Smart Card Authentication for Mobile Devices

Wayne Jansen
National Institute of Standards
and Technology
jansen@nist.gov

Serban Gavrila
VDG, Inc.
gavrila@nist.gov

Clément Séveillac
Amadeus
clem@via.ecp.fr

Abstract: *While mobile handheld devices provide productivity benefits, they also pose new risks. User authentication is the best safeguard against the risk of unauthorized use and access to a device's contents. This paper describes two novel types of smart card with unconventional form factors, designed to take advantage of common interfaces built into many current handheld devices.*

Keywords: *Mobile Devices, Authentication, Smart Cards*

Introduction

With the trend toward a highly mobile workforce, the use of handheld devices such as Personal Digital Assistants (PDAs) is growing at an ever-increasing rate. These devices are moderately inexpensive productivity tools that have become a necessity for government and industry. While such devices have their limitations, they are nonetheless extremely useful in managing appointments and contact information, reviewing documents and spreadsheets, corresponding via electronic mail and instant messaging, delivering presentations, accessing remote corporate data, and handling voice calls. Over time, significant amounts of sensitive corporate information can accumulate on them and automatic access to corporate resources via wireless and wired communications can be enabled, making those resources a potential target of an attack.

One of the most serious security threats to any computing device is unauthorized use. User authentication is the first line of defense against this threat. Smart card authentication is perhaps the best-known example of a proof by possession mechanism. Other classes of authentication mechanisms include proof by knowledge (e.g., passwords) and proof by property (e.g., fingerprints). Smart cards are credit card-size, plastic cards that hold an embedded computer chip containing an operating system, programs, and data. They can be imprinted with a photo, a magnetic strip, or other information, for dual use as a physical identification badge (Polemi, 1997). Smart cards can help to improve the security of a device and also provide additional security services. Many organizational security infrastructures incorporate smart cards. However, standard size smart cards are generally not amenable to handheld devices because of the comparatively large size of the card, the need for a suitable card reader, and the difficulty and cumbersomeness of interfacing a reader to the device.

This paper describes two types of smart card that use standard interfaces supported by most handheld devices, in lieu of those interfaces favored by most smart card readers. The paper explains how these forms of smart card can be applied to authenticate users on handheld devices and provides details of the solutions' design and implementation.

Background

Smart cards are designed to protect the information they contain. Tamper resistance techniques are used to protect the contents of the chip embedded on the card. The computer chip requires a smart card reader to obtain power and a clock signal and to communicate with the computing platform. Once contact is made with the reader, a smart card uses a serial interface to communicate with software running on the computing platform. Java Card is currently one of the more popular operating systems for smart cards.[1] Data units received by the smart card are processed by Java applications installed on the card. The Java Card runtime facilitates the development and deployment of Java applications to support authentication and other security-related applications, such as those involving electronic commerce.

For a smart card to allow access, it typically requires the user to enter a PIN (Personal Identity Number) first, to verify that the individual in possession of the card is the person to whom the card was issued. Incorrect PINs

[1] Certain commercial products and trade names are identified in this paper to illustrate technical concepts. However, it does not imply a recommendation or an endorsement by NIST

keep the card from functioning and eventually cause it to lock. Once the PIN is successfully entered, a dialogue between the handheld device and smart card occurs, by which the device confirms that the card and the credentials of the user on the card are valid.

The capabilities and form factor of standard credit card-size smart cards are compatible with some handheld devices, provided that a reader can somehow interface with the device and a compatible driver is available for the platform's operating system. For example, several manufacturers produce smart card readers as hardware modules that fit into a type II PCMCIA Card slot. These readers accept standard-size smart cards, obtained separately. A platform, such as an iPAQ 5550 PDA, whose expansion options include both single and double PCMCIA slot expansion sleeves, can readily accept such readers and operate them, once a suitable driver is installed. More elegant solutions also exist such as the Blue Jacket (Axcess Mobile Communications, 2005), which incorporates a smart card reader within the expansion sleeve and can support an optional Bluetooth communications and a type II compact flash interface. Either solution, however, is limited to certain types of PDAs and adds considerable bulk to the device.

Smart cards come in other form factors. A popular format emerging for smart cards is a USB key fob. This chewing gum pack-size hardware component has a printed circuit board with a processor and memory encased within a plastic housing containing a USB connector at one end. Many manufacturers produce USB devices that function identically to smart cards and, since they interface through a USB port, eliminate the need for a reader. Currently, however, very few handheld devices support host USB ports, which are needed to interface to these peripherals. One constraining factor is that the handheld device would need to draw on its battery to power any peripherals plugged into the USB port.

Another alternative is the iButton, a 16mm computer chip contained in a stainless steel shell, able to be mounted in jewelry such as a ring (Maxim/Dallas Semiconductor Corp, 2002). Capabilities of these button-size devices range from a simple memory token to a microprocessor and arithmetic accelerator able to support a Java Card-compliant Virtual Machine. However, a button receptacle incorporated into the device or easily added (e.g., via a compact flash card) is needed to foster their use with PDA devices. USB holders are also available for iButtons, but would require a host USB port.

The authentication mechanisms described in this paper rely on packaging smart card functionality in a form factor that is compact, unencumbering, and compatible with the capabilities possessed by handheld devices. The mechanisms were designed to authenticate the user via an issued smart card security token. Once the user succeeds in authenticating, the token is closely monitored to confirm its presence throughout the user's interaction with the device. Removing the token from the device or turning it off, automatically terminates access to the device. The two smart card authentication tokens used are distinguished as either contact or contactless. These compact tokens require only that participating handheld devices respectively support a standard memory card interface or a common wireless interface for Personal Area Network (PAN) communications.

The authentication mechanisms were implemented in C and C++ on an iPAQ PDA, running the Familiar distribution of the Linux operating system from handhelds.org and the Open Palmtop Integrated Environment (OPIE). The Familiar distribution was modified with MAF, a framework for multimode authentication (Jansen et al., 2003b). The framework includes a policy enforcement engine that governs the behavior of the device (Jansen et al., 2003a) and a facility to add new authentication mechanism modules and have them execute in a prescribed order. MAF authentication mechanisms consist of two parts: an authentication handler, which embodies the procedure that performs the actual authentication, and a user interface, which performs all necessary interactions with the user. The authentication mechanisms described in this paper are referred to as the smart multimedia card and Bluetooth smart card mechanisms, and were implemented specifically for MAF. MAF functionality was used to protect authentication components and any security-related files stored on the handheld device.

Smart Multimedia Card Authentication

The Smart Multimedia Card (SMC) authentication mechanism relies on a smart card chip packaged in a multimedia card format. The postage stamp-size card houses a MultiMedia Card (MMC) controller, smart card, and additional flash memory. Many PDAs and other handheld devices support an MMC card slot, making such cards a viable means to provide smart card functionality. The MultiMediaCard Association has recently drafted standard specifications for secure multimedia cards.

A pre-production Smart MMC produced by Renesas called the X-Mobile Card (XMC) observes the draft standard and was used for the prototype implementation. The tamper resistant hardware module complies with Java Card 2.2.1, Global Platform 2.1, FIPS 140-2 (currently under evaluation), and other standards. To use the XMC, a Linux device driver was developed, which is now available at an open source site for Linux smart card software.

Operation

The SMC mechanism relies on the user to possess an issued smart card security token to satisfy authentication. The handler software, which runs in user space on the handheld device, monitors card insertion and removal, and controls all the necessary steps regarding the authentication mechanism. The aim of the mechanism is twofold: to authenticate the user to the handheld device, and to ensure that the smart card with which the user authenticated remains in force. To carry out its function, the SMC handler communicates with the modified Linux kernel, the OPIE plug-in components that make up the user interface, and a special purpose "Enroller" applet on the smart card.

In its initial communications with the kernel, the handler indicates that it is a polling handler and specifies the polling interval for callback. It then receives orders from the kernel either to poll the smart card or to perform authentication. For the former, detected state changes in the card from the previous poll cause the handler to request a callback with an order to perform authentication. For the latter, the handler replies to the kernel giving the result of the authentication.

In communications with the OPIE plug-in, the handler tells the user interface to display certain informative messages, when needed, and to accept PIN entry from the user.

In communications with the XMC smart card and the Java Card "Enroller" applet, the handler uses the PC/SC Lite software protocol stack. PC/SC Lite is an open source software stack for Linux based on the PC/SC (Personal Computer/Smart Card) specification (PC/SC Workgroup, 2005), a popular general-purpose architecture for smart cards. The communication with the card consists of exchanging APDUs (application protocol data units) with the on-card applet over a secure channel. The "Enroller" applet validates the PIN supplied by the user via the handler and verifies the user's claimed identity using the FIPS 196 challenge-response protocol. The applet and PIN are placed on the smart card along with the user's public key credentials during card personalization.

Implementation

The SMC handler operates in two modes, as directed by the kernel: a polling handler, periodically checking the status of the smart card during a polling moment, and an authenticator, performing an authentication with the smart card.

Performing authentication is accomplished by the handler obtaining the PIN from the user and using appropriate APDUs to establish an authentication session with the smart card, create a secure channel to the card, issue a challenge, and verify the response. The challenge-response protocol used is compliant with FIPS 196 (National Institute of Standards and Technology, 1997). FIPS 196 is designed with measures to conceal the base secret used and avoid replay. Figure 1 illustrates the scheme, omitting the requisite PIN satisfaction step that occurs. The upper part of the diagram shows the initial exchange used to enroll a smart card token at right with the handheld device at left, while the remainder shows the exchanges used to verify the claimed identity following FIPS 196 procedures:

- The device, acting as the verifier, generates a random challenge "B" and passes it to the smart card for signing with the private key associated with the enrolled identity certificate;
- The smart card, acting as the claimant, generates a random value "A," signs A||B with the private key on the card ('||' denotes concatenation), and returns A and the signature to the device;
- The device retrieves the enrolled identity certificate, verifies it, then verifies the card's signature over A||B using the public key in the certificate;
- If everything successfully verifies, authentication succeeds; otherwise, the authentication attempt fails.

In signing the challenge and verifying the signature, the handler uses OpenSSL v0.9.7 APIs that comply with the PKCS #1 standard, while the applet uses an available on-card function.

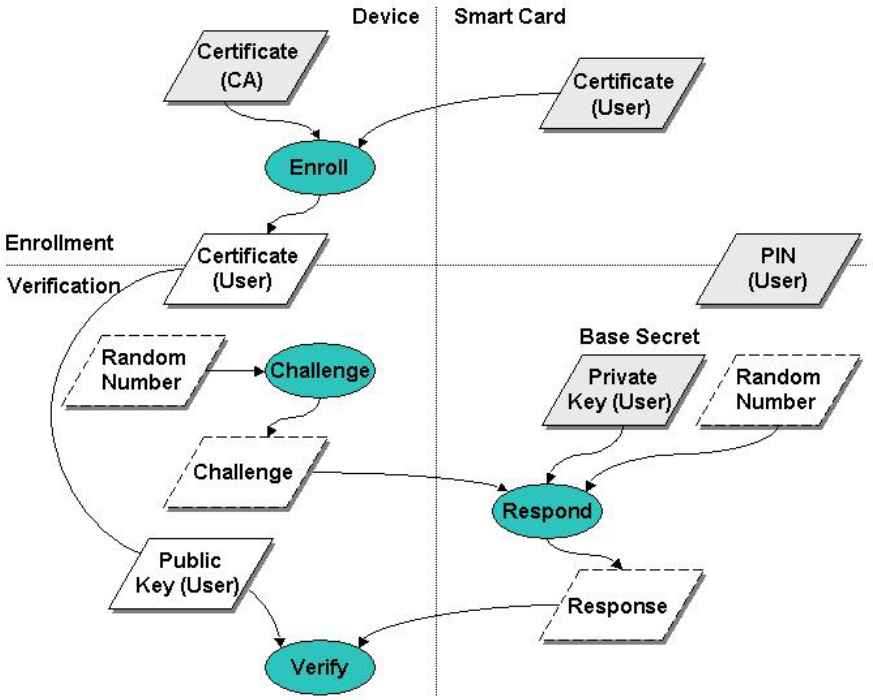

Figure 1: Challenge-Response Exchange

Performing polling operations is slightly more involved. The handler begins by obtaining the card's state and weighing several factors: whether a previous authentication session exists, the card's current and previous states, and whether the card was reinserted (and maybe replaced) between the previous and the current polling moments. The possible cases that can occur during a poll and the action taken by the handler in each case appear in Table 1.

Table 1: Decision Matrix

Previous Authentication Session Exists	Card's Current State	Card's Previous State	Card Reinserted	Action
Yes	Present	Present	Yes	Tell kernel the authentication failed
			No	Do nothing
		Absent	N/A	Tell kernel to attempt authentication
	Absent	Present	N/A	Tell kernel the authentication failed
		Absent	N/A	Do nothing
No	Present	Present	No	Do nothing
		Absent	N/A	Tell kernel to attempt authentication
	Absent	Present	N/A	Tell kernel the authentication failed
		Absent	N/A	Do nothing

The most interesting case in the table is the first entry, where the card is present, but was removed and inserted (and possibly replaced) since the last polling moment. To force reauthentication to occur in this situation, the handler changes the current card state to "absent," while the previous state remains "present." It then tells the kernel that the authentication failed. In the next iteration, the previous card state and current card state are updated, resulting respectively in "absent" and "present" settings. The handler then tells the kernel that conditions exist to attempt authentication, so that it will be called back subsequently to perform an authentication operation.

The Enroller applet works in conjunction with the handler. It is a Java Card applet designed for use in the XMC and other Java Card-compliant smart cards. The name of the applet is a bit of a misnomer, since the applet participates both in the personalization of the card and the authentication process. The applet conforms to Java Card 2.1.2 specifications and supports secure channel communications with a host application as specified in Global Platform 2.1. The applet supports a set of APDUs that provides the following functionality:

- Generates an RSA private/public key pair
- Stores an RSA public or private key
- Retrieves the RSA public key
- Stores or retrieves an X.509 certificate
- Sets or verifies the user PIN
- Signs a host challenge with the private RSA key
- Supports the creation of a secure communication channel with a host application

The SMC token requires correct PINs to unlock its functions. Several bad PIN entry attempts lock the card. A Global Platform Secure Channel is used to protect the PIN and any other information sent to the card. The private key of the user and the user's PIN established during personalization cannot be exported from the token.

Bluetooth Smart Card (BSC) Authentication

The Bluetooth Smart Card (BSC) authentication mechanism, as with the SMC, relies on a smart card chip packaged together with other components in a compact-size form factor (e.g., a key fob). However, rather than bringing a smart card into physical contact with a handheld device, a wireless interface is used. Bluetooth Smart Card authentication can provide the security of smart card-based authentication to a device with the following advantages:

- No need exists for a specialized smart card reader
- The token can be small enough to fit comfortably on a person
- It can work within a few meters of the device, without a direct line of sight
- It does not draw power from the handheld device
- It can be discrete (i.e., non-discoverable by third parties)

A BSC token houses a Bluetooth radio, smart card, processor and memory, and battery. The token could also include a display and a keypad to allow PIN entry and other management functions. Since many PDAs and other handheld devices support a Bluetooth radio, such tokens are a viable means to incorporate smart card functionality. A BSC token could also be used with Bluetooth-enabled workstations. The mechanism is also amenable to other types of low-power Personal Area Network (PAN) communications.

Operation

The Bluetooth Smart Card has many similarities with the SMC insofar as both solutions depend on the functionality of a Java Card-compliant smart card chip, use the same challenge-response protocol for user authentication, and are implemented to execute within the MAF environment. Therefore, wherever possible, components of the SMC were reused for BSC.

The main difference from SMC is that communications between the device and token takes place using a Bluetooth channel rather than an MMC bus. Another difference is that PIN entry may occur at the BSC token rather than at the handheld device.

In developing the solution, an effective way was found to split the PC/SC functionality between the handheld device and the BSC token to allow Bluetooth communications, yet have minimal impact on any smart card application. Figure 2 illustrates how the PC/SC Lite components used previously in the SMC were divided and allocated between the device and token. Three main PC/SC Lite software components support a smart card application: the service provider, a resource manager, and a driver for the smart card reader. The software application, such as the handler, normally communicates with the driver indirectly via the service provider, which in turn uses the standardized PC/SC interface to the resource manager. Similarly, the resource manager uses a standard interface component called the IFD handler to communicate with the driver. The driver is closely tied to characteristics of the smart card reader, while the service provider is closely tied to the characteristics of the smart card, allowing an application to use any smart card with any reader, provided that the respective service provider and driver software are available.

The IFD handler, shown as a small box appearing between the resource manager and driver, was the key to adapting Bluetooth communications. The IFD Handler provides a standard interface to the resource manager on one side and maps the functions over the Bluetooth channel to the other, permitting the BSC token to implement an entire IFD subsystem independently of the other PC/SC Lite components. APDUs are sent over the L2CAP Bluetooth layer using a socket interface.

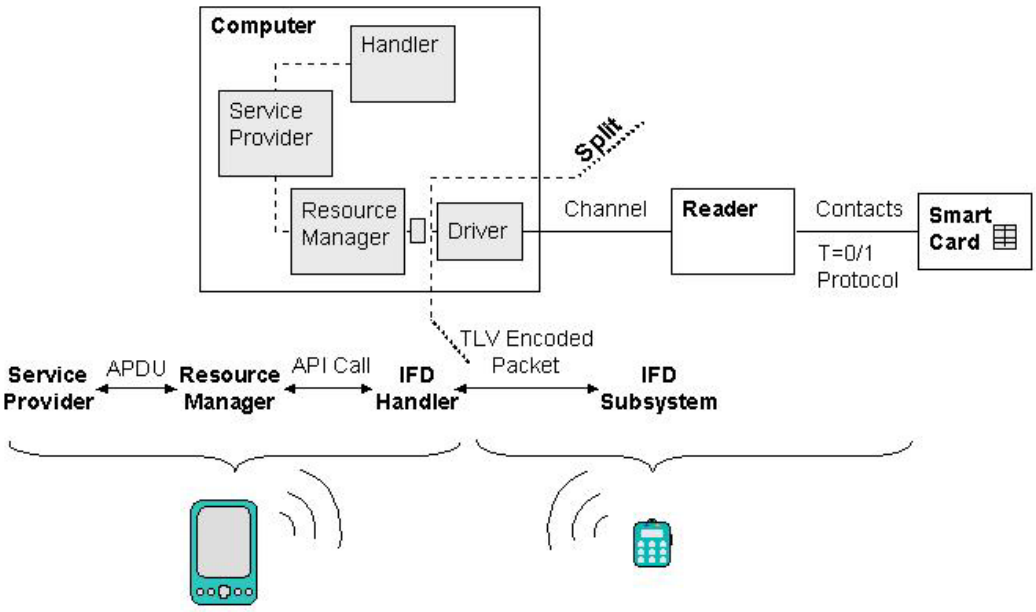

Figure 2: Reification of the PC/SC Lite Architecture

The arrangement allows the Bluetooth Smart Card solution to work with any type of Java Card-compliant smart card recognized by the PC/SC Lite framework. For simplicity, however, X-Mobile Cards were used in the token prototype.

Implementation

The SMC handler on the device side and the SMC applet on the card side were reused in the BSC implementation. Only a small addition to the SMC handler had to be made to enable remote PIN verification. Remote PIN verification is a request from the handler to the IFD subsystem on the token to inquire about its capabilities to accept PIN entries. If the capability is not present, the handler prompts the user for this information on the device, as done for the SMC. If the capability is present, the handler bypasses those steps and instead relies on the BSC token to obtain the PIN from the user. This change works equally well for the SMC and BSC tokens, allowing the same updated handler to be used for both authentication mechanisms. Thus, the part of the BSC implementation that distinguishes it from the SMC is the IFD handler developed to communicate over Bluetooth to the IFD subsystem on the token.

The IFD handler is software executing on the handheld device that implements a standard, hardware-independent, and I/O channel-independent interface into the IFD subsystem. The IFD handler has to map the standard interface it offers onto the Interface Device functionality (e.g., a smart card reader), to allow data to be exchanged with a smart card. Communicating with a device driver often suffices. However, for the BSC token, which supports a complete IFD subsystem independently from the host, the IFD handler merely maps the identical interface to the smart card functionality over a Bluetooth channel. The new IFD handler for the Bluetooth Smart Card looks like a normal Smart Card reader driver interface for the PC/SC Lite stack, but operates as a proxy for another IFD handler present on the BSC token reachable via Bluetooth.

The protocol used to forward the IFD functions and arguments to the BSC token, and to receive the corresponding responses, is a custom Tag/Length/Value-based serialization protocol. Any forward message starts with a byte representing the IFD function being transmitted. From this identifier, both the client and the server know the number of the arguments, their type, and their order for each defined function. Following the identifier byte, each argument comes in proper sequence, encoded as follows:

- If the argument is fixed length, it is directly appended.
- If the argument is variable length, it is preceded by its length and then appended.

The return messages are even simpler, since most of them are fixed length. Any arguments returned are serialized using the same encoding described above.

Since the device and token can be paired, a trusted connection exists between them during operation. Each unit automatically accepts communication from the other in encrypted form, bypassing the discovery and

authentication process that normally occurs during Bluetooth interactions. Two Bluetooth devices should be paired to one another during the initial enrollment phase, when the token is first registered with the device. Bluetooth pairing establishes a shared symmetric key on both units, used to authenticate and encrypt exchanged L2CAP packets respectively via a MAC and stream cipher.

One level higher, pre-established symmetric keys (i.e., one for MAC, and one for TripleDES encryption) are used to protect transmitted APDUs using the Global Platform Secure Channel (GPSC) Protocol 01 format. Under this protocol, only the data portion of an APDU is encrypted and encryption is applied unidirectionally from the device to the smart card token. If needed, the host and card applications could be modified to encrypt all or parts of the messages returned from the smart card, but this is not currently part of the GPSC Protocol. However, apart from the PIN transmission, none of the other information exchanged (i.e., the FIPS 196 challenge and response) affects the overall mechanism, if exposed.

Figure 3 illustrates the two levels of confidentiality protection afforded by GPSC and Bluetooth. The shaded portions represent the encrypted parts of the dialogue.

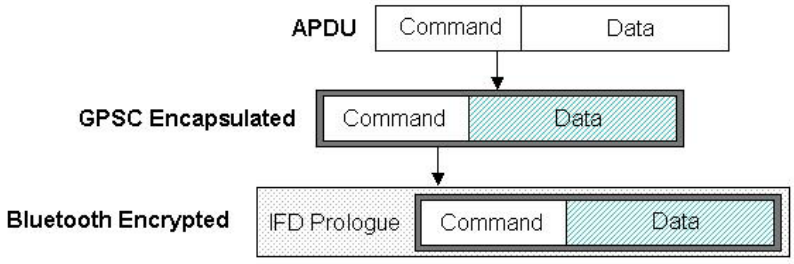

Figure 3: Communication Protection

The token was implemented as an OPIE application running on an iPAQ PDA. A virtual token appears on the PDA's display, emulating the real token. Figure 4 shows a screen shot of a token in a key fob form factor. A server module on the token handles key functions, including all of the smart card and Bluetooth socket interactions.

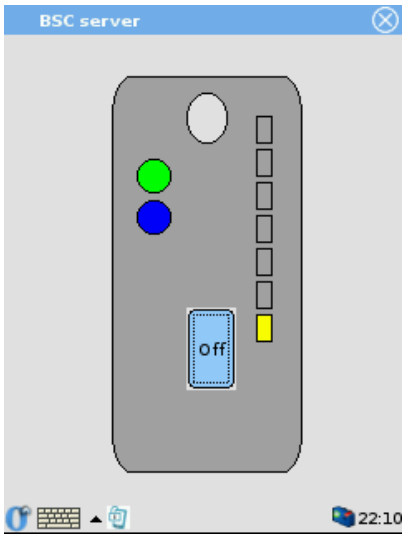

Figure 4: Bluetooth Smart Card Token

A user interacts with the token through the On/Off button. By default, neither Bluetooth nor the server module is launched until the On/Off button is pressed, turning the token on. Bluetooth is started first, which is indicated by the lower circular LED at left becoming blue. The server module then follows, which is indicated by the top circular LED at left becoming green. Once the server is operational, it represents its activity through the stacked LEDs at right.

Another variant of the token allows the user to enter the smart card PIN directly at the token, avoiding any Bluetooth eavesdropping attack. Figure 5 shows the virtual token displayed on the PDA screen. The same common functions as the previous variant are supported. However, adding a numeric pad with control buttons

and an alphanumeric display screen allows PIN entry, as well as possible passkey entry for Bluetooth pairing. The same server module is used by both tokens (i.e., with and without a PIN keypad). A flag (--nopin) tells the server if it should run in remote PIN verification mode or not.

Figure 5: Token with PIN Entry

The application for the PIN entry token receives a notification from the server when a PIN is needed, and reacts by displaying a prompt on its screen and activating the keypad. When the user enters a PIN and presses OK or Cancel, the application sends back a command to the server to inform it either that the PIN was entered (and its value) or that entry was canceled.

Conclusions

While mobile devices provide productivity benefits, they also pose new risks. This paper demonstrates how smart card authentication can be implemented to reduce them. The approach provides users a simpler and less cumbersome way to interface smart card functionality that with conventional types of smart cards.

References

Axcess Mobile Communications (2005) Blue Jacket Product Information, URL: http://www.axcess-mobile.com/products/BlueJacketFlyer.pdf.

Jansen, W., Karygiannis, T., Iorga, M., Gavrila, S., Korolev, V. (2003a) Security Policy Management for Handheld Devices, The 2003 International Conference on Security and Management, June 2003.

Jansen, W., Korolev, V., Gavrila, S., Heute, T., Séveillac, C., (2003b) A Framework for Multi-Mode Authentication: Overview and Implementation Guide, NIST Interagency Report 7046, August 2003.

Maxim/Dallas Semiconductor Corp (2005) What is an iButton?, URL: http://www.ibutton.com/ibuttons/index.html.

National Institute of Standards and Technology (1997) Entity Authentication Using Public Key Cryptography, Federal Information Processing Standards Publication (FIPS PUB) 196, U.S. Department of Commerce, February 18, 1997.

PC/SC Workgroup (2005) Interoperability Specification for ICCs and Personal Computer Systems, Part 1 - Introduction and Architecture Overview, Revision 2.01.00, June 2005, URL: http://www.pcscworkgroup.com/specifications/files/pcsc1_v2.01.0.pdf.

Polemi, D. (1997) Biometric Techniques: Review and Evaluation of Biometric Techniques for Identification and Authentication, Institute of Communication and Computer Systems, National Technical University of Athens, April 1997, URL: ftp://ftp.cordis.lu/pub/infosec/docs/biomet.doc.