Application Level Smart Card Support through Networked Mobile Devices

Pierpaolo Baglietto[1], Francesco Moggia[1], Nicola Zingirian[2], and Massimo Maresca[2]

[1] DIST University of Genoa, via Opera Pia 13,
16145 Genova, Italy
{p.baglietto, fram}@dist.unige.it

[2] DEI, University of Padua, via Gradenigo, 16145 Padova, Italy
{mm, panico}@dei.unipd.it

Abstract. This paper addresses the problem of using networked mobile devices as providers of cryptographic functions. More specifically the paper describes a system developed to allow the usage of portable devices, such as PDAs and mobile phones, as remote smart card readers when connected into a TCP/IP network. This system is completely transparent to desktop applications. The digital signature technology, at its highest level of security, requires the use of smart cards and smart card readers not yet widely deployed. This requirement limits the mobility and may turn out to be an obstacle to the wide adoption of the digital signature technology. Our work aims precisely at facilitating the adoption of the smart card technology by means of PDAs and mobile phones.

1 Introduction

The main challenge for a successful and wide deployment of a new technology is the implementation of new user friendly services possibly characterized by a minimal impact on the an existing infrastructure.

The use of smart cards to execute cryptographic operations, such as the electronic signature, is presently possible only by means of smart card readers directly connected to the PC. The goal of our work has been to allow accessing the services of a microprocessor card located on a remote device without the introduction of a new API. On the contrary, at the application level we adopt standard interfaces that are already widely adopted.

In past projects, such as the CASTING project [1], a GSM phone has been proposed as a tool for authentication and for the storage of confidential information. In that case, private keys were securely stored on a SIM card that was able to communicate with a desktop PC using the SECTUS protocol, an ad hoc wireless protocol. In other works [2][3][4] has been proposed to make smart card accessible in a more general distributed environment.

On the contrary, the work described in this paper is focused on the development of a system which simply allows to remotely access the services provided by smart cards

A. Lioy and D. Mazzocchi (Eds.): CMS 2003, LNCS 2828, pp. 172–180, 2003.

through the standard PKCS#11 [5] or CSP (Cryptographic Service Provider) [6] application interfaces in TCP/IP networks. More specifically the architecture presented in this paper is based on a client-server model and it consists of two components. These components are both compliant with either the PKCS#11 or the CryptoAPI specifications. The client module establishes a secure TLS [7] channel with its corresponding server application and it sends calls to the PKCS#11 or CSP module to the server module. The server module receives the information about the requested cryptographic operations and executes them on the smart card inserted on a local smart card reader by means of usual PKCS#11 or CSP modules. The whole process is transparent to the user.

In section 2 and 3 we briefly present application level cryptographic standards we have used in the implementation of the system and the related work in the remote smart card application field. In section 4 we describe the usage scenario we considered during both the specification of the system architecture, described in section 5, and the development of a prototype system described in section 6. Finally, we give some concluding remarks in section 7.

2 Smart Card Application Programming Interfaces

The digital signature devices are used in the application by means of two standard called respectively CryptoAPI and PKCS#11, that define a high level application programming interface (API) layer. This API isolates an application from the details of the cryptographic device. The application does not have to change the interface for a different type of device or to run in a different environment; thus, the application is portable.

2.1 Cryptographic APIs

Cryptographic APIs (CryptoAPIs) are set of functions that allow applications to encrypt and digitally sign data in a flexible manner, granting privacy, secrecy and protection for the user's sensitive private-key data.

CryptoAPIs provide services that enable developers to add cryptography and certificate management functionality to their applications. Applications can use CryptoAPI functions ignoring the underlying implementation, in the same way that an application can use a graphic library putting no attention about the particular graphic hardware configuration.

All cryptographic operations, furthermore, are performed by independent modules known as cryptographic service providers (CSPs). Each CSP provides a different implementation of the CryptoAPI. Some CSPs interact directly with hardware components such as smartcards.

2.2 PKCS#11

This standard specifies an application programming interface (API), called "Cryptoki," for devices which hold cryptographic information and perform cryptographic functions. Cryptoki addresses the goals of technology independence (any kind of device) and resource sharing (multiple applications accessing multiple devices), presenting to applications a common, logical view of the device called "cryptographic token".

3 Scenario

The basic usage scenario is the signature of an e-mail, a document, or everything that have to be signed for authentication, integrity and non-repudiation purposes, without using a local smart card reader. In particular, let's consider the signature task of a document. If a user wants to digitally sign a document in a traditional way he has to use a smart card reader attached to his personal computer and the relative software. This is a restriction to spreading and utilization of digital signature, making it feasible only with full-configured and full-equipped PC. On the contrary, our system allows the user to perform a digital signature operation everywhere, by means of a mobile device. In everyday life few people take with them a PC smart card reader but everyone possesses a cellular phone or, surely in the next future, a more intelligent device such a PDA or a SmartPhone. Our idea is to use these mobile devices as remote smart card readers.

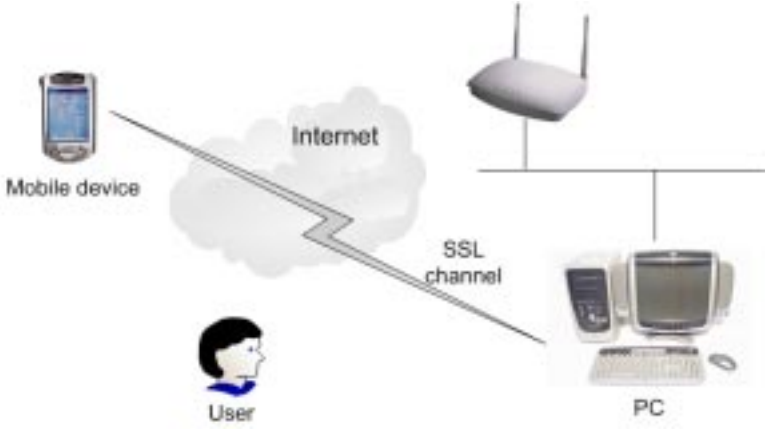

Fig. 1. The system basic elements

Another usage scenario may be the following. Let's consider the trust relationship usually existing between a company secretary and her sales manager in case of normal business transactions. The company secretary needs that some documents, not particularly significant for the enterprise, to be signed by the sales manager. The manager

trusts the secretary and puts his signature without spending time to read them accurately.

This is a widespread interaction-model and we want to reproduce it using digital signature through networked mobile devices.

The proposed schema is based on a client-server model that involves three main components: a client application on a desktop computer, a server application on a mobile device, and a secure channel between them. The mobile device is equipped with a smart card that contains the secret keys of the signer. We suppose that the mobile device has a PC/SC compliant smart card reader. The environment is a TCP/IP network, so that our device can be identified by an unique IP address. So this approach is valid both in a wireless LAN or in a WAN through the Internet.

4 Related Work in Remote Smart Card Applications

The problem of making the services offered by a smart card accessible from the Internet has been addressed in literature. In the WebSIM approach [5] a GSM SIM is transparently visible from the Internet via HTTP. This is possible by means of a small Web server placed on the SIM, whose unique function is to provide a Web-like interface to the SIM. All the services are implemented as server side scripts on the card and, so, they can be accessed via HTTP requests. The Web server is an applet on the top of a GSM SIM Toolkit platform that allows interacting with the SIM and provides a set of API for I/O operations. The problem of connecting the Web server to the Internet is solved by introducing a proxy server into the WebSIM architecture. The proxy has a public IP address which an Internet host can send its HTTP request to. The proxy encapsulates the HTTP request into a specially tagged SMS and sends it to the user's mobile phone. Once de-capsulated, the HTTP request is processed by the Web server and the response is sent back to the proxy using the same SMS tunneling mechanism. Finally, the proxy extracts HTTP response from the SMS and sends it to the Internet host. The WebSIM main purpose is to univocally authenticate and identify a user in the Web.

In the CASTING project [4] a mobile phone, containing the user certificate and the corresponding private key, is used for authentication purposes, allowing each user to securely access to his personal data, placed on a Web sever, from a pool of fixed PCs (terminals). The link between the mobile phone and the user terminal is secured by a wireless ad hoc protocol named SECTUS, while the long distance Internet connection is protected with SSL/TLS. The idea behind this work is to install a custom PKCS#11 Netscape token on the user terminal, that asks the mobile phone to execute on the card all necessary cryptographic functions involving the user private key in the SSL handshake with the Web server, and leaves to the terminal the task to perform the remaining operations. The goal of the SECTUS protocol is to establish an authentic channel between the terminal and the mobile phone by sharing a secret between them. The secret is necessary to compute the MAC of all messages that are sent across the wireless link, assuring that they are not been altered. The SECTUS protocol consists of the following steps: the mobile phone generates a temporary secret, called r, that is in-

serted by the user into the terminal. The mobile phone sends its certificate to the terminal that generates an authentication key, called s. This is encrypted with the user public key and sent back to the mobile phone together with a MAC calculated with r. The mobile phone verifies the MAC, decrypts s and, finally, sends the MAC of its certificate calculated with s to the terminal that, so, can verify it. In the TLS handshake with the Web server the client proves its identity by signing an hash value. The authenticity of the transmission of the hash and the signature is assured by MAC computed with the authentication key s.

A different approach, based on a distributed object model, is adopted by Chan and Tse [6], whose aim is to give a Corba interface to java card applications. The proposed architecture defines an OrbCard adaptor that works as a proxy gateway, translating the client Corba-specific requests into their APDU representations and sending them to the on-card applet. This applet generates the responses in the APDU format, which the adaptor maps into Corba responses and sends back to the client. The OrbCard adaptor gives a Corba interface to the on-card services, so that they appear as typical Corba objects.

5 Architecture

Our starting point is that the system must be transparent to the user, who can use standard application to perform a sign task. The digital signature devices are accessed in our application by means of two standard called respectively Crypto Service Provider (CSP) and PKCS#11. Both of them define a high level API layer that is independent from the specific cryptographic device.

For example Internet Explorer and Outlook Express uses CSPs (through CryptoAPI interface) to sign a document or an e-mail, while Netscape Navigator uses a PKCS#11 module. Our system relies on these standard interfaces, assuring a good level of portability. We have developed both PKCS#11 and CSP modules that expose all the essential functions for digital signature, that are a subset of whole CryptoAPI / PKCS#11 functions.

The idea is to transmit through a socket channel the necessary data to perform a sign task on the mobile device. The use of TCP/IP and Internet as, respectively, communication protocol and communication channel between the application and the smart card in place of RS232 and a serial cable entails the use of some technique to make the channel secure as if we had a local smart card reader. We must be sure about the identity of the mobile device which we are sending information to, we have to be sure about the identity of the person that executes the sign request and we want to assure the communication secrecy.

5.1 Smart Card Reader Authentication

Our solution for smart card reader authentication and communication secrecy is to use an SSL channel between the client and the server application, so that the PDA can authenticate itself presenting its certificate to the client and the communication chan-

nel between client and server can be encrypted for information hiding purposes. Smart card authentication is necessary to avoid that someone could impersonate the mobile device. In that case, the ill-intentioned could easily put himself between the user and the mobile device, read the whole communication and modify it in order to sign his document instead of the user.

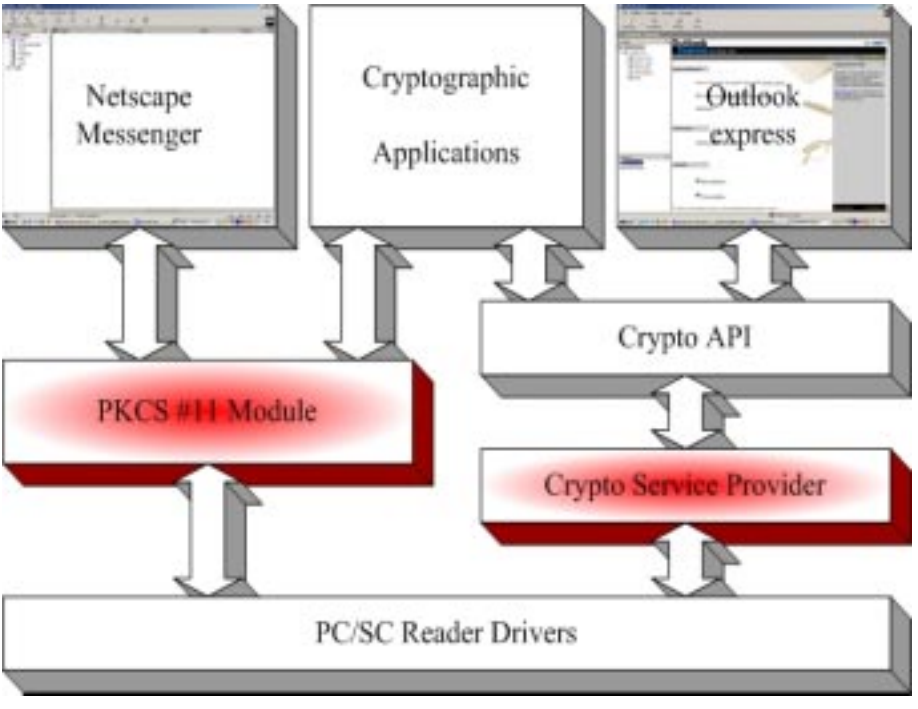

Fig. 2. The modules we have developed

5.2 User Authentication

For the user authentication, instead, we use an ad hoc solution. When the user wants to sign a document or e-mail, in fact, the application that performs the operation asks to the user to insert a PIN. The PIN, generally, protects sensible data on the card and allows only an authenticated access to the on board functions. In our application, the PIN takes the meaning of a user identification code. The user identification code, serves exclusively to allow the user to be sure that the signature request, appearing on his PDA, is the request started on the personal computer. Once the user has verified that the request is correct by means of the user identification code on the PDA, he must enter the PIN code to perform the signature. Inserting PIN code on PDA is more secure than inserting PIN code on a personal computer. Altering a software or the operative system on a PDA, in fact, is more difficult than modify them on a PC, simply because the PDA is less accessible than a personal computer. On the personal

computer, in fact, a ill-intentioned could install a software that, while the user inserts a PIN, captures the keys pressed on the keyboard and rescues the PIN.

The user identification code, for example, is necessary to avoid the man in the middle attack. If we send a request to the mobile device to sign a document without the user identification code, an hacker could block our request and take our place and we wouldn't have the necessary means to reveal this attack. So the user, interpreting the request on his PDA as trusted, would insert the PIN code to finalize the signature task, but the signed document owner would be the ill-intentioned and not the user.

Fig. 3. System architecture

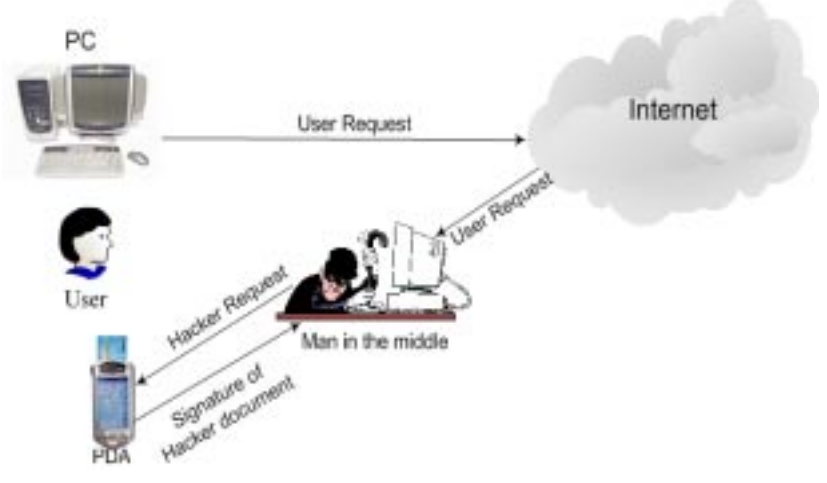

Fig. 4. A possible attack to the system

6 Implementation

At the implementation level, we have developed a system based on a client-server paradigm. The client side application includes a PKCS#11 module, while the server side application a SSL server. Moreover, the client side application communicates with the server side application by means of an ad hoc communication protocol.

6.1 Ad hoc Communication Protocol

Our ad hoc communication protocol consists of a simple mechanism of serialization. In particular, we have implemented a common base class that holds a function identifier and a member variable containing the server side error code generated by the invoked function. Each PKCS#11/CSP function is mapped in a class, derived from the base, that contains specific parameters.

The necessary steps to invoke a function are the following:

- Instantiate the right class corresponding to the function (Function id is automatically assigned in the constructor of the class)
- Insert the values of the function parameters in the instantiated class
- Send the class instance to the server
- Receive the server response
- Parse the response and return to the client application the right values

6.2 Client Side Application

A PKCS #11 module is the main part of the client side application. For simplicity, we have not implemented all the functions defined by the standard but only those necessary for being able to perform digital signature operations, and only some of them are executed remotely. The functions locally performed refer to information about the PKCS#11 module, to functionalities supported by the module and to all that is not used directly by the smart card. The client module opens an SSL channel with the mobile device when the application requests the execution of the first remote function.

For being able to verify if the mobile device to which we are sending confidential information is trusted device, the PC screen visualizes the certificate sent by mobile device in the SSL handshake and asks to the user if the content is correct. At this point, the SSL channel is established and all remote functions can be correctly invoked by means of the ad hoc protocol above-mentioned. At the end, the client closes the SSL session when the digital signature task is finished or when a timeout expires.

6.3 Server Side Application

The server side application waits for the incoming client request and try to establish a secure SSL channel with it. The server identity is granted by the SSL certificate and by the corresponding private key generated with OpenSSL.

If the SSL handshake has succeeded, the server is ready to execute the function requested by the client. The server performs all operations, parsing the client request and identifying the function called. This is accomplished by checking the first byte received that specifies the function identifier invoked by the client application. This process continues until the client requests to execute the C_Login function. At this point the server shows on the display of the PDA the one-time user identification code, so that the user can verify it and, if it's correct, insert the PIN code to perform digital signature.

For the management of the SSL channel we have used the OpenSSL libraries compiled for Windows CE.

7 Conclusions and Future Works

We have presented an architecture model based on well-defined standards that allows signing of a document remotely and in a transparent way.

We have made a prototype that uses a handheld PC with a wireless LAN card, and we are working to extend this work to cellular phones. SIM Application Toolkit provides the SIM card with an embedded java virtual machine on which it is possible to run applets. Our idea is to use a proxy that terminates the SSL connection of the client and works as a SMS gateway for the mobile device.

Moreover, a direct porting of our application on mobile phones that adopt Windows CE as operating environment is straightforward.

References

1. Michael Rohs, Harald Vogt., Smart Card Applications and Mobility in a World of Short Distance Communication, CASTING Project Tech Report., ETH Zurich, January 2001.
2. Roger Kehr, Michael Rohs, Harald Vogt, Mobile Code as an Enabling Technology for Service-oriented Smartcard Middleware, Proc. 2nd Int. Symposium on Distributed Objects and Applications DOA'2000, Antwerp, Belgium, IEEE Computer Society, pp. 119–130, September 2000.
3. Scott Guthery, Roger Kehr, Joachim Posegga, and Harald Vogt. GSM SIMs as Web Servers. In 7th International Conference on Intelligence in Services and Networks IS&N, Athens, Greece, Feb. 2000
4. A. Chan, F. Tse, J. Cao and H. Leong, Enabling distributed Corba Access to smart card applications, Ieee Internet Computing, pp. 27–36, Vol. 6 N. 3, May 2002.
5. RSA Laboratories. PKCS#11 v2.11: Cryptographic Token Interface Standard, November 2001.
6. Robert Coleridge, The Cryptography API, or How to Keep a Secret, Microsoft Developer Network Technology Group., August 1996.
7. C. Allen and T. Dierks. The TLS Protocol, Version 1.0. Internet RFC 2246, January 1999.