Digital Signature Platform on Mobile Devices

José Manuel Fornés Rumbao
Department of Telematic Engineering
Seville University
Seville, Spain
fornes@trajano.us.es

Francisco Rodriguez Rubio
Department of Systems and Automatic Engineering
Seville University
Seville, Spain
rubio@esi.us.es

***Abstract*— Since ancient times, the obsession with security and authenticity has been an issue that has produced the development of diverse technologies, to avoid the access of third persons to information or to private places, and to guarantee the identity of a person with regard to a certain fact. Undoubtedly, in a society like today, which comes naming like "of the information", these issues are a basic aspect given the large number of everyday situations that occur relating to the use of confidential information and purposes of authenticity. For it, it becomes necessary to investigate, legislate, and to develop applications and systems that help to preserve the security and authenticity of a user and so, to provide them with sufficient capacity in order that these aspects in telematic networks are exported to the world of the mobile devices. This article describes the mechanisms that can be used as digital signatures and certification to obtain the electronic security in mobile devices. In addition, we propose a real platform already realized for the implementation of the digital signature in mobile terminals.**

Keywords - digital signature; certificate; midlets; servlet; cryptography.

I. INTRODUCTION

In today's society mobile devices have become widely accepted. They have achieved a great popularity, thanks to its easy operation and low cost being widely used by people all around the world, increasing their power every day. This processing power, which grows more and more, will do possible the operations of calculating of summary data algorithms (hash functions) [1] and the digital signature, also; it will be possible to do on a mobile phone in a little period of time. And all this facts combined with the great improvement in the capacity of mobile networks makes it very interesting to research and it develops technologies that are suited to the terminals for conducting electronic signatures [2].

The mobile world should adapt itself to new trends in electronic signatures and digital certificates that it is concerned, both for electronic commerce and to the carry out administrative´s procedures online, as well as other possibilities offered by technology in this field [3]. Therefore, we find that, on the one hand, mobile phones are functional devices with a great potential. On the other hand, there is a way to get that the citizens interact with the government and other existing services in this field, through electronic means.

This article seeks two objectives, to report about the current existing technologies and develop an application that allows electronic signature capabilities through mobile phone with a digital certificate installed.

The paper is organized as follows. In Section II, we review the theory of cryptography, its objectives, its implementation on mobile devices and the different alternatives that we have. In Section III, we present the development of our platform divided in client and server. Section IV shows the results of our platform and in Section V we propose the possible improvements. Finally, Section VI concludes the paper.

II. STATE OF THE ART

A. *Electronic Security*

The basic pillars of security [4] in communications and secure exchange of electronic documents are:

- Privacy: Preventing that a third party may intercept (read) the information submitted in the case of sensitive data.
- Integrity: Preventing that another agent outside the issuer, get modify (insert, delete, mess, etc.) the information sent.
- Authenticity: Preventing that a third party can impersonate the other party.
- Non-repudiation: Preventing the other party could deny the participation in communication (either as a source or destination).

These are conditions that must be met in order to establish a secure environment in the digital world and we will see that this security can be ensured by the use of cryptography, digital signatures and certificates.

Current cryptography is mainly divided into two very distinct branches which are detailed below.

1) Symmetric Cryptography

Symmetric cryptography [5] refers to the set of methods that allow secure communication between the parties, because the key has been exchanged previously, which is called symmetric key. Symmetry means that parties have the same key to encrypt and decrypt. This type of cryptography is also known as private key cryptography.

Although there is no standard type of design, perhaps the most popular is the DES (Data Encryption Standard), which is essentially a cryptographic system that takes as input a block of 64 bits of the message and is submitted to 16 interactions, with a key of 56-bit.

With a brute force method, it could break DES by making it unsafe for high security purposes. The option to replace DES has been a new encryption system which is now known as triple-DES or TDES which consists of applying DES three times.

A large number of symmetric cryptographic systems was designed in the past 20 years, including some of them which are: RC-5 [6], IDEA [7], FEAL [8], LOKI'91 [9], DESX [10], Blowfish [11], CAST [12], GOST [13], etc. However, they have not had the scope of DES, although some of them have better properties.

2) Asymmtric Cryptography

Asymmetric encryption algorithms [14] use a different key pair in communication, one to encrypt and another to decrypt. Both keys are mathematically related and it is virtually impossible to derive one from another. The key pair is generated based on asymmetric encryption algorithm used, being a secret (private) and the another is known for others (public key).

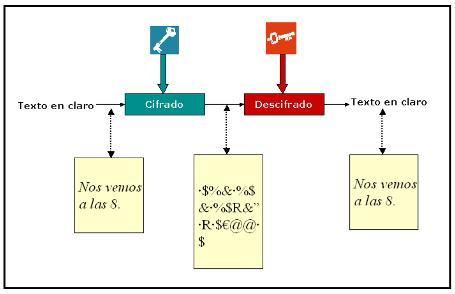


Figure 1. Process communication in an asymmetric key system [15].

The clear message is encrypted with the private / public key and only his partner (private / public) can decrypt it. The safety of this system is based on the impossibility of calculating a key from another, besides, of course, to keep the private key secret.

A public key cryptosystem to note is the RSA algorithm [16]. It is based on the difficulty of factoring large numbers. The messages sent using the RSA algorithm are represented by numbers and the operation is based on the product of two large prime numbers (greater than 10^{100}) chosen at random and the decryption key. His security is that there are no quick ways of factoring large numbers into prime factors using conventional computers. The size of his key is not fixed, meaning that if the factoring of RSA modules employees is committed; keys with greater lengths can be chosen to maintain the security of the cryptosystem.

Should be noted that one of the most important applications of public key cryptography is the digital signature. The origin signs a message with its private key.

B. Signature and Digital Certificate

The RSA algorithm is reversible; ie, in addition to allow the public key encryption and decrypt the message with the private, it also allows to encrypt with the private key and decrypt with the public. This latter mode of encryption does not provide confidentiality because anyone can decrypt the original message since they can always get the public component of the speaker, however, encrypt a message with the secret key of a user involves a clear identification (so get the authenticity and non-repudiation) and that only with the key associated with their identity can decipher, as does a handwritten signature, so this process is known as Digital Signature.

A Digital Signature [17] consists basically on three parts:

1. Key pair generation, private (with which it is signed) and public (with the one verified by a third party). This key generation is done according to a particular algorithm, as we have been seeing before: the RSA.
2. Signature of the document. With the private key signed the message.
3. Signature verification by a third party. Given the signature and public key, another user can validate the signature.

As the computational cost of public key algorithms is fairly high, to sign a large amount of data, when a message is large, the digital signature could be extremely slowly. For all this, is applied to the document a summary way function (hash function) to obtain a hash value, which is only a summary of the document. The digest or hash functions [18] are used to compress a text or document in a fixed length block. Hash functions should be public and irreversible, that is, from the abstract can not recover the original text. Not encrypt only compressed text or documents in a fixed length block. The hashing algorithm SHA-1 has been examined closely by the public cryptographic community, and has not found any effective attack.

The Digital Signature, thanks to the asymmetric cryptographic algorithms, can replace the traditional signature on paper, as it offers these features:

- Document integrity. If the document was modified during transmission, the signature verification will be missed.
- Identity and authenticity. Only a public key associated with the user who signed with his private key can correctly decrypt the digital signature.
- Non-repudiation. The user can not deny that he signed his authorship of the Digital Signature.

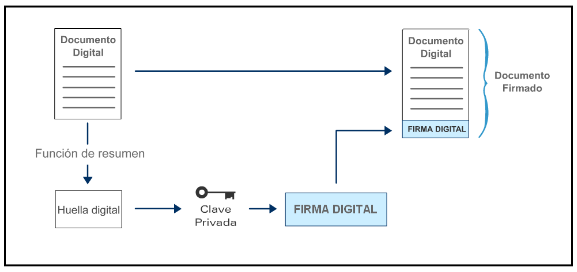


Figure 2. Generating Process of a Digital Signature [15].

The digital signature ensures data integrity and authenticity, so that errors caused during transmission, it would become apparent during the process of signature verification:

- Separate content unencrypted (original) of the signature itself.
- The receiver proceeds to calculate the hash of the received document according to the algorithm used (in this case, the SHA-1). As a result you get 160 bits.
- Now we proceed to calculate the decryption of the signature received with the sender's public key, and by the same asymmetric algorithm that was encrypted (RSA in this case). This gives a string of 160 bits should match the digital signature calculated in the previous step.

If both matches, the verification is correct, the document was signed by the issuer and data were no corrupted.

The format of the digital signature depends on the way in which they perform. It is important note the signature with syntax ASN.1 referenced to the standard RFC 3852 and it is based on the set of standards PKCS # 7 / CMS.

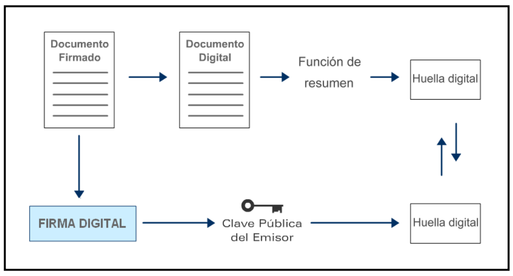


Figure 3. Verification Process of a Digital Signature [15].

A digital certificate [19] is a document issued and digitally signed by a Certification Authority that combines the distinctive name of a person or entity with its public key for a period of time. They are digital documents that serve to ensure the accuracy of the public key certificate belonging to the owner or the entity, which is digitally signed documents that can provide the most absolute security guarantees.

This mechanism provides the users to verify the authenticity through the network, controlling access to resources, etc. There are several formats for certificates and the most widespread is the X.509 version 3. This format is a standard of ITU and ISO / IEC.

C. *Security on Mobile Devices*

Different technologies are capable of offering digital signature capabilities and user authentication using a mobile phone. There are several articles about this [20], [21], [22]; however they are abstracts and with unrealistic solutions. Then, we explain those most important tools that have been used for the development of the platform.

We can attend to commercial and technical reasons [23] for the selection of the platform. In the first, we will consider the market penetration of the different alternatives. As for technical reasons, we will study the speed and simplicity of coding and prototyping. The following image shows an analysis of the status of the market related with installed based vs available apps disparity. Here we can see that the "old guard" (Symbian, Java ME and flash) has a larger installed base, but a smaller number of available apps than the newer platforms (Android, Iphone and BlackBerry).

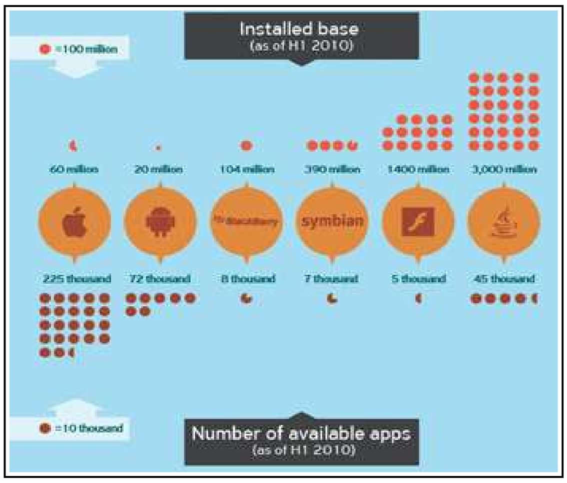


Figure 4. Installed base and number of available apps [24].

If now we attend to the quick coding and prototyping, we can consult another study reflected in this image. Here, we see that the easiest platform to master is Android (5 moths), while the hardest is Symbian (15 months).

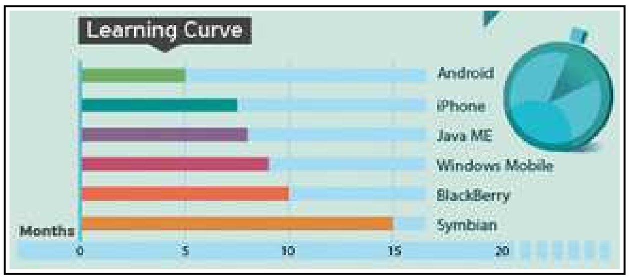


Figure 5. Platform´s learning curve [24].

In this article, we have chosen the Java ME platform for the rapid development and great penetration into the market. Moreover, this solution has low cost about developer tools.

1) *Java Mobile Edition*

Java ME, formerly known as Java 2 Platform Micro Edition or J2ME, is a technology for developing Java applications on mobile devices such as phones and PDAs. Java ME consists on programming specifications and a special virtual machine which allows that a Java ME program can be run on a mobile device. JavaME services are based on local programs, called MIDlets, which the user can download and install on his terminal. MIDlets can be run locally on the device and also provide client-server sessions. There are two settings for Java ME, configuration for devices with limited connection (CDLC) and other for connected devices (CDC).

The development of a Java ME application follows several steps, which we detail in four:

- Edit the source code.
- Compilation: The code is compiled and the class files are obtained.
- Pre-verification: To prevent malicious applications are installed.
- Packaging: It generates a .JAR.
- Deployment: Installing MIDlet on the mobile terminal.

Noted that the Java ME cryptographic operations are performed with the addition of cryptographic libraries according to the standardized set of APIs in the Java Cryptography Architecture (Java Cryptographic Architecture - JCA).

2) Security and Trust Services APIs

API Security and Trust Services (Security and Trust Services API) [25] is a specification for Java ME and give the possibility of opening a channel of communication between a Java MIDlet and a "security element." This security element can be a smart card with cryptographic module WIM, or the internal security element that provides the S.O. of a terminal.

Using SATSA, JSR-177 is also a security element which can perform all security processes such as electronic signature or authentication of users in Java ME applications. The API of Satsa has four optional packages for the different needs of communication with the security element. The communication mode depends on the type of application. These packages are APDU and JCRMI, intended to communicate with smart cards. The package PKI is used for credential management and digital signature. Finally, CRYPTO is used to perform cryptographic operations.

3) Symbian OS

This operating system [26] has been developed exclusively for mobile terminals and its code is provided to major phone manufacturers like Nokia. Provides important functions related to authentication, and confidentiality and integrity of data. Also, Symbian allows the management of certificates with a cryptographic module, and implementation of standard cryptographic algorithms, hash functions, key and random number generation.

These technologies have been used to create some examples of existing services such as Mobipay or some proprietary driven by large operators such as Vodafone and Telefónica.

III. DEVELOPMENT OF THE PLATFORM

The basic objective is to implement a digital signature mechanism in a mobile phone. Of all the existing technologies, Java is chosen to try to ensure more widely as possible among the existing terminals. For it, a Java ME application (MIDlet) for a Nokia N95, has been developed which uses the private key of a personal digital certificate from the Fabrica Nacional de Moneda y Timbre, issued in the conventional way, and so perform cryptographic tasks in the mobile digital signature.

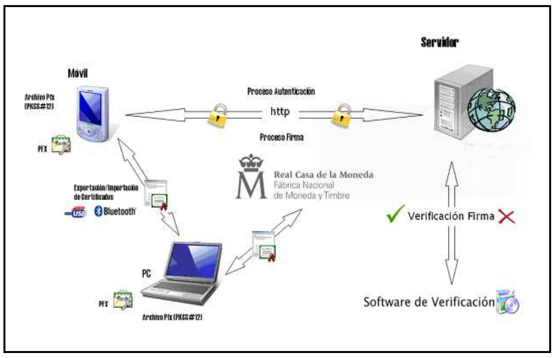


Figure 6. General scheme of the Platform [27].

The main elements involved in this platform are:

- Mobile phone N95. With Java and JSR-177 factory installed.
- Conventional certificate in PKCS # 12 format issued by the Fabrica Nacional de Moneda y Timbre.
- Server with Tomcat version 6 installed.
- Development environment NetBeans IDE 6.0.1.
- Software digital signature verification.

We will study the environment of the two most important components of this platform, the client and server.

A. Client

To give the client the functionality required for this platform is necessary, to incorporate into the mobile device (N95); two fundamental elements: the digital certificate and the MIDlet [28]. These will be discussed below, however, previously it would be useful to show the structure that contains the device protocols:

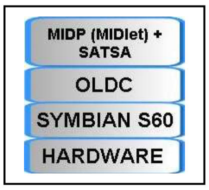


Figure 7. Structure of protocols of the mobile device [29].

1) Digital Certificate

The first thing is to have a personal digital certificate issued by the Fabrica Nacional de Moneda y Timbre. The steps to follow to obtain are the following:

- Request the certificate through the internet.
- Proof of identity in a registry office.
- Download digital certificate user.
- Export Certificate.

Once obtained the personal digital certificate in PFX / PKCS # 12 format, is exported to the mobile terminal. For it; the transfer file has been made through Bluetooth from the PC although it could be used for a USB cable. In the inbox is stored the message received by Bluetooth, which contains the file with the personal digital certificate and private key. When we try to open this message; a key is requested, which was inserted during the previous export stage. When you enter the password correctly the phone detects that contains the file and will proceed to the installation of the certificate.

Thus, the private key is stored in the security element and the user certificate and the certificate authority in his default store.

2) Midlet

The application on the client side consists of a MIDlet called MozartPKI. This MIDlet has been developed for terminals with profile MIDP 2.0 and limited CLDC 1.0. connection settings. The MIDlet has been developed so that, using SATSA cryptographic services, a user can send to server a plain text message digitally signed by him, and that another person may recover the server and check it later. The MIDlet consists of 2 classes in the package es.minerva.mozart. A MIDlet class itself is called MozartPKI.java and the other is Infodata.java. The Infodata.java class is responsible for reading the IP address of an external file server, so if you want to change the IP delivery of the signature file (the server), we would not need to tweak the source code, just enough to change the file properties.

To install the MIDlet the first thing to do is compiling and to package the source code and resources of the client application. For it, NetBeans 6.0 is used for the development environment (compile and build the JAR file).

To install on the mobile phone it is enough to send the JAR file, which is the MIDlet itself. For this test, we will use the Bluetooth technology. Thus, once the mobile has received the JAR file, is able to install the MIDlet. With the MIDlet installed on the Nokia N95 can be checked by accessing the phone's application menu and noting that this application is available as the others:

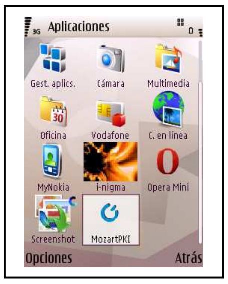


Figure 8. Correct installation of the MIDlet "MozartPKI" [27].

B. Server

This server consists of a Java Servlet, ServerPKI with a web page, index.jsp. Java Servlets [30] are objects that are within the context of a servlet container (in this platform will be Tomcat 6). The Servlet works very similar to the MIDlet, when the client that runs on mobile phone, establishes a connection to the server and sends the POST request, the servlet responds by trying to keep the inflow as array of bytes in a file on his disk. If the process is successful, it opens an output stream to send the MIDlet acknowledgment message indicating that the process has finished successfully.

The Java Servlet is compiled by NetBeans and which has been developed to run on the server. As a result of the compilation and construction of the Web application; a file .WAR is created in NetBeans and must be installed on the Servlet container, which in this case is Tomcat 6.0. To do this you access the Tomcat management console and select the WAR file to deploy it for.

IV. RESULTS

A. Client

Accessing MozarPKI, we see the operation of this MIDlet. Once launched, the welcome screen appears like you can see in the image and the entrance to the right:

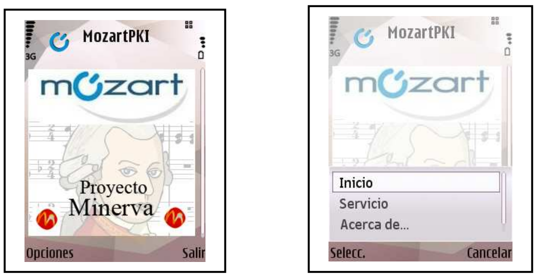
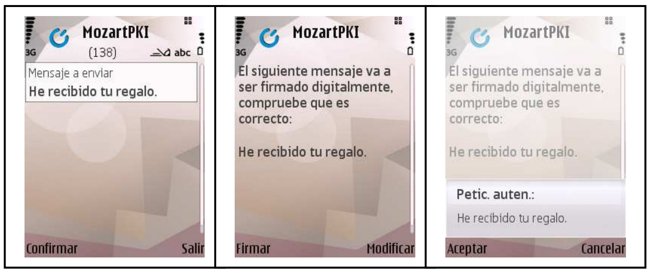


Figure 9. Screen of welcome and beginning of the N95 [27].

First, the MIDlet requests to the service user, the insertion of the message to sign, and once you have written is confirmed by pressing the corresponding button. The message to be signed by way of example is "I received your gift". The message is then introduced and offers the possibility to modify it if the client detects a fault in writing:

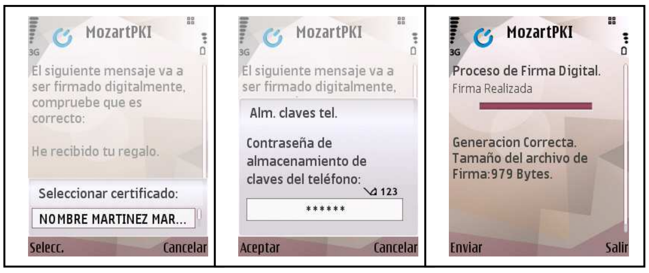

MozartPKI
(138) abc
Mensaje a enviar
He recibido tu regalo.
Confirmar Salir

MozartPKI
El siguiente mensaje va a ser firmado digitalmente, compruebe que es correcto:
He recibido tu regalo.
Firmar Modificar

MozartPKI
El siguiente mensaje va a ser firmado digitalmente, compruebe que es correcto:
He recibido tu regalo.
Petic. auten.:
He recibido tu regalo.
Aceptar Cancelar

Figure 10. Text editor and confirmation [27].

If the user checks that the message entered is correct, click on the button "Sign" and the cryptographic process start. The security element of Symbian notice that an application (the MIDlet) is requesting access to it, and made a prompt the user to accept if you agree. Once the user has accepted the request from the MIDlet, to access the security element, SATSA does a "sweep" by the security element which, in the case of Nokia N95 is the internal operating system Symbian. The installation of the user's certificate is found and is shown on screen in order to be selected.


MozartPKI
El siguiente mensaje va a ser firmado digitalmente, compruebe que es correcto:
He recibido tu regalo.
Seleccionar certificado:
NOMBRE MARTINEZ MAR...
Selecc. Cancelar

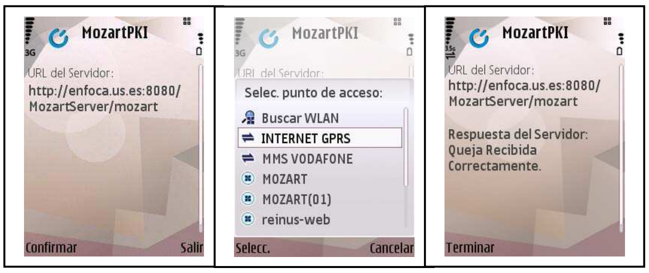
MozartPKI
El siguiente mensaje va a ser firmado digitalmente,
Alm. claves tel.
Contraseña de almacenamiento de claves del teléfono:
******
Aceptar Cancelar

MozartPKI
Proceso de Firma Digital.
Firma Realizada
Generacion Correcta.
Tamaño del archivo de Firma:979 Bytes.
Enviar Salir

Figure 11. Certificate selection, access to elem. Sec And Signature [27].

When the user selects the certificate that he wants, SATSA tries to access the private key. For security reasons, it is under a password. If the key is entered correctly SATSA proceeds to perform the digital signature of the inserted data in the format CMS / PKCS # 7 SignedData containing the signer's certificate, the data that have been signed and the digital signature itself.

Once the signing process is complete, the file containing the digital signature should be sent to the server which is running the Servlet developed. When the user clicks on the button "Send" starts the process and then the MIDlet retrieves from the file properties; by the methods of the class InfoData.java, the URL of the server and displayed on screen for the user to confirm. The user must also to confirm the request of the MIDlet to connect to the Internet to send the file to the server, since this connection could lead to charges associated with the carried through the GPRS / UMTS. Once the HTTP connection is established the MIDlet looks forward to receiving the consent of the server.


MozartPKI
URL del Servidor:
http://enfoca.us.es:8080/MozartServer/mozart
Confirmar Salir

MozartPKI
URL del Servidor:
Selec. punto de acceso:
Buscar WLAN
INTERNET GPRS
MMS VODAFONE
MOZART
MOZART(01)
reinus-web
Selecc. Cancelar

MozartPKI
URL del Servidor:
http://enfoca.us.es:8080/MozartServer/mozart
Respuesta del Servidor:
Queja Recibida Correctamente.
Terminar

Figure 12. URL's obtaining and sending to Server [27].

The MIDlet, as detailed above, sends the CMS file and looks forward to receiving the assent or the server-side error. If everything works properly, the message "OK" is got on screen as shown in the previous image. The process has been completed for the MIDlet as signed message has been stored correctly as indicated by the assent of Servlet.

B. Server

Looking index.jsp, the website of Servlet is presented: Server Status and message received.

The followings images shows the appearance of the website. The link server status is available to inform the user if the service is active or not. To download the signed file, click on the link and opens the download window.

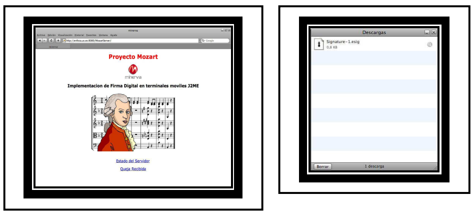

Proyecto Mozart
Implementacion de Firma Digital en terminales moviles J2ME
Estado del Servidor
Queja Recibida
Descargas
Signature-1.ezig

Figure 13. Server Web page and download window [27].

C. Verification

The PKCS # 7 / CMS file, that contains the signed data; needs to be checked to complete the process and check that the signature has been generated by the mobile correctly. To carry out the process of verifying of the digital signature; by the user on their mobile device, we have used the free tool eSign Viewer [31].

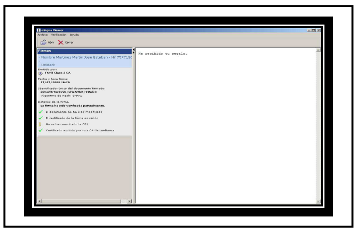

He recibido tu regalo.

Figure 14. ESigna Interface Viewer [27].

This tool allows three things: view the contents of the signed, the signatures and the validity of the signatures applied to the same.

In the image, right, appears correctly separated the message sent from your digital signature. On the left, the program displays the report with the results of opening the file and checks the signature correction. This information shows how Esigna tool verifies if the signature made by the J2ME application in the mobile terminal is valid and if complies with the PKCS # 7 / CMS format. A third party who want to verify the authorship of the message can do it and be sure who sent it.

V. FUTURE WORKLINES

Three future worklines can be analyzed, taking into account the most important elements of the platform made:

We used certificates issued by the Fábrica Nacional de Moneda y Timbre (FNMT) since it is considered the major Certification Authority in Spain. In private business, the FNMT certificates are not used, because so the complicated problem of the validity check in LDAP directories is released, which have restricted access. This would be a possibility for a future attempt to open the application, and integrate developments, certificates issued by other Certification Authorities as Camerfirma.

This Project has used Java ME with libraries SATSA for signing and certificate management. It could mean a restriction on developments since the service provider may require other forms of signature or the use of certain cryptographic algorithms that do not implement this API. At this point, it is worth mentioning the possible to use of other Java libraries like BouncyCastle or explore alternatives for the integration of cryptographic cards in mobile devices.

The fact that the digital signature has been implemented in the development of this project is conducted in CMS format, ensures interoperability of the developments made. In this respect, as a possible future work is the integration and improvement of the application developed in Java ME for interconnection with a public platform or other third parties wishing to provide service.

VI. Conclusions

The main goal was to develop a study of the state of the art in the field of digital signature applied to mobile telephony as well as investigate the possibility of integrating the use of digital certificates on mobile devices business. In this aspect, we have managed to integrate a digital certificate on a mobile phone and use it for services of authentication, digital signing of data and establish secure connections. All this in a closed and controlled environment, but easily extended to more realistic situations.

On the other hand, the field of digital signature and certification is currently at a low growth and maturity among the population, and less on mobile use, so job opportunities are numerous. The truth is that, as noted in this article, there is sufficient technological resources to address the field of cryptography and digital signatures using digital certificates. All that is needed is investment and effort to try to get what may be another big boom in mobile communications.

References

[1] Kim Mooseop, Ryou Jaecheol, and Jun Sungik, “Compact Implementation of SHA-1 Hash Function for Mobile Trusted Module,” Information Security Applicaions, Volume 5379/2009, pp: 292-304, 2009. http://www.springerlink.com/content/f2v64u64324w6q47/

[2] The National Electronic Commerce Coordinating Council (NECCC), “Impact of Electronic Signatures on Security Practices for Electronic Documents,” 2001. http://www.azsos.gov/pa/ec3/Security_Practices_ED.pdf

[3] Trustgate Berhad. “The advent of an Interoperable Ecosystem for Secure Mobile Transactions,” 2009. http://www.msctrustgate.com/pdf/Mobile_Signature.pdf

[4] Lex Nova Magacine Report, “*Firma electrónica: Seguridad a través de la red*,” 2004. http://tecnojur.blogs.lexnova.es/2011/02/05/aspectos-basicos-de-la-firma-electronica/

[5] José Jesús Angel, “*Criptografía Simétrica*,” 2007. http://elsitiodetelecomunicaciones.iespana.es/cbasica.pdf

[6] R. L. Rivest, "The RC5 Encryption Algorithm," Proceedings of the Second International Workshop on Fast Software Encryption (FSE) 1994e. pp. 86–96, 1994.

[7] Joan Daemen, Rene Govaerts, and Joos Vandewalle, "Weak Keys for IDEA," Advances in Cryptology, CRYPTO 93. pp: 224–231, 1993.

[8] Shoji Miyaguchi, “The FEAL Cipher Family,” CRYPTO 1990, pp: 627–638, 1993.

[9] Lars R. Knudsen, "Cryptanalysis of LOKI," Advances in Cryptology - ASIACRYPT'91, LNCS 739, pp 22–35, H Imai et al. (eds), Springer-Verlag, 1993.

[10] Eli Biham and Adi Shamir, “Differential Cryptanalysis of the Data Encryption Standard,” Springer Verlag. ISBN 0-387-97930-1, ISBN 3-540-97930-1, 1993.

[11] Bruce Schneier, “ Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish),” 1993. http://www.schneier.com/paper-blowfish-fse.html

[12] "RFC 5830: GOST 28147-89 encryption, decryption and MAC algorithms," IETF, 2010-03. http://tools.ietf.org/html/rfc5830

[13] C.M. Adams, “Constructing Symmetric Ciphers Using the CAST Design Procedure,” Designs, Codes, and Cryptography, 12(3), pp. 283–316, 1997.

[14] L. M. Leyva, “Sistema criptográfico,” 2008. http://investigacion.uagro.mx/3coloquio/exa/11.pdf

[15] Figure 1, 2 and 3, Source: gdp.globus.org, 2009.

[16] Real Academia de Ciencias, “*Criptografía de clave pública*,” El sistema RSA, 2006. http://www.uam.es/personal_pdi/ciencias/ehernan/Talento/VicenteMunoz/rsa_2006.pdf

[17] Mauricio Devoto, “*Comercio electrónico y la firma Digital*,” 2008. 74.125.155.132/scholar?q=cache:sr3OWIhUeX0J:scholar.google.com/+firma+digital&hl=es&as_sdt=0,5

[18] Unizar, “*La seguridad en informática -Funciones Hash*,” 2005. http://criptosec.unizar.es/doc/tema_c7_criptosec.pdf

[19] Sergio Talens – Oliag, “*Introducción a los Certificados Digitales*,” 2003. http://www.uv.es/sto/articulos/BEI-200311/certificados_digitales.pdf

[20] Yu Lei, Deren Chen, and Zhongding Jiang, “Generating digital signaturas on Mobile devices,” 18th International Conference on Advanced Inf. Networking and Applications, 2004. http://www.computer.org/portal/web/csdl/doi/10.1109/AINA.2004.1283860

[21] Santi Jarusombat and Surin Kittitornkun, “Digital Signature on Mobile Devices based on Location International,” Symposium on Comm. and Information Technologies, 2006. ieeexplore.ieee.org/xpls/absall.jsp?arnumber=4141339&tag=1

[22] Scott Cambell, “Supporting digital in mobile environments,” Twelfth IEEE International Workshops on Infrastructure for Collaborative Enterprises; 2003. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1231414

[23] Vision mobile, “The mobile developer journey,” 2010. http://stackoverflow.com/questions/1414288/j2me-vs-android-vs-iphone-vs-symbian-vs-windows-ce

[24] Figure 4 and 5, Source: visionmobile.com, 2011.

[25] C. Enrique Ortiz, “*The Security and Trust Services API for J2ME,” Part 1, March,* 2005. http://developers.sun.com/mobility/apis/articles/satsa1/

[26] Fundación Symbian**,** **“***Symbian,” 2008.* http://www.xatakamovil.com/sistemas-operativos/fundacion-symbian

[27] Figure 6, 8, 9, 10, 11, 12, 13 and 14, Source: Own elaboration, 2011.

[28] Alessandro Distefano, A. Grillo, A. Lentini, Gianluigi Me, and Riccardo Galbani**,** **“***Communications in Computer and Information Science,” 2009.* http://www.springerlink.com/content/l8n978414l371t88/

[29] Figure 7, Source: Universidad de Málaga – sicuma.uma.es, 2011.

[30] Javier García de Jalón, J. Ignacio Rodríguez, and Aitor Imaz **“***Los Java Servlet,*” 2008. http://es.wikipedia.org/wiki/Java_Servlet

[31] Esigna Viewer, “*Visor gratuito,”* 2005. http://www.indenova.com/indenova.php?opc=5