Secure Middleware for Mobile Phones and UICC Applications

Ioannis Kounelis[1], Hao Zhao[2], and Sead Muftic[2]

[1] Institute for the Protection and Security of the Citizen (IPSC)
Joint Research Centre – European Commission, Ispra (Va), Italy
[2] School of Information and Communication Technology
Royal Institute of Technology (KTH), Stockholm, Sweden
{kounelis,hzhao,sead}@kth.se

Abstract. In this paper we describe our concept, design and current prototype implementation of a new middleware for mobile phones and UICC. The purpose of the middleware is to be used as an interface between applications, loaded in mobile phones, and functionalities of the corresponding supporting modules (applets) stored in UICC. At the moment, our middleware supports only security and mobile payment functions. Our primary goal was to explore the features that multi–application chips provide and to create a new way for handling of sensitive information when stored and used in mobile phones. Another goal is to extend the middleware to hide technology details of underlying UICC and their applets, so that applications developed on the top of the middleware are independent of the underlying mobile phone technologies. We plan to extend the current version of our middleware module to be used with other UICC applications and alternative mobile operating systems.

Keywords: UICC, middleware, mobile phones.

1 Introduction and Goals

In recent years mobile phones have become a necessity in our everyday life. Since the number of mobile phones used globally exceeded 5 billion in July 2010 [1], one can understand the importance of mobile technologies. Mobile phones have evolved a lot since their first release and nowadays they are not just communication devices to make calls and send SMS messages.

A significant number of today's mobile devices has a complete operating system enabling them to execute all sorts of applications. From Symbian and BlackBerry's O.S. to the iPhone's O.S. and Android, mobile devices provide powerful platforms for communications, but also for execution of applications. Browsing the Internet is simpler than ever. With GPRS/3G/4G and wireless Internet connections, mobile phones can easily connect to the Internet and enhance browsing experience.

However, along with the new capabilities and opportunities come extended needs and demands, especially for security. One of the first and the most important is secure storage of data in mobile phones. At the moment, most of the applications loaded and executed in mobile devices, regardless of the phone's manufacturer, use the phone's memory (or memory card) to store data. It is similar to the use of a hard disk in PCs. However, the trend is to store data and use security functions based on capabilities of new, Javacard chips in mobile phones, called Universal Integrated Circuit Card (UICC) [9].

One of the goals of our research was to design and implement middleware that will provide to mobile applications a new, safer and more secure way of handling data operations, by using UICC [12]. Therefore, our middleware is a layered structure, at the top providing high–level Application Programming Interfaces (APIs) to mobile phone applications and at the bottom level communicating with UICC applets based on Application Programming Data Units (APDUs).

Besides secure storage, functionalities are also provided by the UICC to enable mobile applications to securely execute their functions and use their internal data. Such functionalities include financial transactions (credit card payments, electronic cash, etc.), security enhancements (private and public key generation, storage and verification of certificates, encryption, etc.) and administrative commands.

2 Interactions with Mobile Applications

High level applications, which are available to mobile phone users, use high-level programming APIs provided by the upper-end of the middleware. On the other hand UICC applets "understand" only APDU commands. Therefore, in order to make communications between mobile applications and UICC applet as smooth as possible, we introduced the middleware layer.

The middleware provides high level APIs to mobile application developers. These APIs do not require from application developers any knowledge of UICC internal functions and therefore they can be easily used through corresponding APIs. Moreover, the middleware communicates with UICC and applets using APDUs. So, middleware in fact performs communications between the mobile applications and the UICC by "translating" application-enabled APIs to card-level APDUs.

There are two basic approaches to implement the middleware. The first one is to extend an existing application with some extra features that will provide this functionality. The second approach is to create a separate module – the middleware that will provide communication between an application, which wants to use UICC functions and data storage in chip, and the UICC itself.

2.1 Middleware as a Separate Module

In this case the middleware is designed and implemented as a module separate from any application. It runs simultaneously with the application that use UICC functionalities. This means that there are two different layers in which the middleware provides communication. The upper layer deals with communication between the application and the middleware, while the lower layer deals with communication between middleware and UICC and its applets.

When considering communication at the upper layer, we actually need two functionalities:

- To pass data, sent to the UICC, between the application and the middleware or to execute some function of the underlying applets, in a secure way
- To receive feedback, positive or negative, of the communication process, i.e. whether data was successfully sent to the UICC or not and the result of the invoked operation.

At the lower layer we handle APDU commands exchanged with the UICC and its applets. In order to do that, an application must first create a connection with the UICC in order to communicate with an applet. This connection supports exchange of APDU messages in a format standardized by ISO 7816-4 [2]. Each connection has a logical channel reserved for its use. Our current implementation supports up to twenty logical channels for communication with the UICC chip, so our middleware truly supports multi–application UICC [8, 9]. The details of upper and lower layer communications depend also on the operating system of the mobile phone.

2.2 Middleware Integrated with an Application

An alternative approach is to create the middleware not as a separate module, but as an extension of an application that uses it. This means that the application calls middleware's APIs internally without using external invocation methods. The advantage of this approach is that upper layer communication and exchange of data between the application and the middleware can be done directly without the need to adjust to the mobile phone's operating system. As for the communication towards UICC, the approach is exactly the same as when the middleware is used as a separate module.

This approach makes the communication between the middleware and the application much easier to implement. No extra connection is needed between the two entities and transfer of data towards the UICC is performed smoothly. Moreover, since the middleware is integrated into the application, it can be completely adjusted to the application's specific needs and as a result, it is more efficient and effective during its use and operations.

3 Design Principles

Designing the middleware requires a different approach than creating a standard application. As it is intended to be used by mobile phone application developers, it does not provide any user interface, but it needs availability to easily connect to and communicate with other mobile applications. Therefore, some requirements are immediately obvious from this approach.

Developers do not need to know how the middleware works. It is not necessary to have knowledge of the internal logic of the middleware and of the way it implements its internal methods. This new way of using UICC and their applets should not change the way the user interacts with a phone. As Saltzer and Schroeder's Psychological Acceptability design principle suggests, if the new security features change the way of interaction and make it harder, users will simply avoid using them [3].

Moreover, the process must be transparent to the upper layer applications. The upper layer should not understand any difference between receiving and storing data to the UICC and manipulating data internally by its own applets. As a result, the middleware must provide friendly, easy to use and simple interface especially at the upper, API level.

The middleware must have no semantic knowledge of any data that it parses and passes in both directions. It must treat all data in the same way and not be dependent on different types of variables and values. This enables the middleware to handle data which are encrypted at the upper layer. Nonetheless it should have no knowledge of any passwords or keys that are used for authentication or cryptography. Implementing the middleware in this way makes it much more secure, flexible and easy to use.

Another important property of the middleware is to try to access and use UICC as infrequently as possible. UICC’s lifetime is highly dependent on the frequency of its use. Although new UICCs tend to have bigger endurance to read/write cycles (100,000 cycles for [14]) we consider essential that the middleware should be designed in such a way that it reads and stores data on the chip as few times as possible.

4 Implementation Example

Within this research and development, we not only designed our middleware, but we also implemented and tested a prototype version of it. As our tool for designing and developing the middleware we used Java ME. Java ME is the most used independent development and operational platform. It is currently used in over 8000 devices and it is supported by many different vendors (Nokia, Sony Ericsson, LG, Samsung, and many others) [4]. Moreover, it has the Security and Trust Services API (SATSA), which enables communication with the UICC with the use of APDUs [5].

In order to test the middleware in a real case scenario we used our Secure Mobile Wallet application [6]. We extended the Wallet with middleware functionality as described in section 2.2. As a result, interactions between the Wallet and the middleware are performed internally. On the lower layer, communication towards the UICC is achieved with the use of the SATSA-APDU package.

An example of how the middleware is used can be seen in Figure 1. An upper layer application wants to retrieve some values from the UICC. In order to do so, it calls the corresponding methods from the middleware. The first time the middleware is called it establishes a connection with the UICC and leaves it open throughout the execution time of the upper layer application. Afterwards, since the connection is open, the middleware communicates with the UICC and fetches the data buffer, which contains the values that are requested. Then it parses the buffer and returns to the upper layer application the correct value.

In our implementation we provide the following API categories to the upper layer: Identification Data, Financial Data, Mobile Application Data, Security Data, System Data, PIN Commands and Middleware Operations. Each category further consists of a different set of buffers which store corresponding data. For example, the Identification Data category consists of the Wallet Owner, the Wallet Issuer and the SAFE System data buffers. Each of these buffers handles specific values, like the ones shown in figure 1. Each buffer has a different access method that can require authentication before giving read/write rights to the upper layer. Therefore strong access control can also be enforced.

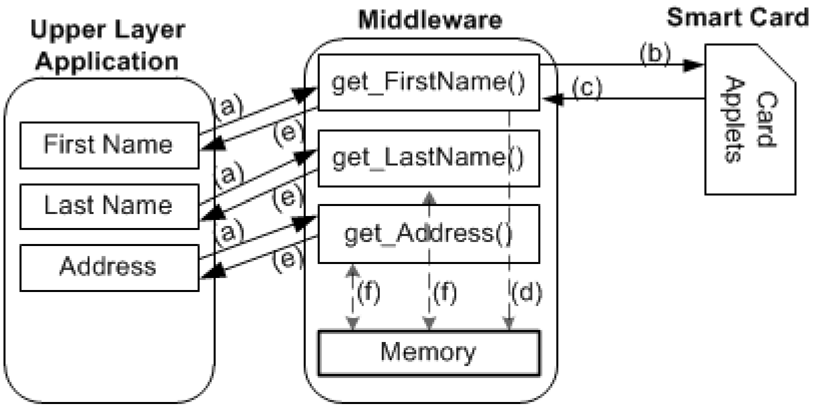

Fig. 1. An example use of the middleware. The upper layer application wants to retrieve three attributes from the UICC. To do so it calls the corresponding methods from the middleware. a) Call corresponding middleware method. b) Request data buffer which contains the desired variable. c) Send the requested buffer. d) Save the buffer in the memory. e) Parse the buffer, return the value. f) Read the buffer from the memory

For our middleware we introduced internal working memory. So, when the middleware reads a buffer from the UICC, it stores it temporary in its memory. As a result, when the middleware needs to fetch another value that belongs to the same buffer, it will first check if it has already stored that buffer in its memory and then read the data from there. This leads to a significant decrease in the number of transactions with the UICC and hence makes the whole process a lot faster to complete while at the same time it extends the UICC's lifetime.

The decision of how to manipulate the memory is left to the mobile application developers. For example, they may choose not to use the memory at all or use it as much as possible in order to minimize the transactions with the UICC. This can depend on the hardware specifications of the UICC. If the UICC can support a very large number of read/write cycles, [13] for instance supports 500,000 cycles, use of memory buffers may not be needed.

5 Interactions with UICC

UICC is the new generation of chips used for mobile communications, usually known as Subscriber Identity Module (SIM). It is introduced by organizations like ETSI, ISO and GSM that promote next generation of SIM chips as an open and service-oriented platform. UICC is no longer a closed environment and technology, but a multi-application hardware and software environment [7]. It allows other applications, besides standard SIM application, to reside and run on the same chip. SIM and other applications are parallel and managed simultaneously by the card operating system, i.e. Global Platform [8]. Figure 2 shows the infrastructure of the UICC [9].

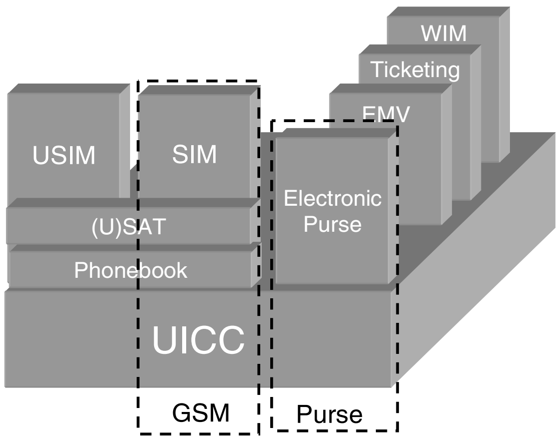

Fig. 2. The Internal Structure of UICC and Its Applications (adopted from [9])

Another significant innovation of UICC technologies is that they are now enabled by the Java virtual machine. So, all UICC internal applications are developed using Javacard Framework, which is much smaller and more compact compared with other platforms, for instance Java EE. These on-card Java applications are called Javacard applets. Even though the on-card Javacard language and environment is a subset of full Java language and environment, it integrates some other technologies specially created for SIM chips, called SIM Toolkit. The SIM Toolkit is a set of international standardized commands. Using it Javacard applets can construct their own Graphical User Interface [10]. Javacard technology and SIM Toolkit make it easier to develop a complete application for UICCs.

Javacard applets reside on the UICC and they are accessed by the middleware. Actually, Javacard applets stored in UICC support and respond to the middleware's requests. The middleware provides high level APIs to upper layer developers by "translating" them into sequences of corresponding APDUs. After that, it sends those APDUs to Javacard applets. All APDUs are predefined and standardized. In our implementation, we have two types of APDUs: for standard functions of Javacard applets we use standard APDUs [8]. However, for our Wallet application that uses our Wallet applet, there are no standard APDUs, so we have created our own APDUs supporting secure mobile payment functions and data handling. Only the middleware and the specific applet loaded in the UICC understand the meaning of these APDU commands, so they can successfully communicate with each other. When the UICC receives an APDU, the command is analyzed by the on-card operating system and then the corresponding applet is activated. The applet executes specific function that the APDU requires and sends back the response. The whole process is completed within the UICC; data are passed back to the middleware through a secure channel, so UICC provides support to and guarantee full data security. The complete process is shown in Figure 3 (adopted from [11]).

The secure channel that is created between the UICC and the reader (handset) is based on authentication using each one's set of Master keys. The middleware residing in the handset uses connection tools (SATSA for example) sending APDUs to UICC chips. The master keys of the chip are pre-set in the connection tools. During the authentication the security requirements, i.e. encryption, integrity, are negotiated. Finally, with a set of temporary keys, deriving from the Master keys, the session keys are established. This whole process is transparent to the upper layer and is described in the GlobalPlatform Card Specification [8].

There are two different approaches in order to load the applet into the UICC. The first one is to load the applet before the UICC SIM card is removed from the plastic housing. The application is loaded into the card with other applications, i.e., communication applications. This is done by the network operator during the chip personalization process.

The second approach is to load the applet using the OTA protocol. OTA protocol is designed to load and manage applications in UICC modules after they are delivered to subscribers. The back server sends a set of commands to the UICC module in order to update either the applications or the data. OTA protocol provides the possibility for service providers to maintain their applications OTA, whenever it is necessary.

To implement the OTA protocol, several issues should be solved. These are: over-the-air barrier to transport applications; methods to update applications and data in UICC modules; approaches to manage applications remotely. The solutions for these issues are all standardized. The barrier used to transport applications can be standardized message services like SMS. 3GPP TS 11.11 specifies commands for UICC modules that are operating files and if these commands are carried in class 2 SMS messages, the UICC modules can recognize and execute them. Global Platform Card Specification V2.2 [8] specifies the administration operations to the applications. To prevent the OTA administration approach being used illegally, 3GPP TS 03.48 specifies the security mechanisms.

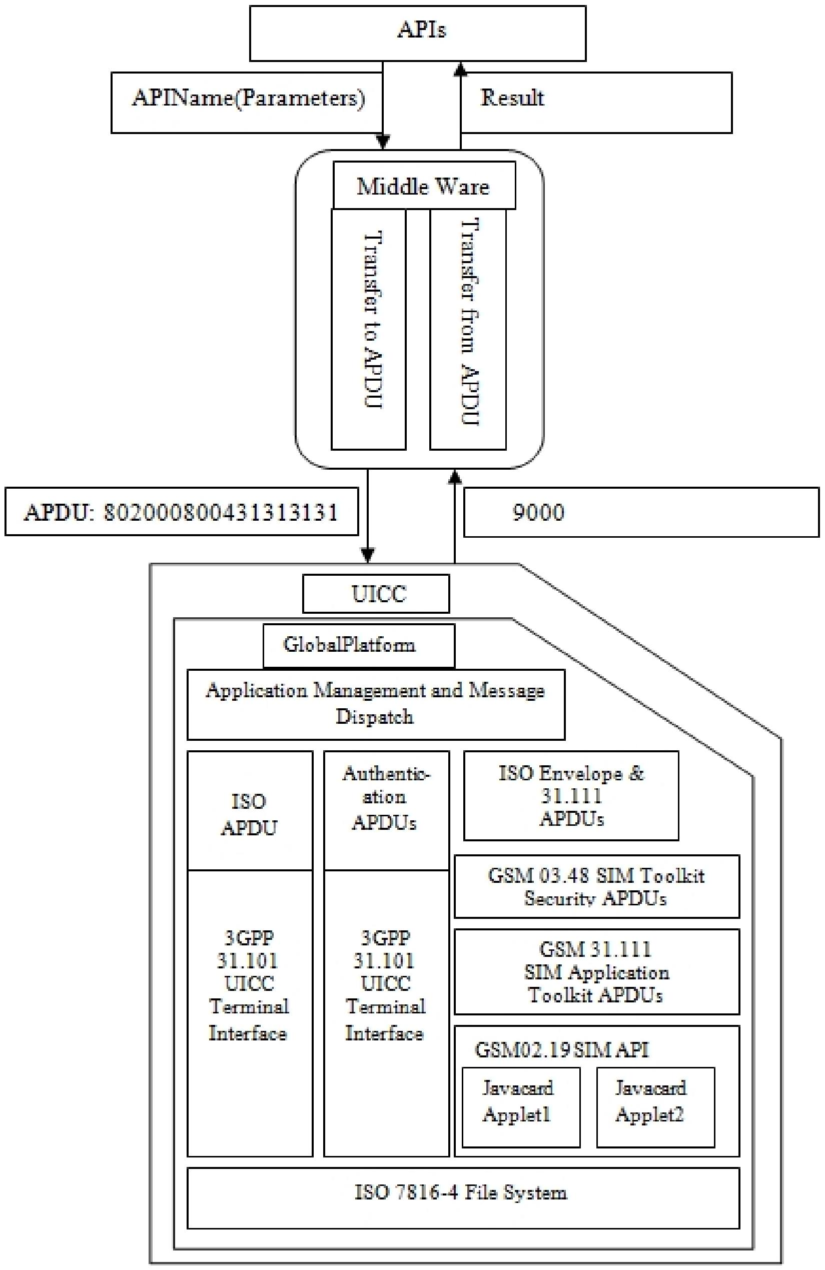
APIs
APIName(Parameters)
Result
Middle Ware
Transfer to APDU
Transfer from APDU
APDU: 8020008004313131 31
9000
UICC
GlobalPlatform
Application Management and Message Dispatch
ISO APDU
Authentic-ation APDUs
ISO Envelope & 31.111 APDUs
GSM 03.48 SIM Toolkit Security APDUs
3GPP 31.101 UICC Terminal Interface
3GPP 31.101 UICC Terminal Interface
GSM 31.111 SIM Application Toolkit APDUs
GSM02.19 SIM API
Javacard Applet1
Javacard Applet2
ISO 7816-4 File System

Fig. 3. The Process of Exchanging Data between the Upper Layer and the UICC

6 Conclusions

With the design and even prototype implementation of our middleware, we are now able to store data in a secure way on the UICC module and retrieve it with ease. We also implemented many functions and data handling operations for secure mobile financial transactions. The process of using those APIs is transparent to upper layer applications what makes the middleware very easy to use. No extra knowledge is needed in order to take advantage of its capabilities. Moreover, the whole process is as “light”, from a computational point of view, as possible, so that no extra hardware capabilities are needed.

Users of mobile phone applications that are based on our middleware can now securely store data on their UICC and take advantage of the capabilities it provides. They can be confident that data cannot be retrieved by an unauthorized person. In addition, they can take advantage of the UICCs mobility. Namely, users can transfer their UICC to another phone and all the data will be available to them as if nothing changed. This gives users the possibility to enjoy true mobility of their mobile phone applications.

7 Future Research

During this research we implemented the middleware using Java ME which was built into an existing application (Secure Mobile Wallet). Further research, however, could involve the process of making the middleware totally independent from any specific mobile phone application. It should be created as a standalone functional module that communicates with other applications in a secure way and then stores and retrieves data from the UICC.

Moreover, besides secure storage, various additional functionalities should also be added. The middleware should be able to provide an interface for all the functionalities that are supported by all applets stored on the UICC. It should be also possible to support over–the–air applet management functions, such as the possibility to load and unload new applets or to check the available memory.

Finally, further research could also focus on implementing our middleware for other platforms. Although Java ME is the most ubiquitous application platform for mobile devices, the fast spread of Android, as well as the already popular iPhone O.S., makes the need of creating a middleware that is able to run with those operating systems absolutely necessary. Unfortunately, by the time this paper was written, there were no standardized APIs for the above operating systems supporting exchange of APDUs with a UICC. However, it is most likely that such APIs will be soon available to developers.

References

[1] Wireless Intelligence, `http://www.wirelessintelligence.com` (accessed January 26, 2011)

[2] International Standards Organization (ISO): Identification cards – Integrated Circuit Cards – Part 4: Organization, security and commands for interchange. International Standards Organization, ISO 7816-4 (2005)

[3] Bishop, M.: Computer Security: Art and Science, pp. 348–349. Addison-Wesley, Boston (2003)

[4] Oracle: Java Platform Micro Edition Software Development Kit 3.0 for Windows, http://www.oracle.com/technetwork/java/javame/downloads/sdk30-jsp-139759.html (accessed January 26, 2011)

[5] Java Community Process: Security and Trust Services API (SATSA) for JAVATM 2 Platform, Micro Edition. Java Community Process, JSR-177 (2007)

[6] Zhang, F.: Secure Applications for Financial Environments (SAFE) System. Licentiate Thesis Report, pp. 66–73 (2010)

[7] European Telecommunications Standards Institute (ETSI): UICCs: Secured Packet Structure for UICC-based Applications. ETSI TS 102.225 V7, http://www.etsi.org (accessed September 21, 2010)

[8] Global Platform: Global Platform Card Specifications, version 2.2, http://www.golbalplatform.org/secificationscard.asp (accessed September 23, 2009)

[9] Lenhart, G.: The UICC Platform. ETSI Technical Committee UICC Platform, http://portal.etsi.org/scp/summary.asp (accessed September 23, 2009)

[10] Guthery, S.B., Cronin, M.J.: Mobile Application Development with SMS and the SIM Toolkit, pp. 131–155. McGraw-Hill Professional (2001)

[11] Guthery, S.B., Cronin, M.J.: Mobile Application Development with SMS and the SIM Toolkit, p. 117. McGraw-Hill Professional (2001)

[12] Kounelis, I.: Design and Implementation of Secure Mobile Phone Middleware for UICC Chips. M.Sc. Thesis, ICT/KTH (June 2010)

[13] STMicroelectronics: ST21Y144, Smartcard MCU with 114 Kbytes High Density EEPROM (March 2007), http://www.st.com (accessed May 5, 2011)

[14] Shanghai Huanhong Integrated Circuit Co.: SHC 1206, http://www.arm.com (accessed: May 5, 2011)