Smart Cards for mobile devices

Wayne Jansen*

National Institute of Standards and Technology,
Gaithersburg, Maryland, USA
E-mail: jansen@nist.gov
*Corresponding author

Serban Gavrila

VDG Inc., Chevy Chase, Maryland, USA
E-mail: gavrila@nist.gov

Clément Séveillac

Amadeus, Sophia Antipolis, France
E-mail: 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 unauthorised use and access to a device's contents. This paper describes two novel types of Smart Card (SC) with unconventional form factors, designed to take advantage of common interfaces built into many current handheld devices.

Keywords: mobile devices; authentication; Smart Cards; SCs.

Reference to this paper should be made as follows: Jansen, W., Gavrila, S. and Séveillac, C. (2007) 'Smart Card for mobile devices', *Int. J. Information and Computer Security*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Wayne Jansen is a Principal Computer Scientist at the National Institute of Standards and Technology (NIST), Computer Security Division in Gaithersburg, Maryland. His research interests lie mainly in communications, distributed applications, and security. Currently, he heads the Mobile Security Project, focused on securing mobile software and handheld devices.

Serban Gavrila received his Bachelor's and Master's Degree in Computer Science from the University of Bucharest, Romania. He joined VDG Inc. of Chevy Chase, Maryland, USA in 1995 as a senior research scientist. He has been involved in compiler construction and development of access control management systems.

Clément Séveillac has been passionate about Information Technology and Information Security even before entering Ecole Centrale Paris, a Prestigious French Engineering School. During his studies there, he had the opportunity to work for various companies, including SchlumbergerSema (now Axalto). Just after graduating, he went to NIST for two years in the Mobile Security Project. He is now developing technologies for the travel industry in the South of France.

1 Introduction

With the continuing trend towards a highly mobile workforce, the use of mobile handheld devices such as Personal Digital Assistants (PDAs) and smart phones is growing at an ever-increasing rate. These devices are moderately inexpensive productivity tools that have become a necessity for government and industry. While mobile devices have their limitations, they are nevertheless 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, users can accumulate significant amounts of sensitive corporate information on them and enable automatic access to corporate resources via wireless and wired communications, making that information a potential target for attack.

One of the most serious security threats to any computing device is unauthorised use. This threat is especially acute for mobile handheld devices, since their small size allows them to be easily misplaced, lost, or stolen. User authentication, the ability to differentiate legitimate users from illegitimate ones, is the first line of defence against unauthorised use, and provides a foundation for access control and confidentiality.

Smart Card (SC) 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). SCs are credit card-size, plastic cards that hold an embedded computer chip containing an operating system, programs, and data (Husemann, 1999; Vedder, 1992). SCs can improve the security of a device by providing an independent tamper-resistant processing environment for use in authentication and other security services (Turban and McElroy, 1998). Many organisational security infrastructures incorporate SCs. However, standard-size SCs 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 SC that use standard interfaces supported by most handheld devices, in lieu of those interfaces favoured by most SC readers. The paper explains how these novel forms of SC can be applied to authenticate users on handheld devices, and provides details of the solutions' design and implementation.

2 Background

SCs 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 SC reader to obtain power and a clock signal and to communicate with the computing platform. Once contact is made with the reader, a SC 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 SCs.[1] Data units received by the SC 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 SC to allow access, it typically requires the user to enter a Personal Identity Number (PIN) first, to verify that the individual in possession of the card is the person to whom the card was issued. Incorrect PINs keep the card from functioning and eventually

cause it to lock. Once the PIN is successfully entered, a dialogue between the computing platform and SC occurs, by which the platform 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 SCs 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 SC readers as hardware modules that fit into a Type II PCMCIA Card slot. These readers accept standard-size SCs, 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 SC 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.

SCs come in other form factors. A popular format emerging for SCs is a Universal Serial Bus (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 SCs 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 16 mm computer chip contained in a stainless steel shell, able to be mounted in jewellery 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 PDAs and other handheld 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 SC 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 SC 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 SC authentication tokens used are distinguished from one another as being 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 behaviour 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 (SMC) and Bluetooth Smart Card (BSC) 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.

3 Smart Multimedia Card (SMC) authentication

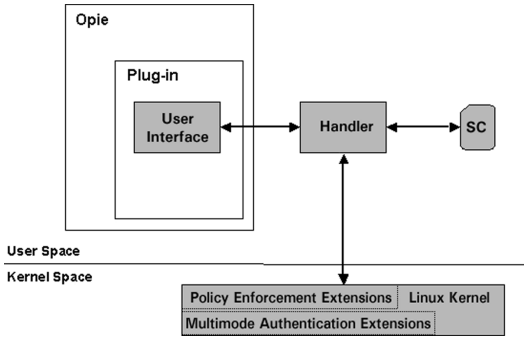
The SMC authentication mechanism relies on a SC chip packaged in a multimedia card format. The postage stamp-size card houses a MultiMedia Card (MMC) controller, SC, and additional flash memory. Many PDAs and other handheld devices support an MMC card slot, making such cards a viable means to provide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

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 SC software.

3.1 Operation

The SMC mechanism relies on the user to possess an issued SC 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 SC 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 component that forms the user interface, and a special purpose ‘Enroller’ applet on the SC. Figure 1 illustrates the situation.

Figure 1 Authentication Handler Communications



In its initial communications with the kernel, the handler indicates that it is a special type of handler, operating in polling mode, and specifies the polling interval for call back. It then receives orders from the kernel either to poll the SC or to perform authentication. For the former, detected state changes in the card from the previous poll cause the handler to request a call back with an order to perform authentication. For the latter, the handler replies to the kernel with the verdict 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 SC 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 Personal Computer/Smart Card (PC/SC) specification (PC/SC Workgroup, 2005), a popular general-purpose architecture for SCs. The communication with the card consists of exchanging Application Protocol Data Units (APDUs) with the on-card applet over a Global Platform 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 SC along with the user's public key credentials during card personalisation.

3.2 Implementation

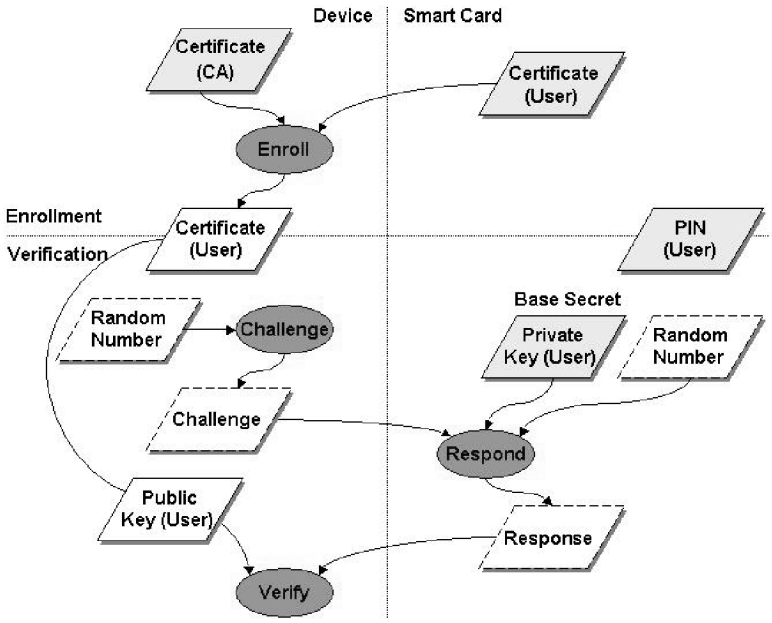
The SMC handler operates in two modes, as directed by the kernel: a polling mode, periodically checking the status of the SC during a polling moment, and an authenticator mode, performing an authentication with the SC.

Performing authentication is accomplished by the handler obtaining the PIN from the user and issuing the appropriate APDUs to establish an authentication session with the SC, 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 2 illustrates the scheme, omitting the requisite PIN satisfaction step that occurs.

The upper part of the diagram shows the initial exchange used to enrol a SC 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 SC for signing with the private key associated with the enrolled identity certificate
- the SC, 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.

Figure 2 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 SCs (Ortiz, 2003). The name of the applet is a bit of a misnomer, since the applet participates both in the personalisation 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.

3.3 Safeguards

The fundamental threat to user authentication is an attacker impersonating a user and gaining control of the device and its contents. Both the device and SC and the communications between them are potential targets.

Security measures that apply to the device rely on MAF and on the security of the underlying operating system. MAF policy rules regarding device resources are enforced at the kernel level, independently of access permissions assigned to users. The SMC handler is protected from substitution and overwrites through the multimode authentication and policy enforcement functionalities of MAF. The handler uses the following security-related files stored on the handheld device, which are also protected through the policy enforcement functionality of MAF:

- the 16-byte authentication key used to set up the secure channel of communication to the SMMC card – installed through security administration and accessed only by the handler
- the 16-byte MAC key used to set up the secure channel of communication to the XMC card – installed through security administration and accessed only by the handler
- the user's PIN – read by the handler, but not maintained in memory

- the user's X.509 certificate – captured from the token at enrolment, and accessed only by the handler
- the X.509 certificate of the root CA used to validate the user's certificate on the token – installed through security administration.

SCs such as the XMC are designed to resist tampering and monitoring of the card, including sophisticated attacks that involve reverse engineering, fault injection, and signal leakage. For this authentication mechanism, the card must avoid disclosing its base secret – the private key used to sign challenges it receives, the user PIN created at card personalisation time, and the Global Platform keys used for maintaining the secure channel with the device. The private key should be also used exclusively for authentication.

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 from the device to the card. The private key of the user and the user's PIN established during personalisation cannot be exported from the token.

The challenge-response mechanism specified in FIPS 196 is designed with measures to conceal the base secret used and avoid replay. The authentication of an entity depends on two things: the verification of the claimant's binding with its key pair, and the verification of the claimant's digital signature on the random number challenge. In signing the challenge and verifying the signature, the handler uses OpenSSL v0.9.7 APIs that comply with the PKCS No. 1 standard, while the applet uses an available on card function.

4 Bluetooth Smart Card (BSC) authentication

The BSC authentication mechanism, as with the SMC, relies on a SC chip packaged together with other components in a compact-size form factor, such as a key fob. However, rather than bringing a SC into physical contact with a handheld device, a Bluetooth wireless interface is used. Bluetooth is a short-range wireless communications protocol that operates in the globally available 2.4 GHz frequency band (McDermott-Wells, 2005). Many models of mobile devices are manufactured with built-in Bluetooth radios. SC authentication over Bluetooth can provide the security of SC-based authentication to a device with the following advantages:

- no need exists for a specialised SC 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
- if the device moves outside the proximity of the token (e.g., becomes forgotten or stolen), access is locked
- it can be discrete (i.e., non-discoverable by third parties).

A BSC token houses a Bluetooth radio, SC, 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 SC functionality. A BSC token could also be used with Bluetooth-enabled workstations. The mechanism is also amenable to other types of low-power PAN communications. While designed for handheld device authentication, the token could also be used for user authentication on other computing platforms, such as Bluetooth-enabled notebook computers.

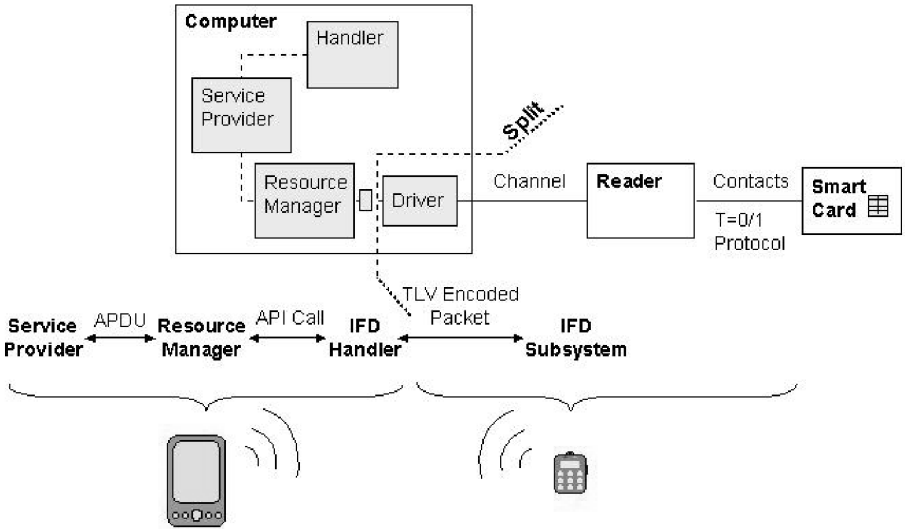
4.1 Operation

The BSC has many similarities with the SMC insofar as both solutions depend on the functionality of a Java Card-compliant SC 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 SC application. Figure 3 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 SC application: the service provider, a resource manager, and a driver for the SC reader. The software application, such as the handler, normally communicates with the driver indirectly via the service provider, which in turn uses the standardised 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 SC reader, while the service provider is closely tied to the characteristics of the SC, allowing an application to use any SC with any reader, provided that the respective service provider and driver software are available.

Figure 3 Reification of the PC/SC Lite architecture



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.

The arrangement allows the BSC solution to work with any type of Java Card-compliant SC recognised by the PC/SC Lite framework. For simplicity, however, XMCs were used in the token prototype.

4.2 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 SC reader), to allow data to be exchanged with a SC. 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 SC functionality over a Bluetooth channel. The new IFD handler for the BSC looks like a normal SC 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 serialisation 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:

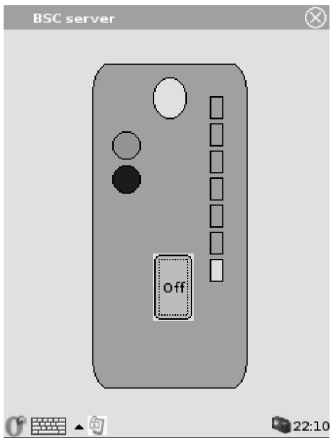
- if the argument is fixed length, it is directly appended
- if the argument is variable length, it is preceded by its length and then appended.

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

The token was implemented virtually as an OPIE application running on an iPAQ PDA. The virtual token appears on the PDA's display and functions as the real token would. A server module on the token handles key functions, including all of the SC and

Bluetooth socket interactions. Figure 4 shows a screen shot of a token in a key fob form factor.

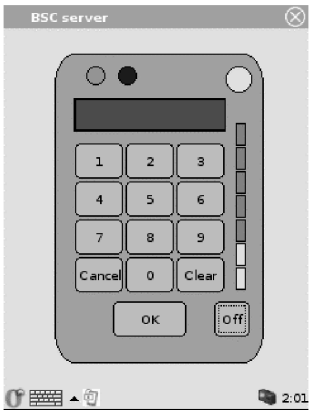
Figure 4 Bluetooth Smart Card (BSC) token



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 launches, which is indicated by the top circular LED at left becoming green. Once the server is operational, it represents its level of activity through the stacked LEDs at right.

Another variant of the token allows the user to enter the SC 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 the possibility of passkey entry for Bluetooth pairing in addition to PIN entry. The same server module is used by both tokens (i.e., with and without a PIN keypad). A flag (--nopin) tells the server whether or not it should run in remote PIN verification mode.

Figure 5 Token with Personal Identity Number (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 cancelled.

4.3 Safeguards

As with the SMC, both the device and SC and the communications between them are potential targets for the BSC. Because of the similarities in implementation, only the differences due mainly to the Bluetooth communications are discussed.

The device must be designed to resist tampering, as with the SMC. In addition to the security-related files inherited from the SMC handler, the BSC handler requires the following additional files on the handheld device, which must be protected through the policy enforcement functionality of MAF:

- *the token's Bluetooth address*: created during device pairing for exclusive use by the handler
- *the Bluetooth link keys*: created during device pairing for exclusive use by the handler.

The BSC token must also be made tamper resistant. This is facilitated by the use of a SC as its foundation.

Because the BSC solution uses a wireless radio-based communication channel, the following issues not present in the SMC must be dealt with:

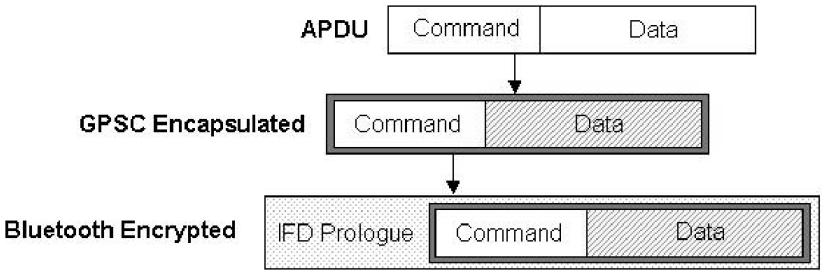
- an attacker can eavesdrop on the communication channel from a distance
- an attacker can send information to the device and token via an active Bluetooth interface to attempt to impersonate one or both parties, or to disrupt communications
- when the device selects a token with which to communicate, the device cannot be certain it contacted the user's token
- when a token is contacted, the token cannot be certain the device that contacted it is the device of its user.

The device and token must be paired to one another as part of the initial enrolment phase, when the token is first registered with the device. Bluetooth pairing establishes shared symmetric keys on both units, used to authenticate and encrypt exchanged L2CAP packets respectively via a MAC and stream cipher. Though security vulnerabilities have been noted in Bluetooth, risks and countermeasures have also been identified (Sun et al., 2001; BSI, 2003). Through continuous pairing (i.e., bonding), a trusted connection exists between the device and token 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.

Residual risks are addressed by sending vital information over the Bluetooth link within a secure channel. As with the SMC, pre-established symmetric keys (i.e., one for MAC, and one for Triple DES encryption) are used to protect transmitted APDUs using the Global Platform Secure Channel Protocol 01 format. Figure 6 illustrates the two

levels of confidentiality protection afforded through the secure channel encapsulation and Bluetooth security features.

Figure 6 Communication protection



Under the Secure Channel Protocol, the data portion of an APDU is encrypted. Encryption is applied unidirectionally from the device to the SC token. Encapsulated APDU commands and unencapsulated responses are encrypted using Bluetooth mechanisms. If needed, the host and card applications could be modified to encrypt all or parts of the response messages returned from the SC, but this is not currently part of the Global Platform Secure Channel 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.

5 Conclusions

While mobile devices provide productivity benefits, they also pose new risks. This paper demonstrates how SC authentication can be implemented to reduce them, using non-traditional form factors and interfaces. The approach provides users a simpler and less cumbersome way to interface SC functionality when compared with conventional types of SCs.

References

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

Bundesamt für Sicherheit in der Informationstechnik (BSI) (2003) *Bluetooth Threats and Security Measures*, November, http://www.bsi.de/english/publications/brosch/B05_bluetooth.pdf.

Husemann, D. (1999) 'The Smart Card: don't leave home without it', *Concurrency*, Vol. 7, No. 2, pp.24–27.

Jansen, W., Karygiannis, T., Iorga, M., Gavrila, S. and Korolev, V. (2003a) 'Security policy management for handheld devices', *Proceedings of the 2003 International Conference on Security and Management*, pp.199–204.

Jansen, W., Korolev, V., Gavrila, S., Heute, T. and Séveillac, C. (2003b) 'A framework for multi-mode authentication: overview and implementation guide', *NIST Interagency Report 7046*, pp.1–30.

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

McDermott-Wells, P. (2005) 'What is Bluetooth?', *Potentials*, Vol. 23, No. 5, pp.33–35.

National Institute of Standards and Technology (1997) 'Entity authentication using public key cryptography', *Federal Information Processing Standards Publication (FIPS PUB) 196*, US Department of Commerce, February 18, Gaithersburg, Maryland.

Ortiz, C.E. (2003) 'An introduction to java card technology', *Sun Developer Network*, May 29, http://developers.sun.com/techtopics/mobility/javacard/articles/javacard1.

PC/SC Workgroup (2005) 'Interoperability specification for ICCs and Personal Computer' systems', *Part 1 – Introduction and Architecture Overview*, June, http://www.pcscworkgroup.com/specifications/files/pcsc1_v2.01.0.pdf.

Sun, J., Howie, D., Koivisto, A. and Sauvola, J. (2001) 'Design, implementation, and evaluation of Bluetooth security', *Proceedings of the IEEE International Conference on Wireless LANs and Home Networks*, Singapore, pp.121–130.

Turban, E. and McElroy, D. (1998) 'Using Smart Cards in electronic commerce', *Proceedings of the Thirty-First Hawaii International Conference on Systems Science*, Hawaii, Vol. 4, pp.62–69.

Vedder, K. (1992) 'Smart Cards, computer systems and software engineering', *Proceedings of the IEEE CompEuro '92 Conference*, pp.630–635.

Note

[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.