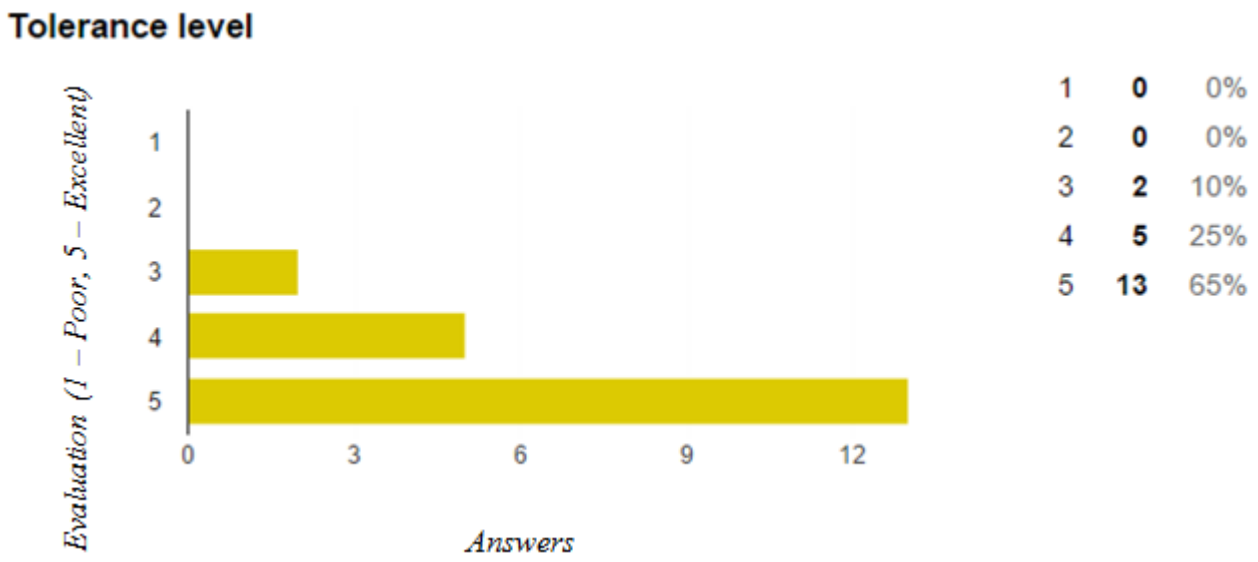



**Figure 5.6:** Smart card transport tolerance

Regarding the acceptability of the solution, all twenty participating users stated that they would use the developed solution. The result can be seen in Figure 5.7.


**Figure 5.7:** Solution acceptability

## 5.5 Compliance with Requirements

The requirements of the implemented solution are defined in Section 1.4. Throughout this document, the design and implementation details of the solution, including its properties and behavior, were presented. This section describes how these details are able to comply with the defined requirements of Section 1.4.

**Non-functional requirements**

- **The performance of the solution should not affect the user's experience**;
  From the results of the usability tests described in Section 5.4, users have an easy interaction with the solution and are satisfied with it.
- **The solution should have a low impact on the computational resources of the host device**;
  As seen in Section 5.3, the solution consumes only a small amount of system resources, therefore its impact on the device is low.
- **The solution should be able to use different mobile devices**.
  An important component of the solution is the smart micro SD card. Through its use, the solution is able to transfer the sensitive data between multiple mobile devices, and can thus be used across these devices, such as between different laptops or smartphones. Their only requisite is having a micro SD or SD slot, a hardware element very common amongst modern devices.

**Functional requirements**

- **The user must be able to digitally sign any electronic document by entering a security PIN (Personal Identification Number)**;
  By design, the digital signature operation implemented by the smart card signing applet does

not allow the signature to be executed without the provision of the correct PIN value. Therefore, the authentication PIN is demanded each time the user requests the signature of a file. This requirement can be verified by the use of the system.

- **The solution must efficiently process one document at a time**;
  By design, the certification application ensures that an operation can only be initiated if no other is in progress. Thus, the solution provides separate processing for each document signing. This requirement can be proved through the use of the system.
- **The solution must be easy to interact with, promoting the automation of the signing process**;
  Upon the user's signing request on the certification application, the data to be signed (more specifically, its hash) is seamlessly forwarded to the smart card, where the stored signature key is automatically selected and used to digitally sign the data. This causes the user's interaction with the system to be reduced to the minimum. According to the results of the usability testing displayed in Section 5.4, users find the system as easy to use.
- **The solution must be usable regardless of user location**;
  The smart card of the solution is an object that has the benefit of being easily transported. Thus, the solution is able to inherit this property, and therefore capable of being used by the user at multiple locations, without having to carry the private data over insecure devices, or other unsafe means.
- **Notification that digital signatures are legally binding: before signing any document, users must be informed that the digital signature is as legally binding as a manual signature**;
  By design, the certification application ensures the delivery of this legal notice to the user when he initiates the signature process. This requirement can be proven by using the system.
- **The solution must provide qualified digital signatures**;
  Through the use of a security token (the smart card), the solution guarantees that the digital signatures are generated only by the token, and that the private signature key used to create those signatures never leaves the token, causing it to be considered a Secure Signature Creation Device. This way, the solution ensures that the generated signatures can only be provided by the entity to which the token belongs to, turning them into qualified signatures.
- **The format of the digital signatures should be according to a standard format, so that they can be verified easily through different cryptographic suites already existing**.
  By design, the solution uses a XAdES library, which as the name implies provides the digital signatures in the standard XML Advanced Electronic Signature format.

**Security requirements**

- **The mobile signatures, to be considered equivalent to the handwritten signatures, should be generated in a Secure Signature Creation Device (the security token, like the smart micro SD card) and the user should have a qualified certificate. Thus, the SSCD shall perform on-board key generation, guaranteeing the protection of the private keys and reinforcing their secrecy since they never go outside the smart card. No one other than the card owner can use the private signature key (ensures the secrecy of the keys as well as non-repudiation)**;
  As previously stated in the functional requirements, the solution comprises a smart card as its security token. This object generates the private signature keys and the signatures inside, and guarantees that the private keys do not leave it, turning it into a SSCD. Thus, only the card owner is able to use the signature key that is stored inside the card. Also contained within the token are the public-key certificates which are later used to verify the files signed by the corresponding private keys.
- **The solution requires the authentication of the user so that it can reliably perform its functions: when digitally signing a document, the authentication of the signer must be ensured through validation of a secret code known only by the signer (PIN code or password). The PIN is a secret that shall be protected in the same way as the keying material (stored in the smart card)**;
  What was said for the first functional requirement also applies here. In addition, by design the PIN object is stored securely in the security token and does not leave it.
- **The solution must take steps to protect the smart card from online attacks, like brute force attacks. Thus, the solution must set a maximum number of authentication attempts, after which the card is blocked or the data contained therein is erased**.
  By design, the smart card signing applet supports the verification of the authentication PIN up to three times. After that, the card is locked and can only be unlocked through the unblock PIN or PUK, which is subject to the same rule. Thus, the solution provides protection against online attacks, such as brute force. This requirement can be proven by use of the system.

## 5.6 Conclusion

Through the assessment of the developed solution, it can be concluded that the objectives proposed by this thesis were achieved. The solution is able to provide a certification system which uses a mobile security card as the security element, a card with higher availability and portability when compared to other state of the art solutions, and also independent of the mobile operator.

The solution protects the sensitive information (such as the private keys used in the signature processes) from malicious entities by storing the data inside a tamper-resistant and password protected smart card. Even if the attacker gains physical control of the card, he or she cannot access the data contained therein. The possibility of a successful online brute force attack is also reduced due to the limited number of attempts an attacker can test passwords against the card. The solution also provides a secure environment where the operations that handle the private data, such as the digital signing, generation of key pairs, and others, can take place. By transporting the smart card, the solution ensures its availability without ever having to carry the sensitive material through insecure means.

The results of the performance tests demonstrate that the solution does not require many system resources, and therefore has a low impact on the computational resources of the host system. The usability tests are also a good indicator of the efficiency of the solution as the users feel that they have a good user experience. These tests also indicate that the implemented solution is well accepted.

Based on the design and implementation details, the solution is able to meet all the requirements initially established, thus providing a reliable certification system that stores the user's entire private material within a secure smart card.

# 6

# Conclusion

## Contents

## 6.1 Conclusions

In order to allow the shift to the digital world, there is a growing need, especially by companies, for digital solutions capable of providing strong and reliable signing services. This shift not only allows the signing process to be digital, but also strengthens the three security properties stipulated by handwritten signatures: authentication, non-repudiation and data integrity. There are several protocols that implement this type of signature: those that use smart cards, those based on biometrics, and the hybrids, which combine features from both. There are also several signature formats available for use, which differ primarily in the type of data they can sign and the type of signature they provide.

The strong presence of mobile technologies in the surrounding environments also has an obvious influence on today's digital solutions. There are various signature mechanisms that use mobile devices, and which are based on asymmetric cryptography and able to provide qualified digital signatures. These can be divided into three categories: those that use only the SIM card of the mobile phone, those that use both the SIM card and the device's middleware to help in the execution of the signature functions, and those using the SIM card and high-level services (which are independent of the device's technologies) to provide the signatures. Each mechanism has its own limitations, but they all share a common drawback: using the SIM card as the security element to store the private data of the user. Such card is dependent on a particular mobile operator, which forces the user to acquire a SIM card that has a usable cryptographic token, which in addition to restricting the use of the solution on devices that are solely covered by the SIM card operator, it is not something that is easy to find.

The proposed solution is an electronic certification system whose main purpose is to provide qualified digital signatures using a secure element contained in a mobile device, thus eliminating the need for external readers. The considered secure element is a mobile security card, specifically a smart card on a micro SD card, which eliminates the previously stated drawback on the state of the art, as it is independent from any mobile operator and has a higher availability in the market. It also has a stronger portability concept, since it is easier to insert/remove this type of card into/from any mobile device, and is greatly supported by many devices. The user's sensitive data is stored inside the smart card, which has tamper-resistant properties and secure computing capabilities. To access the sensitive data, the user has to authenticate himself/herself to the smart card. Online attacks, such as brute force attacks, are prevented by blocking the card once the maximum number of authentication attempts is reached. Thus, the user's data is securely stored, and even if the mobile device is attacked, the security of the digital signatures remains intact. Another advantage of the proposed solution is its interoperability, providing its interface via the PKCS#11 standard. Any application that complies with this commonly used standard can use the developed system without having to know its details when communicating with it.

The implementation of the solution includes various technologies: XAdES standard as the format for reliable digital signatures, which was accomplished with the Java library JDigiDoc; Java Card for the

programming of the smart card's behavior; PKCS#11 standard and also the card's library GTSDUpi to develop the middleware; and Swing to build the client application's interface. In order to assess the solution, some performance tests were performed on the client application, along with several usability tests on both applications. The results from the performance tests indicate that the solution has a low impact on the host device's computational resources, namely in terms of CPU consumption, required memory and the amount of input/output data. As for the usability tests, they demonstrate that: the solution is particularly easy to use, it offers a strong level of security, it is rather flexible by just requiring a smart micro SD card without needing any external card reader, and it is very well accepted. Thus, both types of tests provided good indicators as to the viability and reliability of the solution.

The work of this thesis was performed in a business context, as part of a public administration organization's project related with the Electronic Certification of the Portuguese State, and also had the collaboration of the PDM&FC company.

## 6.2 Future Work

The work presented throughout this document was properly implemented. However, there is still work that can be developed in order to improve the system, particularly:

**Mobile application**

The implemented solution is designed for the Windows platform. The development of a solution capable of running in the Android platform may be a very useful feature. The smart card library – GTSDUpi –, besides having an API for Windows which was used in the implemented solution, also has an API for Android although with some differences. By performing some modifications in the middleware of the solution, specifically in the PKCS#11 module, it is possible to create a new module compatible with the Android system. The new developed Android application may then communicate with the signing applet in the exact same way as the implemented solution, using the module and the PKCS#11 interface, as the applet is independent from the rest of the components.

This mobile application came to be developed and completed after this dissertation have been delivered, this being the reason why it has not been discussed herealong. The features of this application are exactly the same as those of the developed Client application, and they have all been tested (at the functional level) on devices with Android version 4.2 (Jelly Bean) and 4.4 (Kit-Kat).

**Remote component**

The developed solution provides a local interaction, meaning it accomplishes all the certification processes with the secure element (smart card) physically connected to the mobile device, without any kind of external readers. Another interesting option is to allow the remote use of the solution, providing the ability to perform the certification mechanisms, such as digital signing, outside the mobile device that contains the smart card. A possible scenario would be to have a smartphone with the card stored within, and the user could digitally sign a file in his/her laptop by having the laptop communicating remotely with the secure card and using the signature keys stored therein. The communication channel between the two components can be performed using Bluetooth. Also, since some exchanged messages would transport sensitive data, the communication would need to be secure, possibly using temporary session keys, thus guaranteeing the confidentiality and authenticity of the messages.

# Bibliography


[1] Microsoft, "Digital Signatures," https://technet.microsoft.com/en-us/library/cc962021.aspx, online; accessed 20-November-2014.

[2] Adobe, "The AdES family of standards: CAdES, XAdES, and PAdES - Implementation Guidance for Using Electronic Signatures in the European Union [White Paper]," 2009, retrieved from https://blogs.adobe.com/security/91014620_eusig_wp_ue.pdf.

[3] A. Ruiz-Martínez, D. Sánchez-Martínez, M. Martínez-Montesinos, and A. F. Gómez-Skarmeta, "A Survey of Electronic Signature Solutions in Mobile Devices," *J. Theor. Appl. Electron. Commer. Res.*, vol. 2, no. 3, pp. 94–109, Dec. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1330352.1330360

[4] E. Pisko, "Mobile electronic signatures: progression from mobile service to mobile application unit," in *Management of Mobile Business, 2007. ICMB 2007. International Conference on the*. IEEE, 2007, pp. 6–6.

[5] M. H. Samadani, M. Shajari, and M. M. Ahaniha, "Self-proxy mobile signature: A new client-based mobile signature model," in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*. IEEE, 2010, pp. 437–442.

[6] A. Ruiz-Martínez, J. Sánchez-Montesinos, and D. Sánchez-Martínez, "A mobile network operator-independent mobile signature service," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 294–311, 2011.

[7] Net Applications, "Mobile/Tablet Operating System Market Share," http://www.netmarketshare.com/, online; accessed 20-November-2014.

[8] IAIK TU Graz, "Pkcs#11 wrapper," https://jce.iaik.tugraz.at/sic/Products/Core_Crypto_Toolkits/PKCS_11_Wrapper, online; accessed 11-March-2015.

[9] Tomi T. Ahonen, "TomiAhonen Consulting Analysis 7 November 2014: Smartphone Bloodbath Q3 full results Top 10 brands, OS platforms

and installed base," http://communities-dominate.blogs.com/brands/2014/11/smartphone-bloodbath-q3-full-results-top-10-brands-os-platforms-and-installed-base.html, online; accessed 21-November-2014.

[10] The European Parliament and the Concil of the European Union, "Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures," *Official Journal of the European Communities*, vol. L 12, pp. 12–20, 2000. [Online]. Available: http://europa.eu.int/eur-lex/pri/en/oj/dat/2000/l_013/l_01320000119en00120020.pdf

[11] *ARM Security Technology - Building a Secure System using TrustZone Technology*, ARM Limited, 2009.

[12] N. Santos, H. Raj, S. Saroiu, and A. Wolman, "Using ARM Trustzone to Build a Trusted Language Runtime for Mobile Applications," *SIGARCH Comput. Archit. News*, vol. 42, no. 1, pp. 67–80, Feb. 2014. [Online]. Available: http://doi.acm.org/10.1145/2654822.2541949

[13] A. G. Fitzek, "Development of an ARM TrustZone aware operating system ANDIX OS," 2014.

[14] Microsoft, "The STRIDE Threat Model," https://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx, online; accessed 21-November-2014.

[15] S. Antipolis, "New ETSI standard for EU-compliant electronic signatures," http://www.etsi.org/index.php/news-events/news/324-news-release-14th-september-2009, 2009, online; accessed 18-November-2014.

[16] F. Nentwich, E. Kirda, and C. Kruegel, "Practical security aspects of digital signature systems," *Technical University Vienna. Technical*, 2006.

[17] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.

[18] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342

[19] D. M'Raihi and M. Yung, "E-commerce applications of smart cards," *Computer Networks*, vol. 36, no. 4, pp. 453 – 472, 2001, current Directions in Smart Cards. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128601001669

[20] H. Handschuh and P. Paillier, "Smart Card Crypto-Coprocessors for Public-Key Cryptography," in *CARDIS*, ser. Lecture Notes in Computer Science, J.-J. Quisquater and B. Schneier, Eds.,

vol. 1820. Springer, 1998, pp. 372–379. [Online]. Available: http://dblp.uni-trier.de/db/conf/cardis/cardis1998.html#HandschuhP98a

[21] S. C. A. P. A. Council, "Smart Card Technology and the National Strategy for Trusted Identities in Cyberspace (NSTIC) [White Paper]," June 2013, retrieved from http://www.smartcardalliance.org/resources/pdf/SC_Technology_and_NSTIC_Brief_FINAL_060813.pdf.

[22] R. Housley, "Cryptographic Message Syntax (CMS)," RFC 5652 (Draft Standard), Internet Engineering Task Force, September 2009. [Online]. Available: http://www.ietf.org/rfc/rfc5652.txt

[23] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), Internet Engineering Task Force, May 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5280.txt

[24] D. Pinkas, N. Pope, and J. Ross, "CMS Advanced Electronic Signatures (CAdES)," RFC 5126 (Informational), Internet Engineering Task Force, March 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5126.txt

[25] EldoS, "CAdES and Digital Signatures," https://www.eldos.com/security/articles/7031.php?page=all, online; accessed 18-November-2014.

[26] H. Brzica, B. Herceg, and H. Stančić, "Long-term Preservation of Validity of Electronically Signed Records," *electronic form only:: NE*, 2013.

[27] N. Guedes, "Implementação de Solução de Assinaturas Digitais," Ph.D. dissertation, 2008.

[28] N. Hastings, W. Polk, and D. Cooper, "Common Format for Information that is Digitally Signed: A Final Report," 2001.

[29] S. Shirasuna, A. Slominski, L. Fang, and D. Gannon, "Performance Comparison of Security Mechanisms for Grid Services," in *5th IEEE/ACM International Workshop on Grid Computing*, November 2004. [Online]. Available: http://www.extreme.indiana.edu/xgws/papers/sec-perf-short.pdf

[30] H. Liu, S. Pallikara, and G. Fox, "Performance of web services security," in *Proceedings of 13th Annual Mardi Gras Conference*, 2005, pp. 1–8.

[31] ISO, *ISO 32000-1:2008. Document management — Portable document format — Part 1: PDF 1.7*, 2008. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51502

[32] Adobe, "Adobe Reader," https://www.adobe.com/, online; accessed 19-November-2014.

[33] Gemalto, "Gemplus, France Telecom Mobiles and Certplus to pilot Wireless PKI based secure transactions Mobitrust enables secure transactions over a mobile phone using a digital signature based on Public Key Infrastructure," http://www.gemalto.com/press-site/gemplus/2001/telecom/13-02-2001-gemplus-france-tele.htm, 2001, online; accessed 19-November-2014.

[34] ——, "Baltimore Technologies, AU-System and Gemplus deliver world's first digital signature over next generation wireless networks Partnership provides end-to-end security solutions for new packet based GPRS mobile networks ," http://www.gemalto.com/press-site/gemplus/2001/telecom/22-03-2001-baltimore-technologi.htm, 2001, online; accessed 19-November-2014.

[35] ——, "Gemplus supplies SIM/WIM for first financial transaction on WAP with digital signature," http://www.gemalto.com/press-site/gemplus/2001/telecom/23-05-2001-gemplus-supplies-sim.htm, 2001, online; accessed 19-November-2014.

[36] 3GPP, "The 3rd Generation Partnership Project," http://www.3gpp.org/, online; accessed 21-November-2014.

[37] T. M. Jurgensen and S. Guthery, *The Smart Cards: A Developer's Toolkit*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[38] Logos Smart Card, "S@T," http://www.logossmartcard.com/smartcard-products-sat.php, online; accessed 19-November-2014.

[39] SmartTrust, "SmartTrust Wib," http://www.gi-de.com/aus/en/products_and_solutions/products/sim_lifecycle_management_1/wib/wib.jsp, online; accessed 19-November-2014.

[40] Oracle, "Java Micro Edition Software Development Kit Developer's Guide," https://docs.oracle.com/javame/dev-tools/jme-sdk-3.4/ecl/html/satsa.htm, online; accessed 19-November-2014.

[41] *ISO/IEC 7816-4:2013: Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange*, ISO/IEC Std., [Online]. Available: http://www.iso.org/.

[42] *European Telecommunications Standards Institute Specifications*, ETSI Std., [Online]. Available: http://www.etsi.org/.

[43] E. Saharanta, "Outlining the Key Drivers for Mobile Signature Services and Short Overview of the ETSI - MSS Concept and How the SIM Card and Mobile Network Operator's Infrastructure Plays a Key Role," in *Proceedings OASIS Adoption Forum. London, UK*, 2006.

[44] Methics Ltd, "Kiuru MSSP," http://www.methics.fi/products/kiuru-mssp/, online; accessed 20-November-2014.

[45] Valimo Wireless Ltd, "Valimo mobile ID [Brochure]," 2014, retrieved from http://www.valimo.com/.

[46] Giesecke & Devrient, "SmartLicentio [Brochure]," 2011, retrieved from http://www.gi-de.com/.

[47] The Globus Alliance, "Gt 4.0 security: Key concepts," http://toolkit.globus.org/toolkit/docs/4.0/security/key-index.html, online; accessed 20-November-2014.

[48] E. C. for Standardization (CEN), "Security requirements for signature creation applications," *CEN workshop agreement (CWA) 14170*, 2004.

[49] A. Ruiz-Martínez, C. Inmaculada Marín-López, D. Sánchez-Martínez, and I. Castell Egea, "Sipmsign: a lightweight mobile signature service based on the session initiation protocol," *Software: Practice and Experience*, 2012.

[50] Cisco Systems Inc., "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018 [White Paper]," Feb. 2014, retrieved from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[51] Accenture, "Nokia and Accenture Close Symbian Software Development and Support Services Outsourcing Agreement [Press Release]," http://newsroom.accenture.com/news/nokia-and-accenture-close-symbian-software-development-and-support-services-outsourcing-agreement.htm, September 2011.

[52] M Yasir Khan, "Project Report on J2ME implementation in Mobile App Development for Java enabled handsets," Master's thesis, Faculty of Engineering & Technology - Jamia Millia Islamia University, May 2012. [Online]. Available: http://www.academia.edu/4212543/Project_Report_on_J2ME_implementation_in_Mobile_App_Development_for_Java_enabled_handsets

[53] Oracle, "Java Platform, Micro Edition (Java ME)," http://www.oracle.com/technetwork/java/embedded/javame/index.html, online; accessed 21-November-2014.

[54] O. Iliescu, *Pro Java ME Apps: Building Commercial Quality Java ME Apps*, ser. Apresspod Series. Apress, 2011. [Online]. Available: http://books.google.pt/books?id=UtGuoJPWBAgC

[55] S. Analytics, http://strategyanalytics.com, online; accessed 21-November-2014.

[56] Google, https://android.com/, online; accessed 21-November-2014.

[57] Strategy Analytics, "Android Captured Record 85 Percent Share of Global Smartphone Shipments in Q2 2014," http://blogs.strategyanalytics.com/WSS/post/2014/07/30/Android-Captured-Record-85-Percent-Share-of-Global-Smartphone-Shipments-in-Q2-2014.aspx, online; accessed 21-November-2014.

[58] Google, "Android security overview," https://source.android.com/devices/tech/security/, online; accessed 21-November-2014.

[59] ——, “Security tips,” https://developer.android.com/training/articles/security-tips.html, online; accessed 21-November-2014.

[60] ——, “Publishing overview,” https://developer.android.com/tools/publishing/publishing_overview.html, online; accessed 21-November-2014.

[61] Apple Inc, https://www.apple.com/ios/, online; accessed 21-November-2014.

[62] ——, “Apple Announces iOS 8 Available September 17 [Press Info],” https://www.apple.com/pr/library/2014/09/09Apple-Announces-iOS-8-Available-September-17.html, September 2014, online; accessed 21-November-2014.

[63] Microsoft, http://www.windowsphone.com/, online; accessed 21-November-2014.

[64] Statista Inc, http://www.statista.com/, online; accessed 21-November-2014.

[65] ——, “Number of apps available in leading app stores as of July 2014,” http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, online; accessed 21-November-2014.

[66] W. Rankl and K. Cox, *Smart Card Applications: Design models for using and programming smart cards*. Wiley, 2007. [Online]. Available: http://books.google.pt/books?id=sUMpC4Rw4AMC

[67] *ISO/IEC 7816: Identification cards – Integrated circuit cards*, ISO/IEC Std., [Online]. Available: http://www.iso.org/.

[68] *ISO/IEC 14443: Identification cards – Contactless integrated circuit cards – Proximity cards*, ISO/IEC Std., [Online]. Available: http://www.iso.org/.

[69] K. Mayes and K. Markantonakis, *Smart Cards, Tokens, Security and Applications*, 1st ed.

[70] Nuno Miguel Grilo Fonseca Pinheiro, “Secure password management using smart cards,” Master’s thesis, Instituto Superior Técnico, Universidade de Lisboa, Nov. 2013.

[71] Oracle, “Java Card Technology,” http://www.oracle.com/technetwork/java/embedded/javacard/overview/index-jsp-140503.html, online; accessed 21-November-2014.

[72] GlobalPlatform, “Card Specifications,” http://www.globalplatform.org/specificationscard.asp, online; accessed 21-November-2014.

[73] *ISO/IEC 7810:2003 Identification cards – Physical characteristics*, ISO/IEC Std., [Online]. Available: http://www.iso.org/.

[74] ETSI, "ETSI TS 102 221 V11.0.0: Smart Cards; UICC-Terminal interface; Physical and logical characteristics," http://www.etsi.org/deliver/etsi_ts/102200_102299/102221/11.00.00_60/ts_102221v110000p.pdf, June 2012.

[75] EMC, "PKCS 11: Cryptographic Token Interface Standard," http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-11-cryptographic-token-interface-standard.htm, online; accessed 23-November-2014.

[76] GO-Trust, http://www.go-trust.com/, online; accessed 26-May-2014.

[77] Oracle, "Java Native Interface," https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/intro.html, online; accessed 19-March-2014.

[78] "Opensc," https://github.com/OpenSC/OpenSC/wiki, online; accessed 11-March-2015.

[79] Cryptware di Ugo Chirico, "Ncryptoki," http://www.ncryptoki.com/, online; accessed 13-March-2015.

[80] IAIK TU Graz, "Iaik-xades," https://jce.iaik.tugraz.at/sic/Products/XML_Security/XAdES, online; accessed 7-March-2015.

[81] Luis Filipe dos Santos Goncalves, "Xades4j - a java library for xades signature services," Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, 2010.

[82] AS Sertifitseerimiskeskus, "Jdigidoc," http://www.id.ee/?id=35783, online; accessed 8-March-2015.

[83] ——, "JDigiDoc Programmer's Guide," http://www.id.ee/public/SK-JDD-PRG-GUIDE.pdf, Internet Engineering Task Force, June 2013.

[84] ETSI, "ETSI TS 102 918 V1.2.1: Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC)," http://www.etsi.org/deliver/etsi_ts/102900_102999/102918/01.02.01_60/ts_102918v010201p.pdf, February 2012.

[85] Oracle, "Swing," https://docs.oracle.com/javase/tutorial/uiswing/index.html, online; accessed 13-August-2015.

[86] "Process Explorer," https://technet.microsoft.com/en-us/sysinternals/bb896653, online; accessed 15-September-2015.

[87] J.-G. Jo, J.-W. Seo, and H.-W. Lee, "Biometric Digital Signature Key Generation and Cryptography Communication Based on Fingerprint," in *Proceedings of the 1st Annual International Conference on Frontiers in Algorithmics*, ser. FAW'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 38–49. [Online]. Available: http://dl.acm.org/citation.cfm?id=1776166.1776170

[88] P. Janbandhu, *Novel Biometric Digital Signature System for Electronic Commerce Applications*. Nanyang Technological University, School of Electrical and Electronic Engineering, 2002. [Online]. Available: http://books.google.pt/books?id=7vjrNwAACAAJ

[89] T. Kwon and J. Lee, "Practical Digital Signature Generation Using Biometrics." in *ICCSA (1)*, ser. Lecture Notes in Computer Science, A. Laganà, M. L. Gavrilova, V. Kumar, Y. Mun, C. J. K. Tan, and O. Gervasi, Eds., vol. 3043. Springer, 2004, pp. 728–737. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccsa/iccsa2004-1.html#KwonL04

[90] P. K. Janbandhu and M. Y. Siyal, "Modified Private Key Generation for Biometric Signatures," in *Advances in Automation, Multimedia and Video Systems, and Modern Computer Science*. WSEAS, 2001.

[91] S. C. A. P. A. Council, "Smart Cards and Biometrics [White Paper]," 2011, retrieved from http://www.smartcardalliance.org/resources/pdf/Smart_Cards_and_Biometrics_030111.pdf.

# A

# Digital signature protocols

**Biometric-based protocol**

Biometrics is defined as the science of measuring and analyzing an individual's biological data. With the use of digital technologies, it can identify a person based on distinctive measurable physiological or behavioral characteristics [87] such as face, iris pattern, fingerprint, retina, and others. A few digital signature systems based on this type of characteristics, known as biometric-based, exist and consist on generating a digital signature by deriving the signature key from a viable human source.

One of the fundamental aspects that these systems must ensure is private key renewal so it can maintain its resilience if, somehow, the private key is ever disclosed or discovered, or the biometric template is compromised. Since the generation of the signature key is always based on a specific biometric feature, one might think that the key would always be the same. However, there are several proposed systems [87] [88] [89] that are able to generate as many private keys as required. They achieve this by modifying the current key generation process, including additional elements, such as personal secrets or random variables.

A serious downside of the biometric-based systems is that some implementations require users to

store secret data in one location, which always presents a risk. For instance, in [87] users have to initially register their biometrics on a Biometric Certification Authority (BCA), causing the storage of all biometric templates in a centralized location, becoming a very attractive target to attackers.

Another limitation is in capturing the biometric samples from the user, since the capturing devices still have some issues. For example, using the facial technology, which does not require high accuracy, the user's recognition is affected by changes in lighting, his hair, the use of accessories (e.g., glasses), etc. Similarly, using a very high accuracy technology which require very accurate samples, such as iris recognition, is also challenging because it is very difficult to obtain an iris sample without errors. This is due to the difficulties of obtaining a good image of the iris at a distance of about a meter, and sometimes through contact lenses. [90].

The major disadvantage relates to the fact that currently, despite the significant technological developments made in the area of biometrics in recent years, there are still very few computational devices that offer this kind of support, at least for the general public. Take mobile devices, like smartphones and tablets, for instance: they are built with more and more powerful computing capabilities and yet, the vast majority of them still lack biometric capabilities.

**Hybrid protocol**

There are a number of feasible solutions that comprise both the smart card and biometric technologies for performing the digital signature process. These solutions take advantage of the strengths of each technology and merge them into a single collaboration system.

As previously stated, smart cards are widely acknowledged as one of the most secure and reliable means for secure identification and digital signatures. Biometric technology in turn, is considered a strong method of identity verification [91]. Their combination "provides the means to create a positive binding of the smart card to the cardholder thereby enabling a strong verification and authentication of the cardholder's identity" as stated in [91]. Therefore, this hybrid solution works by using biometrics as an authentication measure to access the signature key (or some secret value) stored in the smart card through which one can generate the digital signature. The signature can only be produced if and only if the authentication made by the reader device has been successful. For example, if the biometric element of the system were based on fingerprints, the cryptographic key stored in the smart card would only be accessible when the signer's captured fingerprint samples matched his stored template.

A great advantage of this type of solution is that, as mentioned in the example, the biometric templates can be stored in the smart card. Thus, during the verification of the cardholder's identity the comparison can be made locally, without having to connect to a remote database of biometric identifiers. Also, since all biometric matching is done via templates, it is not necessary to store the

complete biometric image data on the smart card [91]. This eliminates a major problem presented by the biometric-based solution. Each template represents a user's personal characteristics, so obviously its storage presents privacy concerns. By storing the template on the smart card the system enhances individual privacy and increases protection against attacks, since each user controls its own template.

Many commercial solutions have already adopted this kind of system. For instance, several banks explore the use of biometrics and smart cards to better authenticate customers on online banking, trading and purchasing transactions. There are some implementations that improve the security even further by placing the biometric verification process directly on the smart card rather than having it on the reader device (called match-on-card approach). For instance, vendors such as Biometric Associates have built a fingerprint sensor directly into the reader, which in turn sends the biometric to the smart card to perform the verification. Another example is the Portuguese Citizen Card, whose chip has the details of fingerprints of the cardholder and an application to compare these with those collected by an external digital fingerprint reader. With this reader it is possible to make the comparison on-chip without the citizen's information being sent to the outside of the card.

# B

# Graphical User Interface

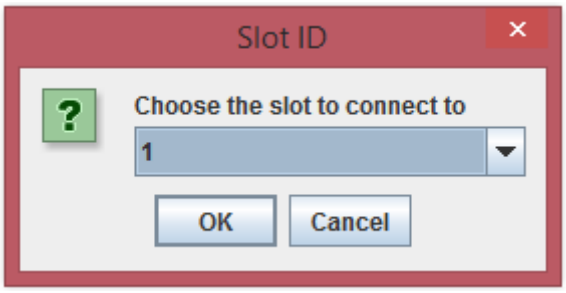



**Figure B.1:** Connection screen

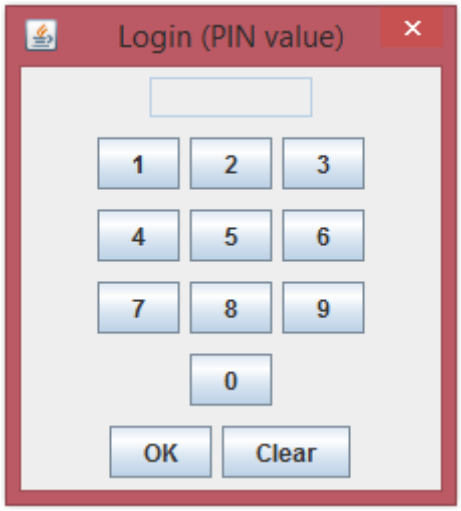



**Figure B.2:** Login screen

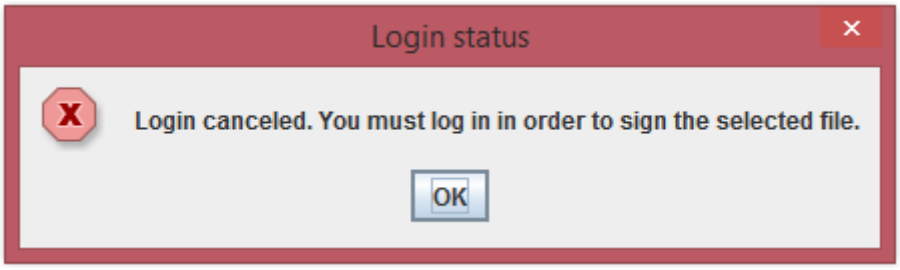



**Figure B.3:** Canceled login information screen

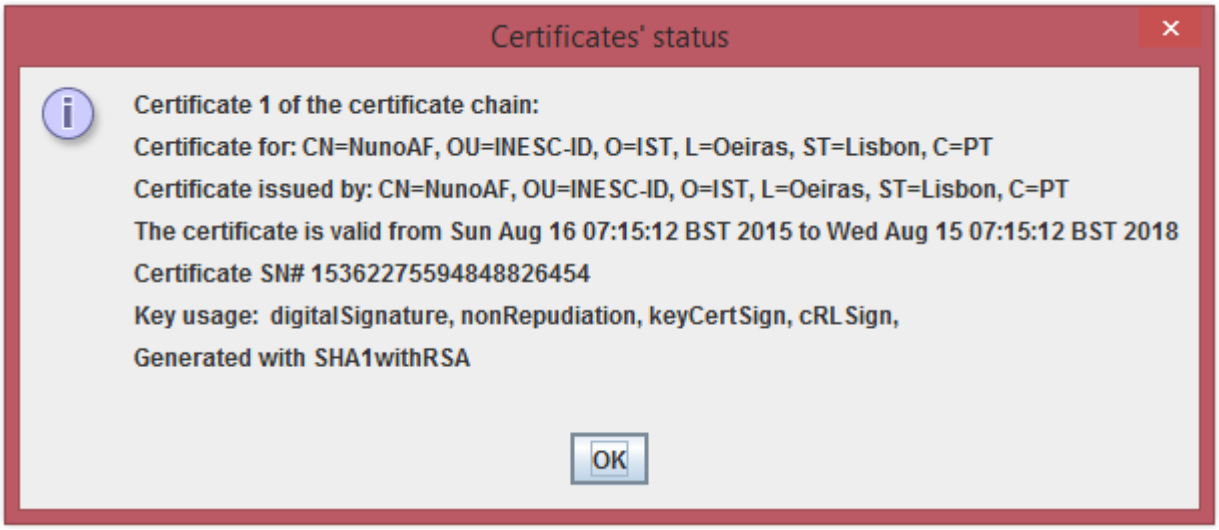



**Figure B.4:** Certificates' information screen

# C

# Usability survey

# Reliable electronic certification on mobile devices

The purpose of this survey is to conduct an evaluation of the solution created under the master thesis "Reliable electronic certification on mobile devices" in terms of its usability. In addition, the performed questions allow to assess the acceptance of the solution.

1. **Gender**
   *Mark only one oval.*

   - Male
   - Female

2. **System usability**
   Rate the system on a scale of 1 to 5 regarding the listed attributes (1 - Poor, 5 - Excellent).
   *Mark only one oval per row.*

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Ease of use | | | | | |
| Speed | | | | | |
| Utility | | | | | |
| Security | | | | | |

3. **System quality**
   Rate the listed properties on a scale of 1 to 5 according to their importance on the quality of the system (1 - Not important, 5 - Very important).
   *Mark only one oval per row.*

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Performing the sensitive operations (e.g., digital signing) inside the smart card | | | | | |
| Not requiring an external card reader | | | | | |
| Storing the private data (e.g., signature keys) inside the smart card | | | | | |
| Possibility of integrating the smartcard with smartphones and tablets | | | | | |
| Requiring a PIN authentication to perform sensitive operations | | | | | |

4. **How often do you use the electronic functionalities of the Portuguese Citizen Card?**
*Mark only one oval.*

- I never used them
- Rarely
- Sometimes
- Frequently
- Very frequently

5. **Have you ever used the digital signing functionality of the Portuguese Citizen Card?**
*Mark only one oval.*

- Yes
- No

6. **If you answered 'Yes' to the previous question, consider the following one. If not, then proceed to the next question.**
To perform a digital signature of a file, which of the two solutions would you say is easier to use?
*Mark only one oval.*

- Portuguese Citizen Card (integrated with applications such as Adobe Reader or Microsoft Word)
- Tested solution

7. **Smart card transportation**
In order to use the system, the user needs to carry a smart micro-SD card. On a scale of 1 to 5, rate your tolerance for the transport of such object (1 - Do not tolerate it, 5 - Do not have any problems in transporting it).
*Mark only one oval per row.*

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Tolerance level | | | | | |

8. **Would you use this system?**
*Mark only one oval.*

- Yes
- No

9. **Suggestions / Ideas / Comments**

.....

.....

.....

.....

.....

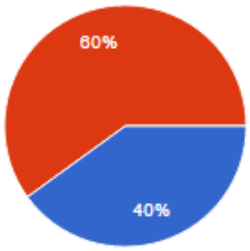
# D

# Usability survey responses

# 20 responses

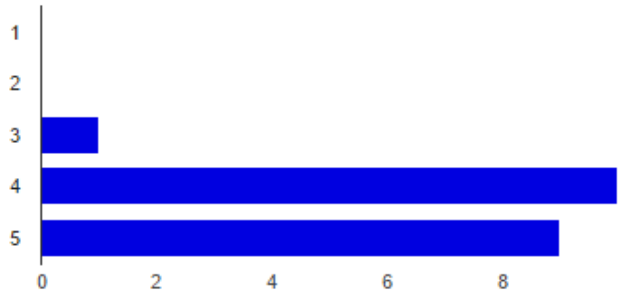
## Summary

### Gender



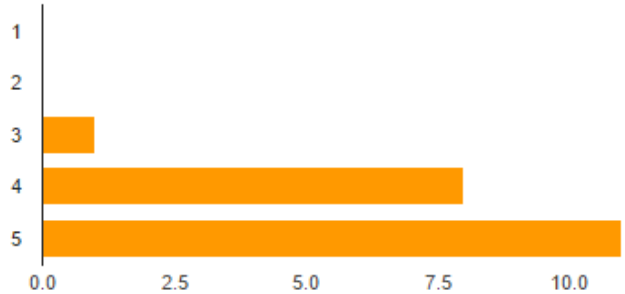

| | | |
|---|---|---|
| Male | 8 | 40% |
| Female | 12 | 60% |

### Ease of use [System usability]



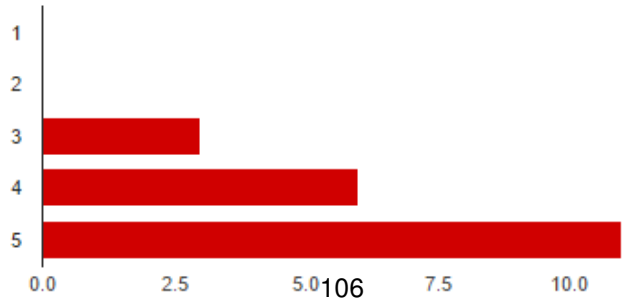

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 1 | 5% |
| 4 | 10 | 50% |
| 5 | 9 | 45% |

### Speed [System usability]



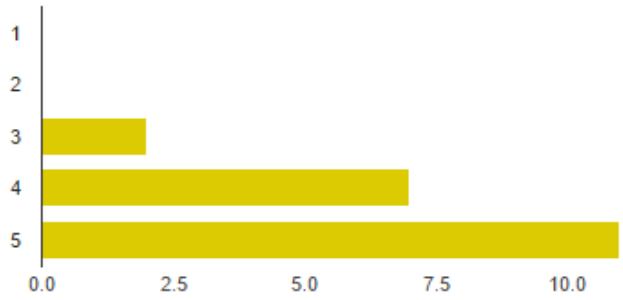

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 1 | 5% |
| 4 | 8 | 40% |
| 5 | 11 | 55% |

### Utility [System usability]



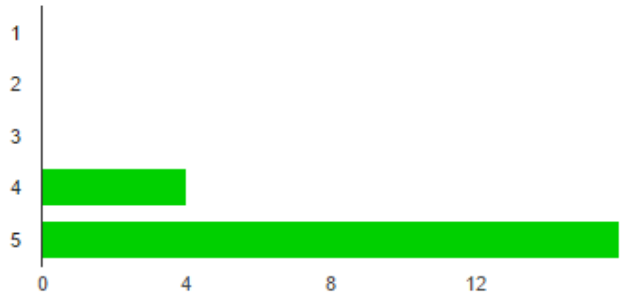

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 3 | 15% |
| 4 | 6 | 30% |
| 5 | 11 | 55% |

**Security [System usability]**



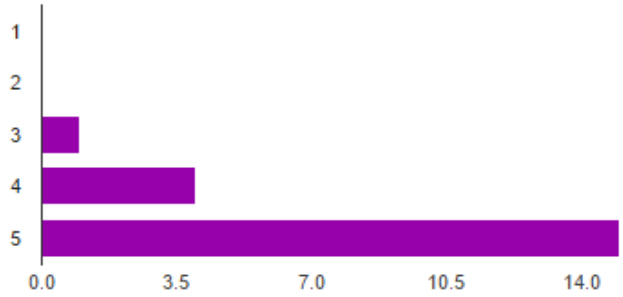

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 2 | 10% |
| 4 | 7 | 35% |
| 5 | 11 | 55% |

**Performing the sensitive operations (e.g., digital signing) inside the smart card [System quality]**



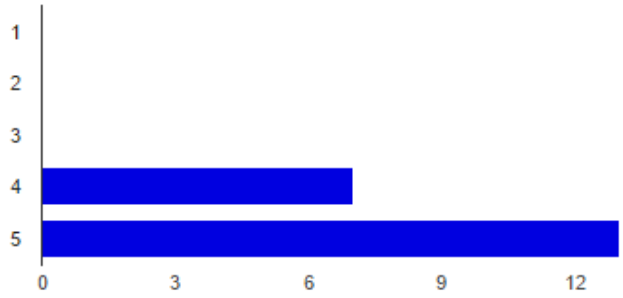

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 0 | 0% |
| 4 | 4 | 20% |
| 5 | 16 | 80% |

**Not requiring an external card reader [System quality]**



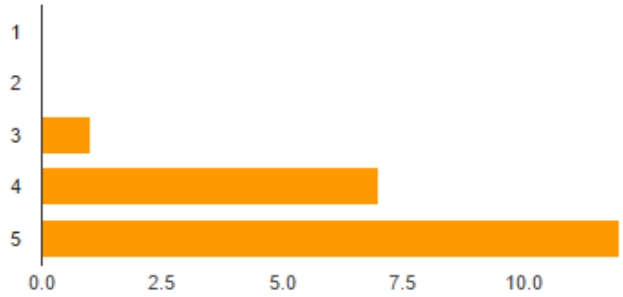

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 1 | 5% |
| 4 | 4 | 20% |
| 5 | 15 | 75% |

**Storing the private data (e.g., signature keys) inside the smart card [System quality]**



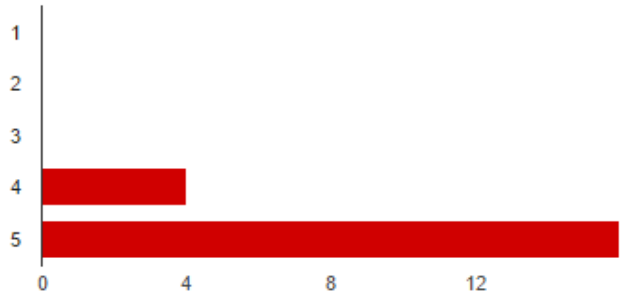

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 0 | 0% |
| 4 | 7 | 35% |
| 5 | 13 | 65% |

**Possibility of integrating the smartcard with smartphones and tablets [System quality]**



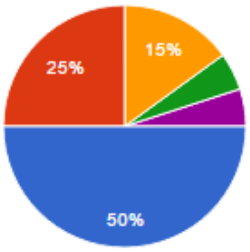

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 1 | 5% |
| 4 | 7 | 35% |
| 5 | 12 | 60% |

**Requiring a PIN authentication to perform sensitive operations [System quality]**



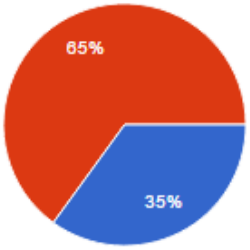

| | | |
|---|---|---|
| 1 | 0 | 0% |
| 2 | 0 | 0% |
| 3 | 0 | 0% |
| 4 | 4 | 20% |
| 5 | 16 | 80% |

**How often do you use the electronic functionalities of the Portuguese Citizen Card?**




| | | |
|---|---|---|
| I never used them | 10 | 50% |
| Rarely | 5 | 25% |
| Sometimes | 3 | 15% |
| Frequently | 1 | 5% |
| Very frequently | 1 | 5% |

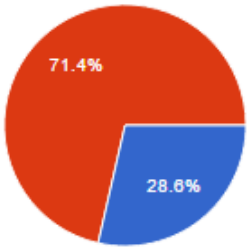
**Have you ever used the digital signing functionality of the Portuguese Citizen Card?**




| | | |
|---|---|---|
| Yes | 7 | 35% |
| No | 13 | 65% |

**If you answered 'Yes' to the previous question, consider the following one. If not, then proceed to the next question.**

| | | |
|---|---|---|
| Portuguese Citizen Card (integrated with applications such as Adobe Reader or Microsoft Word) | **2** | 28.6% |
| Tested solution | **5** | 71.4% |




**Tolerance level [Smart card transportation]**


| | | |
|---|---|---|
| 1 | **0** | 0% |
| 2 | **0** | 0% |
| 3 | **2** | 10% |
| 4 | **5** | 25% |
| 5 | **13** | 65% |

**Would you use this system?**


| | | |
|---|---|---|
| Yes | **20** | 100% |
| No | **0** | 0% |