Qualified Electronic Signature via SIM Card Using JavaCard 3 Connected Edition Platform

Jakub Breier
Physical Analysis and Cryptographic Engineering,
Temasek Laboratories@NTU
School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
jbreier@ntu.edu.sg

Adam Pomothy
Faculty of Informatics and Information Technologies,
Slovak University of Technology
Bratislava, Slovakia
adam.pomothy@gmail.com

***Abstract*—Digital signature is one of the most common ways of determining the origin of a document in a digital way. To ensure authenticity, integrity and non-repudiation when such signatures are used, many countries have their standards and regulations. In EU, a signature that complies with those regulations is called 'Qualified Electronic Signature' (QES).**

There are many QES solutions using dedicated smart cards or security tokens and few of them that use SIM cards as a signature creation device. These SIM-based solutions usually use a third party to perform a signature, such as mobile service operator and operate as a hybrid solutions. Hence, a cooperative connection between a mobile device and a SIM card is needed.

In this paper we propose a solution based on the JavaCard 3.0 Connected Edition platform that operate fulfills following conditions: it is a mobile service operator-independent and mobile phone operating system-independent. The first condition is achieved by performing all the operations directly on a SIM card and the second condition is satisfied by avoiding the application running on a mobile phone operating system. Instead, we propose a web based application to perform the necessary verification methods on the SIM card. So this proposed application can be accessed via mobile phone web browser. Of course, our solution satisfies the Common Criteria standard requirements for the EAL 4 level.

***Keywords*-Qualified Electronic Signature, digital signature, JavaCard, SIM Card, PKI**

I. Introduction

The most common way of determining the origin of a document is a signature. With the requirements of creation of such a signature without physical contact, a digital signature was invented. Digital signatures use asymmetric cryptographic algorithms, usually applied to a hash of a document we want to sign. The scheme of signing and verifying a document is the following. First, the originator signs the hash of a document with his private key and attach this signature to the document. Then the verifier can use the originator's public key to prove that the document was signed by the originator. It is obvious that the security of such a scheme is dependent on the security of a secret key and the integrity of a public key.

For the use of digital signatures in communication with government institutions, it was crucial to establish a legislation that sets the rules regarding on signing documents electronically. For example, the Directive 1999/93/EC of the European Parliament and of the Council [1] is the main European Union document related to digital signatures. It introduces the term 'Qualified Electronic Signature' (QES), which is a signature that satisfies security requirements necessary for achieving certain level of authentication, non-repudiation and integrity. This directive was then adapted by a member states of EU, but similar legislation exists almost in every other country in the world, with minor differences.

The usage of QES involves several components. A user needs a certified Security Signature Creation Device (SSCD), used as a storage of a secret key. This devices is usually a security token or a smart card. Then a computer with a special application is needed, handling a communication with the SSCD. If we use smart card as a security device, we also need a smart card reader.

Operation of the necessary components can be complex and non-intuitive for average technical experienced users. The whole process can be simplified by using a mobile phone for performing a QES, which is a common daily life item, nowadays. The main constraint of this solution is the security of the device, because mobile phones transmit and process usually a big amount of data that cannot be trusted.

This paper introduces a way to implement a QES on a mobile phone, by using Subscriber Identity Module (SIM) card as a key storage and signing device. Our solution uses two applications - one is running on a personal computer and handles document selection and communication with the SIM card, and the second one is the signing application on the SIM card. The main motivation to use a solution based on a mobile phone is the availability of such a device. According to The Mobile Economy 2013 report [6], global SIM penetration was supposed to reach 100% in 2013, growing up to 129% in 2017.

In our solution we use the functionality of the JavaCard 3.0 Connected Edition platform. The main advantage of such

a solution is the communication between a mobile device and a SIM card. The JavaCard 3.0 applet behaves as a servlet to the outside world, we can then directly access the SIM card not only from a mobile phone, but also from a personal computer, if it is in the same local network. This enables us to simplify the solution, so that no communication with the mobile operating system or its applications is needed.

The rest of this paper is structured as follows. Section II provides an overview of a related work dealing with the problem of qualified electronic signatures on SIM cards. Section III proposes our approach and describes method used for Qualified Electronic Signature on SIM card. Section IV provides implementation details of our solution. In section V are stated main parts from the document specifying compliance with the Common Criteria EAL 4 level. Section VI provides discussion of our method, and finally section VII concludes this paper.

II. Related Work

There are two basic mobile digital signature models. According to Samadani, Shajari, and Ahaniha [14], we can use either a client-based or a server-based model. A server-based digital signature is a signature created by a service provider for a mobile client. In this approach, the secret key is stored on a server, therefore a mobile phone is used only to send a request and receive a signature. In opposite, client-based signatures are created either by a mobile phone or by a SIM card, so the secret key is stored in the device owned by a signer. It is more convenient to implement a client-based signature model, the server-based model has several shortcomings according to Rossnagel [11].

As stated by Ruiz-Martínez et al. [12], we can further differentiate the client-based model into three categories. The first category are SIM card based solutions, then we have solutions providing signatures with the use of a handheld device, and finally, we have hybrid solutions, that require collaboration between the SIM card and a mobile device. Obviously, the second category does not comply with the requirements of the QES, because a standard mobile phone does not provide a secure storage for the secret key.

Pisko [10] proposes two types of a mobile signature interaction between the user and a mobile signature application: server-based and application-based. The first one is initiated by a web server, communicating with the mobile device, the second one uses a message communication. Both cases are, however, server-based, so they cannot achieve requirements for the QES.

Turkish mobile phone operator *Turkcell Mobile* provides the only working solution for the qualified electronic signature, entitled Avea Mobile Signature[1]. The main purpose of these signatures are banking operations and the solution was created in a cooperation with Gemalto, one of the main SIM card distributors. It is a proprietary solution, using GSM technologies (Short Message Service) and the Internet. The design does not allow to sign arbitrary documents and is dependent to the mobile operator.

[1]http://www.isbank.com.tr/English/content/EN/Expatriate_Banking/Security/Avea_Mobile_Signature-1089-413.aspx

III. Design

Our solution can be classified as a SIM card client-based solution, using the SIM card as a device for key storage and a signature creation, and a personal computer as a communication device that sends the hash of a document to the SIM card and obtains a valid QES. The verification if the received hash is the one we want to sign is done by using JavaCard based web application, that can be accessed via a web browser on a mobile phone.

The only requirement for the mobile phone is a capability to connect to a Wi-Fi network so that it can become a node in a same network as the personal computer with the signing application is. The requirement for the SIM card is a compliance with the JavaCard 3.0 Connected Edition standard.

A. JavaCard Platform Overview

Before designing the QES, we need to prove that the JavaCard technology satisfies the security requirements and therefore the SIM card is fully capable to act as a SSCD.

JavaCard platform emanates from the Java platform and both share some major security controls [5]:

- Strong data-type control.
- Automatic memory management.
- Bytecode verification.
- Secure class loading.

Besides these, JavaCard provides even more security controls, not encompassed in a standard Java platform:

- The state of objects stored in the RAM does not change after an unexpected termination of a power supply.
- Behavior of all applets is controlled by the Applet Firewall, which ensures that the applet cannot read or make changes that are outside its context. It can share objects among other applets only via a special interface.
- Applets that do not originate from the manufacturer do not have access to native functions of the SIM card.
- JavaCard API provides extensive cryptographic functions that enable secure installation and verification of applets, or storage of encrypted objects in the memory.
- The whole functionality is provided exclusively by applet interfaces. A mobile device can send a request to an applet and obtain a response in a form of messages. There is no direct access to data or functions on the card.
- All data, including application data, is stored in an isolated environment of a JavaCard Virtual Machine.

The platform uses the ISO 7816-4 [7] standard, specifying rules and protocols used for the external communication. For

the information exchange, special messages called APDU (Application Data Unit) are used.

Since version 3.0, JavaCard platform is divided into two versions. JavaCard 3.0 Classic Edition is the evolution of the previous 2.2.2 version. It provides more extensive support of contact-less smart cards and implements new cryptographic algorithms.

The second version is the JavaCard 3.0 Connected Edition [2] and it introduces a completely different model of communication between the host device and a smart card, using the Java Servlet API. The host device recognizes the smart card as a web server. It is designated for host devices that are able to connect to various types of networks, using IP based protocols like TCP, TLS, HTTP, or HTTPS. A mobile phone is therefore a suitable device, while its network connection capabilities can be used by applications running on SIM card.

B. Signature Creation Process

The process of the signature creation is following. First, both a personal computer and a mobile phone have to be in the same local network. User will start the PC application and authenticate himself by his credentials, that were registered on the first run of the application. Then he sets the IP address of the phone and choose a document he wants to sign.

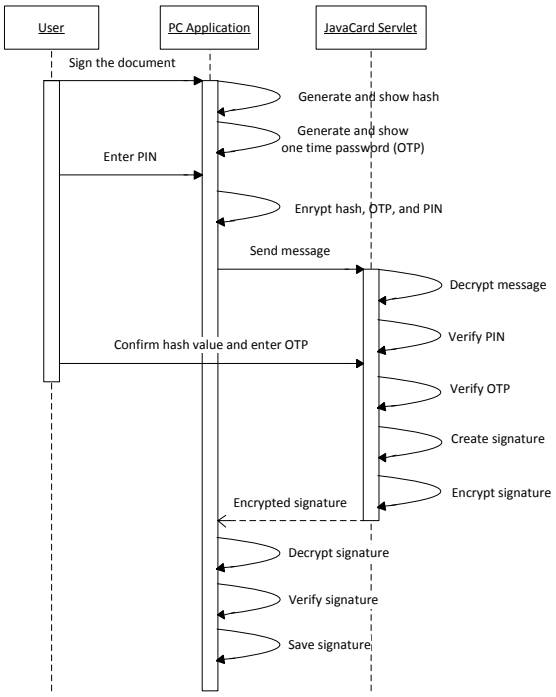

Figure 1. Signature creation process sequence diagram.

In order to send the hash of the document, sign it on a SIM card and send it back to the personal computer securely, we have to figure out following problems:

- The communication between the PC application and the SIM card over LAN is not secured by default.
- PC application does not know if it sends data to the authentic signing servlet.
- JavaCard servlet does not know if the received hash originates from the key owner.

One option would be to use the capabilities of the mobile phone, which was our first design. In this model, we used the Diffie-Hellman key exchange, standardized by the PKCS#3 and then the SSL channel for the secure communication, using symmetric cryptography.

The previous option would be secure and it would comply with the QES requirements, however it would have needed a computational power of a mobile phone. Since we want to minimize the platform dependence, we propose a second solution. This solution uses a key pair generated on a SIM card. The secret key stays in a protected environment of a SIM card and the public key can be obtained only by a Certification Authority (CA), that will produce a public key certificate and provide this certificate in an encrypted form to the key owner. Then he assign this key to the PC application, that will use it in a communication with the SIM card. Therefore the communication is confidential and authenticated.

The other security mechanism is a one time password, that is showed to the user together with the hash. The message from the PC application then contains the hash, the PIN code and the one time password. This message is decrypted by the secret key on the SIM card and the user identity is verified by inserting a correct PIN code in the PC application. If the verification is successful, the hash and the one time password are temporarily stored in a SIM card memory. After this step, the user confirms the hash value by comparing it with the value in the PC application. This hash can be viewed in a web browser of a mobile phone. The user then inserts a correct one time password through the same web interface. After this step, both a user identity and data integrity is verified.

The next step is signing of the hash, encryption of the signature by the secret communication key and sending this cryptogram to the PC application. Acquired signature is then verified by using a user's public key certificate. The sequence diagram of the whole process is depicted in Fig. 1.

C. Installation of the Signing Application on the SIM Card and Generating a Public Key Certificate

The first step is a user's application for a public key certificate resources from the CA that are necessary for the QES. CA then installs the JavaCard application on a user's SIM card and provides him with the PC application. In the process of the applet installation, a new key pair is generated on the SIM card. The public key can be obtain from the SIM card only once. CA will create a certificate for this key in a secure environment and provides user with this certificate.

CA will create the communication key in the same manner and store the key for the case the user looses it.

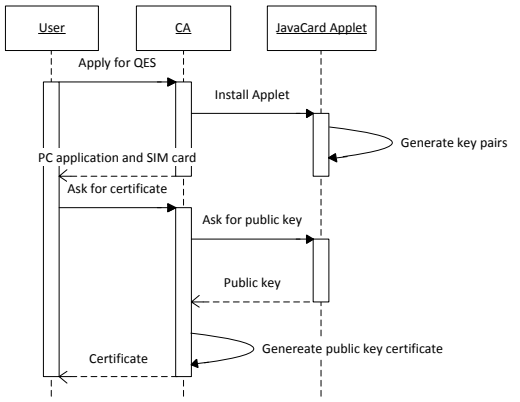

Figure 2. Application and issuance of the QES resources.

Through the whole process, secret keys do not leave the secure device (SIM card). Furthermore, this solution is more secure than today's methods where the creation of keys is performed on a CA side, not on a clients side by using his device. The sequence diagram of this process is depicted in Fig. 2.

IV. Implementation

In this section we will describe our implementation of the design proposed in previous section. Our solution is entitled the 'DigiSigner'.

A. PC Application

The main purpose of this application is to handle documents to be signed. We used Java SE platform together with third-party libraries (Jasypt, BouncyCastle, PDFBox). The application provides following functions:

- View the documents to be signed.
- Creation of a hash of the document and encrypting functions.
- Communication with the mobile application via local network.
- Certificate management and signature verification.

In the next subsections we explain security mechanisms used in the application.

1) User Authentication: The first security mechanism is the user authentication, which is required after each start of the application. In the first run of the application, the user is required to register by his login and password. For the secure password storage we use the Jasypt[2] library, more specifically the *StrongPasswordEncryptor* class. This class generates 16B salt which is concatenated at the beginning of the message and then encrypts the message 100 000 times by using the SHA-256 algorithm.

[2]http://www.jasypt.org

Table I
Initialization message structure.

PIN code	OTP	Hash function identifier	Hash value
2B	4B	19B (For SHA-256)	32B (For SHA-256)

2) Document Import and View: It is necessary to obey a WYSIWYG principle (What You See Is What You Get), so that an adversary cannot substitute the document to be signed between the time of choosing the document by the user and opening the document in the application. The user can therefore view the document he is going to sign and he can also view the public key certificate directly in the application. For viewing PDF documents we used the PDFBox[3] library.

3) Communication With the SIM Card: The communication with the SIM card is done by exchanging messages using the HTTP protocol. As a communication address, the IP address of the mobile phone, a port and the string identifying the target servlet is used. The address has following format: http:/192.168.1.13:8019/digisigner/providePublicKey.

4) Sending and Encrypting Messages: Since exchanged messages contain sensitive data, it is necessary to use cryptographic techniques for their protection. On the mobile phone's side there is a servlet, that listens and waits for requests. The first message is always sent from the PC application. To minimize the number of messages, the initialization message already contains all the data required for the signature creation. The structure of this message is in Table I.

After comparing hash values by user (these are displayed both in PC application and in the web interface on a mobile phone), he enters one time password (OTP) in the web interface. Then he confirms the equality of hashes also in the PC application.

As stated in the design section, the SIM servlet provides the communication key only once. CA uses this key for the creation of public key certificate and stores it in the Java KeyStore[4] (JKS), which is the special storage place of the Java platform aimed for storing sensitive data as certificates, public and secret keys. The password to JKS is known only to the CA. This key store is then provided to the user who keeps it in the directory together with the PC application.

Encryption of messages is performed by using BouncyCastle[5] library. Using this library, we can choose a cipher algorithm, a block mode and a padding method. Selection of these parameters is determined by the capability of the JavaCard platform to decrypt the message. For our purposes we have chosen 2048-bit RSA with the electronic codebook (ECB) mode and the PKCS#1[8] padding method. Since we encrypt only one data block, the ECB mode is not a security

[3]http://pdfbox.apache.org/
[4]http://docs.oracle.com/javase/1.4.2/docs/tooldocs/windows/keytool.html
[5]http://www.bouncycastle.org/

risk in this case. The same parameters are used both in encrypting messages and signing, but using different keys.

5) Generation of Hash Value: For the hash generation, the SHA-256 algorithm is used. This algorithm complies with requirements of the QES. We used the BouncyCastle implementation also in this case in order to maintain compatibility among different versions of Java platform and different virtual machines, since the usage of native libraries could be problematic.

Only the hash of the document is sent to the SIM card. It is mandatory to attach a prefix to the hash value, defining a type of a hash algorithm, otherwise a verification against the certificate would fail. The identification bytes used in this prefix are stated in the PKCS#1 standard.

6) Verifying and Storing Signature: It is necessary to check the integrity of the message containing the signature after receiving. This is done by using the public key certificate. The signature is stored for the further usage after the verification.

7) Securing the Application Source Code: All the other security mechanisms would have been useless if an attacker had an opportunity to change the application source codes. To ensure the integrity of the application, a checksum is computed after application launch and it is compared to the value stored on a CA website.

To minimize a possibility of a decompilation, the application was transformed from the executable *jar* file to an *exe* file, using the JSmooth[6] tool.

B. JavaCard Application

The application consists of two parts: servlet part provides services similar to the classic application server and uses all the benefits of the JavaCard 3.0 Connected Edition platform and application part that handles functions associated with the signature creation.

1) Servlet Part: Servlet is a special type of the Java class that can handle requests and provide responses. This communication uses the HTTP protocol. Servlets are a common component of Java Enterprise Edition applications.

We created four servlets that provide the interface between the application logic on the SIM card and the outside world. These servlets inherit from the *HttpServlet* class and implement its *doPost()* method. Sent and received messages are formatted as a plain text. Transmitted data are in hexadecimal format. Below we provide a description of our servlets:

- **PublicKeyProvider:** this servlet provides the public key for signatures. The key is obtained from the SIM card as a set of parameters that describe this key. A key construction is then the responsibility of the application that requested this key - in our case it is the PC application. This servlet responses only to the requests from a local network. For this purpose, the *LocalNetworkFilter* class is used to filter IP addresses (in accordance with the RFC 1918[7]).
- **CommunicationPublicKeyProvider:** this servlet is almost identical to the previous, however it provides the communication key instead of the signature key.
- **DigiSignerCE:** this servlet is responsible for obtaining an encrypted message and sending it to the class that manages the communication key pair which is able to decrypt this message. The servlet extracts PIN code and OTP and then stores data to be signed in the *HashHolder* class. It returns the digital signature or the state describing failure of particular operations executed by the servlet. It responses only to the local network requests.
- **HashValidator:** this servlet provides a web interface, that shows hash value to user. It is also used for entering the OTP, verifying user identity and confirming that hashes are equal. This servlet responses only to request incoming from the host device - mobile phone. To perform this check, a HTTP servlet filter was implemented as a *LocalHostFilter* class.

2) Application Part: Classes within the application part handle all the key operations that are requested via servlets. JavaCard platform provides two objects that can be used for digital signatures: *javacardx.crypto.Cipher* and *javacard.security.Signature*. The second one is dedicated especially for digital signatures, however it does not work with hashes, it makes a hash from data by itself and then signs it. Since we are working directly with hashes, we are using the class inherited from the *Cipher* object.

We also have to check if the entered PIN code is correct. For this purpose we use the *JCSystem* class which enables access to system parameters, more specifically its *getAppProperty()* method which takes as the parameter an alias of a requested field.

The design we have proposed requires both short and long term storage of the state of some variables. We identified three classes for this purpose:

- **HashHolder:** handles temporary storage of the hash value intended for the signature. The storage time is 2 minutes. It also stores information whether the hash value was verified by the user and checks a time limit for this verification.
- **KeyExportStatusHolder:** controls limits for the export of the public key from the signature key pair.
- **CommunicationKeyExportStatusHolder:** controls limits for the export of the public key from the communication key pair.

[6]http://jsmooth.sourceforge.net/

[7]http://www.ietf.org/rfc/rfc1918.txt

V. Common Criteria Compliance

In order to comply with the requirements for the QES, we have checked the compliance with the Common Criteria 3.1 [3] standard, more specifically with the EAL 4 level.

The standard uses Protection Profiles, which are documents created by groups of security professionals and contain sets of requirements for target products. These profiles are used for the software evaluation. For our purposes, we have chosen the *Protection profiles for Secure signature creation device* [4] protection profile, which is designated for the evaluation of digital signature creation appliances.

There is no space for the whole evaluation, so we will describe only few parts that are important from the QES point of view. Below are described security mechanisms, differences between the ESCA [9] protection profile and finally, product restrictions.

To summarize the compliance, instead of security mechanisms defined in the protection profile, our security target focuses on following:

- Authentication of the user against the PC application.
- Authentication of the user against the key store on the personal computer.
- Authentication of the user against the SIM card.
- Storing sensitive data.
- Encryption of communication between applications.
- Verification of hash value inside the SIM application before signature creation.
- Verification of a created signature.

The security target was developed with the requirements of the ESCA [9] protection profile, however it differs in following articles:

- DigiSigner shows a document to be signed in a raw form and does not use external tools for the document viewing.
- DigiSigner creates a signature for one document at a time only.
- DigiSigner does not allow to choose a signature policy. Only one policy is supported.

Restrictions of the target product are following:

- **Correctness and integrity of generated key pairs:** secure generation of key pairs is handled by the JavaCard platform and it guarantees their correctness and integrity. DigiSigner SIM application has no impact in this process.
- **Integrity of personal computer operating system:** DigiSigner PC application does not have ability to check the integrity of a host computer operating system. User has to make sure the operating system is not compromised.
- **Security of cryptographic operations:** all the cryptographic operations are handled outside of application libraries. Their documentation specifies their security level and it cannot be influenced by the DigiSigner.

VI. Discussion

The main advantage of our solution is its simplicity, achieved by using the JavaCard 3.0 Connected Edition platform. It enabled us to completely avoid mobile phone operating system functions, that are sources of many security flaws. Our SIM card-only solution is easy to use, yet secure, using security mechanisms in order to preserve confidentiality, integrity and non-repudiation in the whole signature creation process.

The closest work proposing QES with the usage of a mobile phone device was published by Ruiz-Martínez et al. [13]. The authors propose a solution based on European Technical Standard Institute (ETSI) standard named Mobile Signature Service. The signing process in their case is dependent on a Mobile Service/Application Provider, who accepts signature requests. Therefore, the third person is necessary in the process, which makes it more complicated and there is more space for malicious actions. Also their mobile application is platform-dependent.

Besides this approach, there are some commercial solutions that depend on a mobile service provider, these solutions are proprietary and cannot be analyzed from the security or functional point of view. One solution that is successfully used is provided by *Turkcell Mobile*, its description is provided in section II.

VII. Conclusions

In this paper we presented a novel procedure to create Qualified Electronic Signature on a mobile phone, using SIM card as a creation and key storage device. Our solution complies with requirements of the Directive 1999/93/EC of the European Parliament and of the Council [1] and fulfills requirements of the Common Criteria EAL 4 level for the Secure signature creation device protection profile.

Our design uses JavaCard 3.0 Connected Edition platform which provides new possibilities and allows us to implement an elegant solution that is independent of a mobile phone operating system. It enables us to use the SIM card as a portable web server by communicating with the outside world via the standard HTTP protocol.

Our solution consists of a two applications, one is implemented on a SIM card while the other on a personal computer. The PC application hashes any provided document and send it directly to the JavaCard application, which is then used to sign the hash with a 2048-bit RSA-hashed signature. For verification purposes the user needs to verify himself at the PC application via a password as well as with the SIM card PIN at the signature application running on the mobile phone. The hash of a document can be viewed in the web interface of the SIM card application in order to determine if the correct document is signed. The communication between both applications is encrypted by symmetric cipher. A one time password is exchanged in

the beginning of the communication to ensure a confidential communication.

Since JavaCard 3.0 platform is relatively new, no SIM cards using this platform are still not available on the market at the moment. Apparently, most of these cards support only JavaCard 2.x platform, which lacks the necessary functionality for our solution. We expect the expansion of the new platform in the next two years, which will then allow our QES solution to become available to end users.

REFERENCES

[1] *Directive 1999/93/EC of the European Parliament and the council of December 1999 on a Community framework for electronic signatures*. 1999.

[2] *Application Programming Notes, Java Card 3 Platform, Version 3.0.2 Connected Edition*. Oracle, 2010.

[3] *Common Criteria for Information Technology Security Evaluation*. CCRA, 2012.

[4] Technical Committee CEN/TC 224. *Protection profiles for Secure signature creation device*. 2009.

[5] M. Baentsch, P. Buhler, T. Eirich, F. Horing, and M. Oestreicher. JavaCard-from hype to reality. *Concurrency, IEEE*, 7(4):36–43, 1999.

[6] GSMA and A.T. Kearney. *The Mobile Economy 2013*. GSMA, 2013.

[7] ISO. *ISO/IEC Std. ISO 7816-4: Organization, security and commands for interchange*. ISO, 2005.

[8] RSA Laboratories. *PKCS #1 V2.2: RSA Cryptography Standard*. RSA Laboratories, 2002.

[9] Agence nationale de la scurit des systmes dinformation. *Protection Profile Electronic Signature Creation Application*. PP-ACSE-CCv3.1, 2011.

[10] 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*, pages 6–6, 2007.

[11] Heiko Rossnagel. Mobile qualified electronic signatures and certification on demand. In SokratisK. Katsikas, Stefanos Gritzalis, and Javier Lpez, editors, *Public Key Infrastructure*, volume 3093 of *Lecture Notes in Computer Science*, pages 274–286. Springer Berlin Heidelberg, 2004.

[12] Antonio Ruiz-Martínez, Daniel Sánchez-Martínez, María Martínez-Montesinos, and Antonio F. Gómez-Skarmeta. A survey of electronic signature solutions in mobile devices. *J. Theor. Appl. Electron. Commer. Res.*, 2(3):94–109, December 2007.

[13] Antonio Ruiz-Martínez, Juan Sánchez-Montesinos, and Daniel Sánchez-Martínez. A mobile network operator-independent mobile signature service. *Journal of Network and Computer Applications*, 34(1):294 – 311, 2011.

[14] Mohammad Hasan Samadani, Mehdi Shajari, and Mehdi Ahaniha. A survey on mobile digital signature models. In *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, ICEC '10, pages 141–145, New York, NY, USA, 2010. ACM.