J2ME Implementation of System for Storing and Accessing of Sensitive Data on Patient's Mobile Device

Wojciech M. Zabołotny[a] and Radosław Wielgórski[a] and Marcin Nowik[a]

[a]Institute of Electronic Systems, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa, Poland

ABSTRACT

This paper presents a system allowing to use a patient's mobile phone or PDA for storing of biomedical data, which then, during medical consultation or intervention may be used by the medical staff. The presented solution is aimed on providing both: reliable protection to sensitive patient's data, and easy access to information for authorized medical staff. In the presented system, data are stored in an encrypted form, and the encryption key is available only for authorized persons. The central authentication server verifies the current access rights of the person trying to obtain the information, before providing him or her with the key needed to access the patient's data. The key provided by the server is valid only for the particular device, which minimizes the risk of its misuse. For rare situations when no connection to the authentication server is available (e.g. intervention in the mountains or rural area), system assures an additional "emergency" method to access the encryption key in controlled, registered way. The system has been implemented in Java language and tested in the simulated environment provided by "Sun Java Wireless Toolkit for CLDC"[1]

Keywords: J2ME, sensitive data storage, biomedical data, encryption, mobile applications

1. INTRODUCTION

Mobile devices like cellular phones (CP) and personal digital assistants (PDA) are widely used by most people in the contemporary world. Current mobile devices offer good communication capabilities, significant computational power and support for high capacity storage like memory card. These features are combined with reasonable price, affordable due to mass scale production. Therefore the personal mobile devices are often proposed as the part of the biomedical monitoring systems[2] or personal medical data acquisition and management systems.[3] In most systems the acquired data are then transmitted to the server, which archives them and makes available to the authorized medical staff. However in some situations, such immediate transmission of the acquired data is not possible (e.g. due to lack of direct Internet connectivity and of GPRS/UMTS network coverage), inconvenient (e.g. due to overload of the central server) or simply too expensive. In this case a secure long term archiving of data on the patient's device may be needed.

The personal mobile device media may also be used to store the additional patient's information. The information stored may include:

- Chronic diseases (e.g. diabetes, hypertension, epilepsy)
- Allergy
- Medicines currently administered by different physicians)
- Information related to non-medical aspects of the therapy, e.g. the insurance policy

This information may be useful, when the patient (e.g. during the holidays, or business trip) has to visit another physician, than usual, or just in an emergency situation, when the additional information may help the rescue team to select the efficient and safe therapy. Especially in the case of emergency (e.g. after an accident) we should also address situations, when the patient's device could get damaged, switched off, or just discharged.

Further author information: (Send correspondence to W.M.Z.)
W.M.Z.: E-mail: wzab@ise.pw.edu.pl, Telephone: +48 22 234 7717

Therefore we have proposed the system, where the information is stored on a removable media, and may be accessed even when the device itself is not operable. The information may be accessed using just a removable media. Certainly if such a system should be deployed, the patient should clearly mark his/her mobile device in a special way, to express consent to access the removable media by the medical staff in the emergency situation.

The first theoretical concept of the system was presented on the conference IbiTel 2008.[4] In this paper we present the more mature concept of the system, which additionally has been implemented and tested on a simulated hardware. The main goal of this paper is to present the proposed system and to subject it to the public discussion. The authors believe, that in the data protection systems the openness and wide discussion is essential to discover possible flaws, before the system is widely deployed.

2. CONCEPT OF THE SYSTEM

2.1 Secure storing of sensitive data on the removable media

The removable media themselves usually do not offer protection of the stored data. Some exception exist like the Secure Digital (SD) memory cards,[5] but their protection system[6] is aimed on copy protection of commercially distributed multimedia, and as the technical details are not publicly available, it is not clear, if it could be used for protection of sensitive biomedical data.

To assure portability of the proposed system we have decided on a data protection which does not rely on any hardware–specific features. The data stored on the removable media are simply encrypted using a symmetric cipher. Access to the stored information does not allow to read the data, unless the user has the proper symmetric key (SK).

The only problem is how to make the symmetric key available only for the authorized medical staff.

In subsequent sections we assume, that the medical staff ("user" of the system) is equipped with the user terminal (PDA, laptop PC or a dedicated device) able to read the removable media taken away from the patient's mobile device.

2.2 Distribution of the symmetric key - standard server based approach

Using of single symmetric key for all devices is not acceptable from the security point of view. Each device should have its own individual symmetric key. To make this key available for the authorized users, this key should be also stored on the removable media, but in an encrypted form - encrypted with the server public key (SPuK).

In this scheme access to the data requires the following steps (see figure 1):

- The medical staff (user) reads the encrypted SK from the removable media
- The SK encrypted with SPuK is transmitted to the server, together with the user public key (UPuK), which unambiguously identifies the user.
- The server verifies that the owner of the transmitted UPuK is currently authorized to read the data protected with the transmitted encrypted SK
- If the user is authorized to access the data, server decrypts the SK with its server private key (SPrK), then encrypts it with the user's public key (UPuK), and sends it back to the user.
- The user decrypts received SK, using his/her user private key (UPrK)
- The user uses the obtained SK to access the data

When access to the data in this scheme is requested, the server checks current access rights of the user. If the user key gets compromised, its further use may be quickly prohibited by revoking it in the server's database. Please note, that the SK is always transmitted in the encrypted form.

Unfortunately, the above mechanism may be implemented only in some mobile devices. The problem is, that standard Java for mobile devices (J2ME with SATSA extensions) does not provide support for private keys and asymmetric (public key) decryption. It is possible to work around this problem using external libraries (e.g. the Bouncy Castle J2ME[7]), but if a solution fully compatible with J2ME is desired, slight modification is needed.

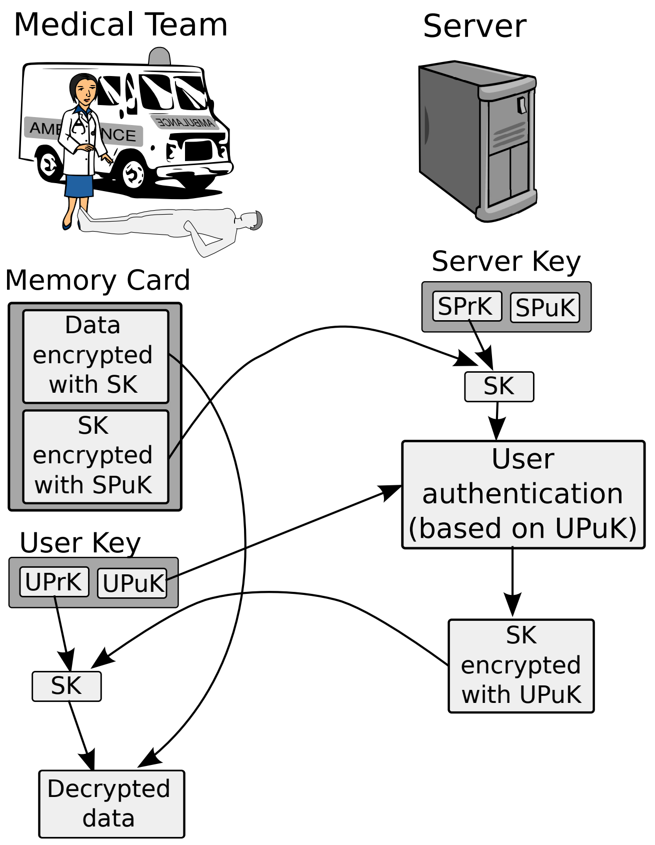

Figure 1. Distribution of the key in a system with standard, server based authorization.

2.3 Distribution of the symmetric key - modified server based approach

To avoid limitations of the SATSA package, the asymmetric user keys (UPuK and UPrK) must be replaced with the symmetric user key (USK). The information sent from the user terminal may be still encrypted with the server's public key (SPuK), but the information sent back from the server must be encrypted with the user key (USK). It means, that USK must be at least temporarily available to the server. To keep the system as secure as possible, we have decided to store only the hash of the user key in server's database. The USK itself (encrypted with SPuK) is sent together with encrypted SK, and used only for verification of user access rights and for encryption of SK. The scheme of modified key distribution is shown in the Fig.2 and is performed in following steps:

- The medical staff (user) reads the encrypted SK from the removable media
- The SK encrypted with SPuK is transmitted to the server, together with the user key (USK) which for safety is also encrypted with SPuK.
- The server decrypts USK, calculates its hash function and checks if the owner of that USK is authorized to read the data protected with the transmitted encrypted SK (for safety reason the database stores only hashes of user keys, not the keys themselves).
- If the user is authorized to access the data, server decrypts the SK with its server private key (SPrK), then encrypts it with the user's key (USK), and sends it back to the user.
- The server immediately forgets the received USK.

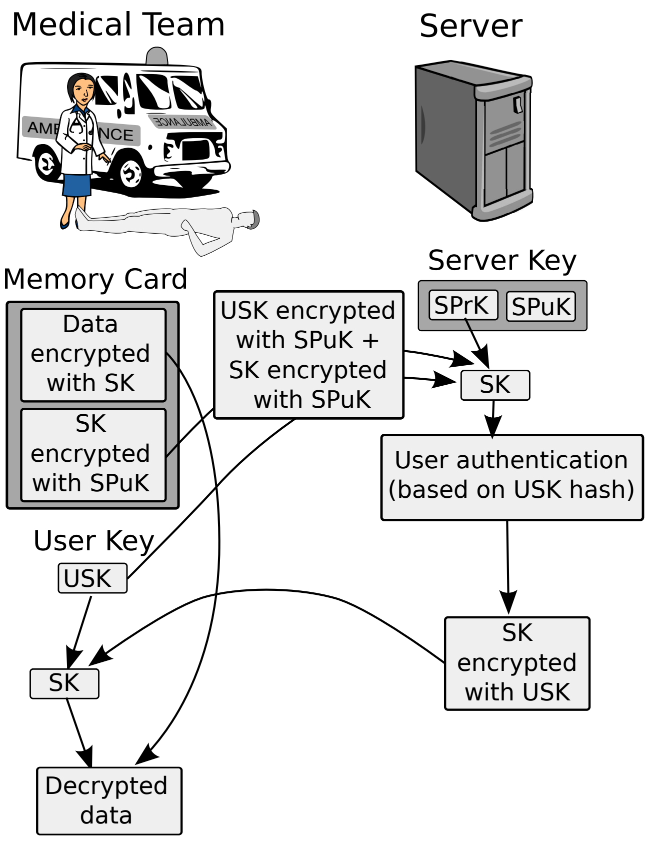

Figure 2. Distribution of the key in a system with server based authorization, modified for standard J2ME devices.

- The user decrypts received SK, using his/her user key (USK)
- The user uses the obtained SK to access the data

In fact both described systems of key distribution are just variations of standard solutions, widely used in different systems with server based authorization. However if only these schemes are used, the possibility to read the protected data fully depends on reliable access to the server. In our system we have also provided possibility to access the data when server is temporarily unavailable.

2.4 Distribution of the symmetric key, when server may be temporarily unavailable

Sometimes access to the data may be needed in areas without network connectivity (e.g. in the rural area, or in an emergency situation in the mountains). To enable the user to obtain the SK, even if the server is currently unavailable, we need to provide an additional channel for SK distribution. It may be achieved by storing another copy of the SK, encrypted with the so called "emergency public key" (EPuK) on the removable media. The authorized users should be equipped with the emergency private key (EPrK), however the use of this key should be registered. In the simplest approach the EPrK could be stored on the sealed media. In more sophisticated approach, the user terminal should record every decryption done with the EPrK.

Such a solution, however, introduces a serious weak point in the whole system. If the EPrK gets compromised, data on all patients' devices are not protected any more. To make the system immune for EPrK compromise, the emergency keys (EK, consisting of EPuK and EPrK) should have limited period of validity.

The system with the short term emergency keys should operate in the following way:

- The removable medium in the patient's device contains:
  - The data encrypted with the individual symmetric key (SK)
  - The symmetric key (SK) encrypted with the server public key (SPuK)
  - The symmetric key (SK) encrypted with the current emergency public key (EPuK)
- When the validity period of the current emergency key expires, the device replaces the SK encrypted with the old EPuK with the SK encrypted with the new EPuK. From this time the owner of the old EPrK is not able to access the data.

The described solution however could pose problems, when the real time clock in the client's device is not properly synchronized. To increase reliability, probably two emergency keys with overlapping validity periods should be used simultaneously.

2.5 Distribution of the short term emergency keys with poor network connectivity

Introduction of short term emergency keys allows to provide an additional channel for SK distribution without compromising of the system's safety. However functioning of such system still requires, that the patient's device is able to connect to the server and download the new EPuK, whenever the validity period of the old EK expires. Lack of the network connectivity will impair the process of renewal of the emergency keys.

However it is possible to generate the emergency keys in advance. The server will generate the emergency keys for a few validity periods in advance. The public keys (EPuK's) may be published immediately, but future private keys should be stored on the server and released to the authorized medical staff just before the beginning of their validity period. In this implementation the patient's device may rarely connect to the authorization server and download a list of future EPuK's.

The final version of the system based on short term emergency keys is shown in the figure 3.

2.6 Advanced functionalities

Both modes of operation (with server based authorization and with access based on the emergency keys) allow to implement highly granular access control.

It is possible, that the encrypted information record, containing the SK, may also contain additional information describing the information stored on the memory card. The server may verify if the particular user is authorized to access this particular data, before the SK encrypted with UPuK (or USK in the modified version) is sent back to the user.

In the emergency mode of operation, similar granularity may be achieved, if the server generates multiple lists of Emergency Keys (EK) - one list for each possible combination of access right. Then the patient's device may download the list of EPuK's appropriate for the type of information stored on the media. As the result, only the medical team provided with the corresponding EPrK (i.e. with the appropriate access rights) will be able to decrypt the data.

2.7 Long term safety of the patient's data

After any access to the patient's data the medical staff gains access to the patient's SK. This key may be used later to access this patient's data, bypassing the authorization system. The only viable solution would be to change the SK, but it requires recoding of all data with the new SK. Such operation is both time and power consuming. Good idea would be to change the SK, after the data have been accessed. This operation probably should be triggered by the patient himself.

Another solution may be possible, if we consider the fact, that the knowledge of the SK does not further affect security of the already recorded data. In fact the medical staff could just copy the decrypted data (we do not discuss the legal aspect of this operation). Therefore only the later recorded data should be encrypted with the new SK. Such implementation does not involve recoding of big amounts of data, but requires that the device should be able to store multiple symmetric keys in the encrypted form.

The problem could be also solved by using more advanced encryption engine, where the SK never leaves the engine. This solution will be mentioned later, when describing our test implementation.

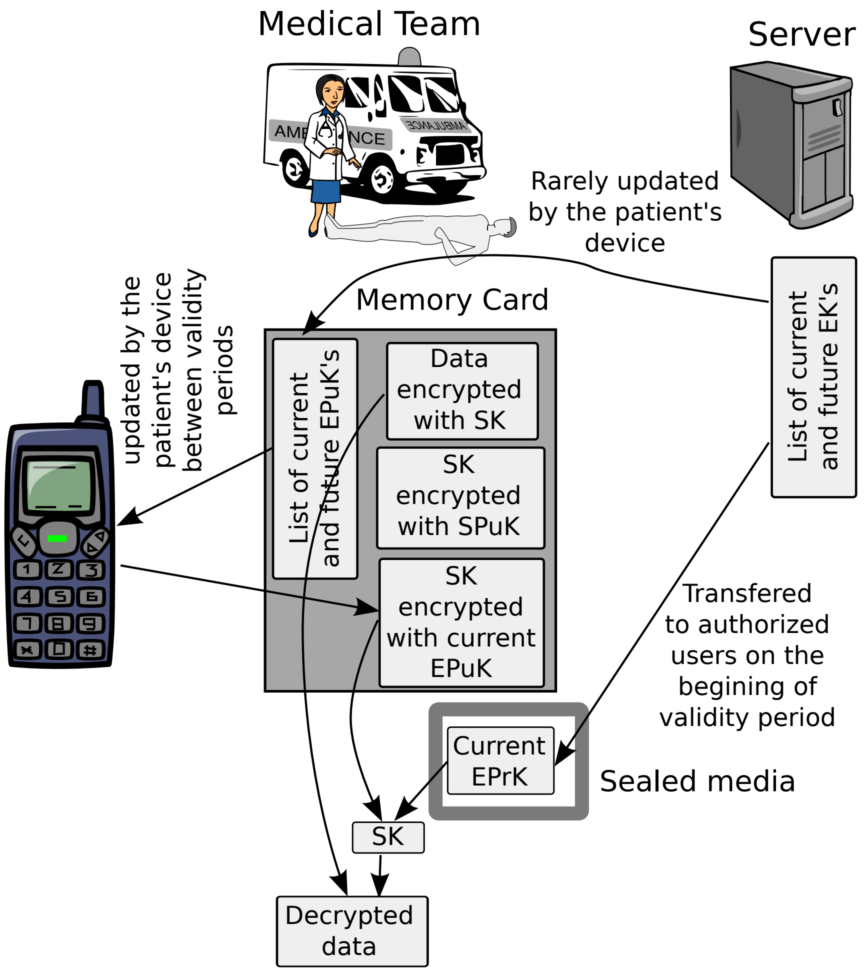
Medical Team
Server
Rarely updated by the patient's device
List of current and future EK's
updated by the patient's device between validity periods
Memory Card
List of current and future EPuK's
Data encrypted with SK
SK encrypted with SPuK
SK encrypted with current EPuK
Transfered to authorized users on the begining of validity period
Current EPrK
Sealed media
SK
Decrypted data

Figure 3. Distribution of the keys in the proposed system with short term emergency keys. This mode of operation may be used when network connectivity is at the moment unavailable.

3. TEST IMPLEMENTATION ON SIMULATED HARDWARE

The server part of the system was implemented on standard PC computer using Java Standard Edition (SE) and Apache Derby[8] database server. The server databases stored:

- the hashes of the user symmetric keys (USK) together with the assigned access rights
- the generated short-term emergency keys (public EPuK - to be distributed to the patient devices, and private EPrK - to be passed in a sealed form to the medical staff members).

The more complex part of our test system were the mobile applications for patient's device and for medical staff terminal. Those devices were implemented using the Wireless Toolkit (WTK)[1] emulator, and the software was written in Java 2 Micro Edition (J2ME).[9] The necessary cryptographic functions were provided with the JSR-177[10] "Security and Trust Services API for J2ME (SATSA)" package. The test implementation used symmetric cryptography (in ECB mode with 128-bit AES key, and in CBC mode with 128-bit AES key and 128-bit initialization vector) and asymmetric cryptography with 512-bit RSA keys. Generation of random keys and initialization vectors required a source of random information in mobile devices. As a source of information with relatively high entropy we used the audio input. The recorded sound was later converted into the random number using the MD5 Message Digest algorithm provided by SATSA. Sound recording functionality was provided by the JSR-135[11] "Mobile Media API" package.

Limitations of the SATSA package (namely – lack of possibility to decrypt the information encrypted with the public key) forced us to modify the operation of the system, as it was described in chapter 2.3. However implementation of the original version with external library – Bouncy Castle J2ME 1.45[7] was also successfully tested.

Storing of information on the external SD card was performed using the JSR-75[12] "PDA Optional Packages for the J2ME Platform (FileConnection)" package.

The patient's mobile application may be also used to collect the data from monitoring or measurement devices, to receive data from the physician's computer, or to transfer the data to the medical staff terminal without moving of the memory card - in this situations yet another optional J2ME package is needed: JSR-82[13] "Java APIs for Bluetooth". The Bluetooth connectivity was also implemented and tested.

The network connection to the server was provided using the standard J2ME packages: JSR-118[14] "Mobile Information Device Profile (MIDP 2.0)" and JSR-139[15] "Connected Limited Device Configuration (CLDC 1.1)". The secure connection was assured by the SSL v3.0 protocol. The server certificate was generated using the standard "keytool" tool available in Java SE, and imported into the mobile applications with the "mekeytool" available in the WTK.

Some activities performed by the patient's mobile application (e.g. refreshing of the list of emergency public keys, and re-encryption of SK with the new EPuK) may require periodical automatic activation of the application - this may be accomplished using the "Push Registry"[16] mechanism, which is part of the MIDP 2.0 profile.

The main security issue in our test implementation was the handling of the symmetric key (SK) in the patient's device. The symmetric key SK must be available for the patient's mobile application, as it used for:

- storing of new patients data
- storing of SK encrypted with the new EPuK, whenever an outdated emergency key must be replaced with the new one.

Therefore storing of SK in the encrypted form in the Memory Card was not sufficient. The original form of SK had to be safely kept, available for the patients application, but not easily accessible from outside. We implemented this functionality with the J2ME Record Management Store,[17] however this is not fully safe solution. Other, much safer approach would require use of the SIM card based on the JavaCard[18–20] technology. [*]

[*] In fact use of JavaCard could improve security of the whole system even further. All encryption and decryption could be performed in a user application located in the JavaCard. In this case the symmetric key may be never accessible from the outside of the card. Therefore the problems described in section 2.7 would not exist any more. Instead of the encrypted SK, we should store on the removable media an encrypted "activation password", which allows to use the encryption/decryption engine. This "activation password" could be

The tests performed in the described simulated environment shown correct operation of the system proving correctness of the concept of the system and of its implementation.

Even though the tests of mobile parts were performed in the simulated hardware, the tested applications may be ported to the real hardware with only minimal modifications (e.g. optimization of GUI).

Please note, that the JSR packages used in our test implementation are available in the most J2ME enabled mobile phones available today. Additionally the measurements of resources consumption during the tests in the WTK shown, that the usage of RAM by the mobile application never exceeds 2MB - so the application should be able to run even on mobile phones with quite moderate (however not the the minimal acceptable) parameters.

4. SUMMARY

The system based on concept presented in this paper allows for safe storing of sensitive biomedical data on removable media in mobile device.

The stored data may be accessed only by authorized medical staff even if the mobile device is discharged or damaged, and even if no connection to authorization server is available.

The system may be implemented in Java with the currently available and relatively cheap technology. The correctness of the system has been proven in simulated environment and the tested implementation may be easily moved to the real hardware.

REFERENCES

[1] "Sun java wireless toolkit for CLDC." `http://www.oracle.com/technetwork/java/index-jsp-137162.html`.

[2] Zhang, P., Kogure, Y., Matsuoka, H., Akutagawa, M., Kinouchi, Y., and Zhang, Q., "A remote patient monitoring system using a java-enabled 3g mobile phone," in [*Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*], 3713–3716 (2007).

[3] Pinsker, M., Schindler, K., Morak, J., Hayn, D., Kastner, P., Riedl, M., Ludvik, B., and Schreier, G., "Experiences using mobile phones as patient-terminal for telemedical home care and therapy monitoring of patients suffering from chronic diseases," 1305–1312 (2008).

[4] "Conference on biomedical engineering and telemedicine." `http://ibitel.elka.pw.edu.pl/` (9 2008).

[5] "SD association." `http://www.sdcard.org/home/`.

[6] "SDSD-CPRM flexible protection for digital content." `http://www.4centity.com/docs/SDSD-CPRM_WP_R2-121107.pdf` (12 2007).

[7] "The legion of the bouncy castle." `http://www.bouncycastle.org`.

[8] "Apache derby." `http://db.apache.org/derby/`.

[9] "About java for mobile devices." `http://www.oracle.com/technetwork/java/javame/javamobile/overview/about/index.html`.

[10] "Security and trust services API for J2ME (SATSA); jsr 177." `http://java.sun.com/products/satsa/`.

[11] "Mobile media API (JSR-135)." `http://download.oracle.com/javame/config/cldc/opt-pkgs/api/mm/jsr135/index.html`.

[12] "Getting started with the fileconnection APIs." `http://developers.sun.com/mobility/apis/articles/fileconnection/` (2004).

[13] "Jsr-000082 javaTM APIs for bluetooth (final release)." `http://www.jcp.org/aboutJava/communityprocess/final/jsr082/`.

[14] "JSR-000118 mobile information device profile 2.0 (final release)." `http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html`.

[15] "JSR-000139 connected limited device configuration 1.1 (maintenance release)." `http://jcp.org/aboutJava/communityprocess/mrel/jsr139/index.html`.

changed periodically without the need to re-encrypt all the data with a new SK. However in this implementation providing of access to the data, when the patient's device is damaged or discharged, requires moving both memory card and the SIM card into the user terminal. Additionally most GSM/UMTS providers do not offer SIM JavaCards with possibility to install user applications, and therefore this solution has not been tested yet.

[16] Ortiz, E., “The MIDP 2.0 push registry.” `http://developers.sun.com/mobility/midp/articles/pushreg/` (1 2003).
[17] Ghosh, S., “J2ME record management store.” `http://www.ibm.com/developerworks/library/wi-rms/` (5 2002).
[18] “Understanding java card 2.0.” `http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html` (3 1998).
[19] “Java card platform specification 2.2.2.” `http://java.sun.com/javacard/specs.html`.
[20] “Java card 3.0 platform specification.” `http://java.sun.com/javacard/3.0/specs.jsp`.